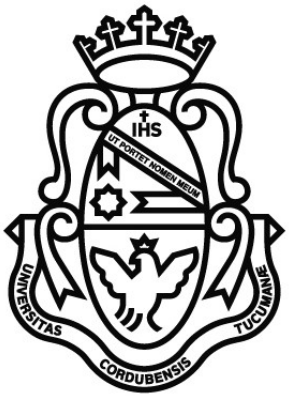
Universidad Nacional de Córdoba
Facultad de Matemática, Astronomía, Física y Computación



Tesis de Doctorado

Estudio de métodos semisupervisados para la desambiguación de sentidos verbales del español

Autor:
Lic. Cristian Cardellino

Directora:
Dra. Laura Alonso Alemany

Trabajo de tesis presentado para obtener el título de Doctor en Ciencias de la Computación

en el

Grupo de Procesamiento de Lenguaje Natural

20 de Abril de 2018

«The aim of a PhD's to ensure that no one, including your advisor, understands what you're doing after the first couple of years.»

China Miéville

Resumen

Esta tesis explora el uso de *técnicas semisupervisadas para la desambiguación de sentidos verbales del español*. El objetivo es el estudio de como la información de datos no etiquetados, que son mayores en tamaño, puede ayudar a un clasificador entrenado desde un conjunto de datos etiquetados pequeño.

La desambiguación de sentidos es una tarea crucial para muchas aplicaciones como son la traducción automática o la extracción de información. Sin embargo todavía es un problema abierto, especialmente la desambiguación de sentidos verbales, debido a la complejidad inherente de los verbos y su polisemia. Esto, sumado al hecho de que el español es un lenguaje con muchos menos recursos que el inglés, hace de la desambiguación de sentidos verbales para el español un área de estudio interesante a la cual contribuir.

La estructura de esta tesis es la siguiente:

En primer lugar hago un estudio minucioso de métodos puramente supervisados y las falencias que estos conllevan. Encuentro que el mayor defecto de los enfoques supervisados son la falta de cobertura y la tendencia a sobreajustar. Entonces, intento superar estos problemas con distintas técnicas semisupervisadas.

Continuo explorando el uso de los *word embeddings* (a veces traducidos como vectores de palabras) como representaciones no supervisadas dentro de clasificadores supervisados. Este enfoque es conocido como *aprendizaje semisupervisado disjunto*: una técnica no supervisada es usada para ayudar a mejorar una técnica supervisada. Los word embeddings ayudan a mejorar la falta de cobertura y la tendencia a sobreajustar, incluso a expensas de perder desempeño en otros aspectos, como la exactitud o el F1-score.

Continúo con la exploración del algoritmo de *autoaprendizaje*, un algoritmo de *aprendizaje automático conjunto* que usa un clasificador supervisado, entrenado en un conjunto de datos anotados inicial, para seleccionar nuevas instancias de un grupo de datos no anotados. Utilizando alguna medida de certeza, el clasificador anota dichas instancias sobre las que tiene mayor confianza y las usa para expandir los datos anotado y reentrenar el clasificador supervisado. Encontré que el autoaprendizaje sufre mucho con los conjuntos de datos desbalanceados. El algoritmo amplifica el sesgo hacia la clase mayoritaria, por lo que el crecimiento de cobertura esperado al agregar nuevos ejemplos al conjunto de entrenamiento es contrarrestado porque los límites de decisión entre las clases se comienzan a confundir.

De manera complementaria al autoaprendizaje, también exploro el *aprendizaje activo*. Este enfoque utiliza un modelo supervisado para seleccionar instancias tomadas de un grupo de datos no anotados, basadas en el impacto que dichas instancias pueden tener en el modelo, y se las da a un experto de dominio humano para que las anote. Muestro que la información obtenida mediante aprendizaje activo, aunque pequeña en comparación, puede superar el desempeño del autoaprendizaje.

Finalmente exploro las *redes neuronales en escalera*, un tipo de red neuronal artificial que minimiza una función de costo combinada. Esta función de costo combinada es una suma de una función de costo supervisada y una función de costo no supervisada. Este algoritmo logra, mediante dicha función combinada, superar el problema de la tendencia a sobreajustar de los métodos supervisados. Asimismo, el desempeño de este método es comparable o incluso mejor al resto de los métodos presentados en esta tesis, haciendo las redes en escalera un candidato muy interesante para el aprendizaje semisupervisado.

Clasificación (ACM CCS 2012)

- **Computing methodologies∼Semi-supervised learning settings**
- **Computing methodologies∼Natural language processing**
- Computing methodologies∼Artificial intelligence

Palabras Clave

Español: Inteligencia artificial – Procesamiento de lenguaje natural – Desambiguación de sentidos – Desambiguación de verbos del español – Aprendizaje automático – Aprendizaje semisupervisado – Aprendizaje profundo.

Inglés: Artificial intelligence – Natural language processing – Word sense disambiguation – Spanish verb sense disambiguation – Machine learning – Semi-supervised learning – Deep learning.

Agradecimientos

Esta tesis es fruto del trabajo y la ayuda de más gente que la que pueda nombrar en una sóla página. Pero haré lo posible para agradecer a todos los que participaron en esta travesía.

Empezaré por mi familia, la que me vió crecer desde chico pero también esa que formé. Mis viejos, Liliana y Oclides, que tanto me apoyaron y me bancaron (y nunca dejaron de hacerlo). Ellos son los responsables de que haya elegido este camino del estudio, porque desde muy chico me inculcaron la pasión por el aprendizaje y el conocimiento, por la curiosidad y la imaginación. Fueron mis padres los que me insistieron que la educación es el único camino a la verdadera libertad, tanto física como metafísica. Me enseñaron la disciplina de sentarme a estudiar y saber esforzarme para lograr mis objetivos. Gracias a su educación tengo hoy en día la integridad con la que pude lograr mis metas. También le agradezco a mi hermano Fernando que, aunque de vez en cuando peleamos, sé que en él siempre hay un oído confiable y una voz razonable que me puede escuchar y aconsejar.

A Vale y a Cami (y a la Pelu y al Tito), que me dieron una familia y con las que he formado un hogar al cuál volver cada día, son la familia por la cuál vale la pena mejorar cada día y buscar superarme más allá de mis límites. Ellas que me soportan el día a día, en las buenas y las malas, siempre preocupadas por que esté bien y que no me falte nada. Es por ellas que busco todos los días ser la mejor versión de mi mismo, son mi razón para vivir y nunca me van a alcanzar las palabras para agradecer todo el cariño que me han dado y me dan cada día. Espero que esto sea el inicio de una nueva etapa que pueda compartir con ellas y aprovechar lo que aprendí para devolverles a ellas todo el afecto que merecen y más aún la vida y las oportunidades para que ambas puedan lograr sus sueños. Gracias chicas, este trabajo está dedicado a ustedes. También no quería dejar de agradecerles a mis suegros, Esther y Jorge, que siempre, junto a mis viejos, nos han ayudado y brindado consejos en esto nuevo para mi que es la vida con mi familia.

Agradezco a mis amigos, que a lo largo de mis años han sido muchos y muy buenos. Compañeros de historias y anécdotas inolvidables. Con los que he vivido miles de situaciones entrañables y con los que compartí risas y buenos momentos. Juntarse con ellos y pasar un buen rato hablando, jugando o simplemente matando el tiempo es algo que no tiene precio. Tanto aquellos viejos amigos de toda la vida que conozco desde pequeño como los excelentes compañeros que conocí en mis años de facultad, es un privilegio haber compartido con todos ustedes una etapa tan importante de mi vida. Gracias a todos por su buena onda, muchas gracias de verdad.

Profesores y maestros, muchos más de los que puedo nombrar, desde el jardín de infantes hasta la universidad, todos aquellos que me brindaron su sabiduría y conocimiento. Gracias a todos por su esfuerzo, por brindarme las herramientas para ser libre, puesto que sin el conocimiento la libertad es simplemente una idea vacía.

A mis compañeros del doctorado, los muchachos de la oficina 272: José, Luis, Mariano y Raúl, que me acompañaron en estos años, siempre dispuestos a dar una mano en todo lo que podían y con los que siempre pasamos un buen momento en las comilonas que hacemos. Gracias por todo muchachos, ha sido un honor compartir con ustedes estos años de doctorado.

A mis colegas de la facultad, tanto los docentes como investigadores. Especialmente a los miembros del grupo de procesamiento de lenguaje natural y el grupo de terapia con el que siempre puedo contar para obtener buen feedback sobre mi trabajo.

A los miembros del jurado de mis tesis de doctorado: Jorge, Marcelo y Paula. Gracias por su tiempo, gracias por sus ganas y gracias por haber sido parte de esto tan importante que fue mi tesis doctoral.

Por último quiero agradecer a mis dos compañeras de oficina y también de investigación, que hacen el día a día en la facultad divertido, interesante y genial. Mili, mi hermana académica, ha sido un privilegio tener a alguien tan activa, alegre y positiva, siempre brindando consejos, siempre dispuesta a debatir y otorgar su visión para ayudar a comprender estas cosas complejas que estudiamos, una compañera de trabajo excelente con la que hemos logrado sinergizar en varios proyectos tanto académicos como docentes, y una amiga extraordinaria con quien puedo contar.

Y Laura, mi madre académica, mi supervisora y mentora en este viaje que ha sido el doctorado. A vos te debo este logro más que a nadie, por apoyarme y guiarme, por nunca bajar los brazos y estar siempre dispuesta a darme toda la ayuda que necesitara, tanto en tus mejores como en tus peores momentos. Nunca me impusiste tu visión o tu agenda y siempre me animaste a seguir adelante, cualquiera fuera el tema de investigación que quisiera explorar. Más allá de eso siempre te las arreglaste para darme los contactos y las herramientas para avanzar profesionalmente y lograr todos mis objetivos. Es gracias a vos que logré formarme como lo hice. Fue mi privilegio y mi honor ser tu doctorando. Muchas gracias por todo Laura.

Dedicado a toda mi famila... Gracias por todo...

Índice general

Índice de figuras

Parte I

Preliminares

Capítulo 1

Introducción

1.1 Introducción y motivación

Esta tesis explora el uso de técnicas de aprendizaje semi-supervisado para la desambiguación de sentidos verbales del español. El objetivo es estudiar cómo añadir información de fuentes no etiquetadas, que son más grandes, puede ayudar a un clasificador que aprendió desde un recurso etiquetado pero pequeño, como es el corpus desambiguado de verbos del español SenSem [Alonso et al., 2007].

El aprendizaje automático supervisado es la tarea que infiere una función que asigna datos sin etiquetar a una etiqueta. Esta función se infiere a partir de datos etiquetados, que se han asociado manualmente a una etiqueta. Esto también se conoce como modelado de datos. Esta función se puede utilizar para encontrar nuevos mapeos dados ejemplos no vistos previamente. Sin embargo, los modelos de aprendizaje supervisado están limitados por la cantidad de datos etiquetados disponibles. Cuantos más datos etiquetados tenga un modelo, mejor será desempeño del modelo. Ahora mismo el *aprendizaje profundo* (o deep learning) requiere de grandes cantidades de datos para generar buenos modelos. Como el etiquetado de datos es una tarea que requiere trabajo humano, tener conjuntos de datos suficientemente grandes resulta costoso. Esto es también depende de la tarea. Algunas tareas son relativamente fáciles de etiquetar para cualquier persona (e.g. ver si una foto tiene un gato dentro). Para otras tareas, los datos etiquetados son más difícil de obtener, ya que requieren que los anotadores humanos sean expertos de dominio (e.g. un abogado, un médico, un lingüista, etc.).

El aprendizaje automático no supervisado es la tarea que intenta encontrar patrones o estructuras ocultas en datos no etiquetados. Debido a que los datos no están etiquetados, estos no pueden ser evaluados basándonos en exactitud o precisión. El aprendizaje no supervisado está más centrado en la exploración de los datos que en encontrar una función útil para una tarea de predicción en particular, a diferencia del aprendizaje supervisado. La ventaja del aprendizaje no supervisado es la gran cantidad de datos disponibles. Como los datos no están anotados, es barato recopilar conjuntos de datos.

El aprendizaje semisupervisado es un método intermedio. Un algoritmo semisupervisado hace uso de datos etiquetados y no etiquetados para inferir mejor una función para una tarea específica (supervisada o no). En este caso, con datos suficientes, una buena técnica de aprendizaje semisupervisado puede mejorar el algoritmo

puramente supervisado o no supervisado que esté utilizando. En particular, tiene la ventaja de tener una forma de evaluarla en términos de exactitud u otras métricas, pero también tiene la ventaja de poder expandir los modelos basándose en los conjuntos de datos no etiquetados disponibles. El objetivo de esta tesis es explorar diferentes modelos semisupervisados para mejorar una tarea supervisada que tiene las propiedades indicadas anteriormente: una falta de grandes conjuntos de datos etiquetados, pero la disponibilidad de grandes conjuntos de datos no etiquetados.

El procesamiento de lenguaje natural ayuda a los humanos a interactuar con las máquinas usando el lenguaje humano en lugar de un lenguaje formal. El principal reto del área es la ambigüedad del lenguaje humano. Pero este no es el único problema. El lenguaje humano, como muchos otros fenómenos sociales, tiene una distribución zipfiana, que es una distribución exponencial. Esta ley establece que dado que un corpus de expresiones en lenguaje natural, la frecuencia de cualquier palabra es inversamente proporcional a su rango en la tabla de frecuencias. Así, la palabra más frecuente ocurrirá aproximadamente dos veces más a menudo que la segunda palabra más frecuente, tres veces más que la tercera palabra más frecuente, etc. En el caso de tareas de clasificación para procesamiento del lenguaje natural, las clases están desequilibradas siguiendo la ley de Zipf, por lo tanto muy pocas clases ocurren muy a menudo, y el resto de tienen pocas o ninguna ocurrencia en el conjunto de datos etiquetado. Asimismo, la disponibilidad de buenos modelos para las tareas en lenguaje natural depende de los datos existentes para esa tarea. En particular, los diferentes idiomas tienen diferentes cantidades de datos, lo que permite obtener modelos mejores o peores. Por ejemplo, el inglés es un idioma con muy buenos modelos debido a la cantidad de recursos; otros idiomas son más difíciles de trabajar porque hay menos recursos disponibles.

La desambiguación de sentidos es una tarea intermedia dentro del procesamiento del lenguaje natural que, como su nombre indica, a partir de una palabra y su contexto trata de discriminar el significado de esa palabra entre un inventario de sentidos predeterminados. Por ejemplo, el sustantivo "granada" puede ser una fruta o un arma dependiendo del contexto en el que se lo utilice. Una subtarea de la desambiguación de sentidos es la *desambiguación de sentidos verbales*, que es crucial para las tareas profundas de procesamiento de un lenguaje, especialmente aquellas que podrían beneficiarse de la información sobre las relaciones entre los participantes proporcionada por el verbo, como la traducción automática, la respuesta a preguntas o la extracción de información. Por ejemplo, en la traducción automática, la desambiguación del sentido es necesaria para determinar la traducción correcta de una palabra, y también las transformaciones necesarias en la estructura sintáctica de la oración. Lo ejempplos 1.1 y 1.2 muestran cómo los diferentes sentidos pueden cambiar la traducción de una misma palabra.

Ejemplo 1.1.

- Juan **hizo** una torta de chocolate.

- *Juan* **made** *a chocolate cake.*

Ejemplo 1.2.

- Juan **hizo** la tarea por la tarde.
- *Juan* **did** *the homework in the afternoon.*

Otro ejemplo, para contestar preguntas sobre datos enlazados (*linked data*), es necesario determinar el sentido de un verbo y las relaciones lógicas entre los participantes para acceder a los nodos y bordes adecuados en una ontología. Ejemplos 1.3 y 1.4 muestran cómo el mismo verbo tiene connotaciones diferentes y, por lo tanto, establece relaciones diferentes entre los participantes que conecta.

Ejemplo 1.3.

- El investigador no pudo **acceder** a la información reservada.

Ejemplo 1.4.

- Pedro **accedió** a escribir una nota de descargo.

Finalmente, una tarea de extracción de información como la extracción de relaciones donde la desambiguación de sentidos verbales se necesita desambiguar esas relaciones, generalmente representadas como verbos en un texto no estructurado.

However, desambiguación de sentidos verbales is a highly difficult task, that achieves even lower precision rates than desambiguación de sentidos for other morphosyntactic categories (nouns, adjectives). This is arguably due to the inherent complexity of verb senses and the fact that their meaning is more pervasive than that of nouns, thus making it more difficult to discretize [Chen and Palmer, 2009]. Find below a detailed example of verb sense ambiguity for the lemma *h*ablar, with five senses in the SenSem lexicon. This can be seen in example 1.5.

Sin embargo, la desambiguación de sentidos verbales es una tarea muy difícil, que logra tasas de precisión aún más bajas que la desambiguación de sentidos para otras categorías morfosintácticas (sustantivos, adjetivos). Esto se debe probablemente a la complejidad inherente de los sentidos verbales y al hecho de que su significado está más omnipresente que el de los sustantivos, lo que hace más difícil discretizarlos [Chen and Palmer, 2009]. A continuación el ejemplo 1.5 detalla la ambigüedad de los sentidos verbales para el lema *h*ablar, con cinco sentidos en el lexicón SenSem.

Ejemplo 1.5.

- (...) una joven realizadora irlandesa que (...) **habla** castellano con acento malagueño ya que de pequeña vivió tres años en Estepona.

Ejemplo 1.6.

- El presidente del Gobierno (...) **hablará** con las principales autoridades de ambos estados sobre las perspectivas de la Unión Europea (...).

En el ejemplo 1.5 *hablar* es un estado con el significado de "ser capaz de hablar un idioma", mientras que en ejemplo 1.6 es un proceso con el significado "establecer una conversación". Los marcos de subcategorización de cada sentido también son muy diferentes: para el primero, tenemos dos temas, mientras que el segundo tiene el marco de comunicación típico, con un Origen-Agente, un Tema y un Receptor-Objetivo.

1.2 Aportes y esquema de esta tesis

La desambiguación de sentidos sigue siendo un problema abierto en el área del procesamiento del lenguaje natural. Se han investigado muchas técnicas y algoritmos diferentes para atacar este problema. Los métodos van desde el uso de recursos de lenguaje y bases de conocimiento, el aprendizaje automático supervisado basado en diccionarios y reglas, hasta técnicas puramente no supervisadas que buscan inferir los sentidos mediante la agrupación de palabras. Hay incluso métodos híbridos que son una combinación de estos. Las técnicas de aprendizaje automático supervisado se basan en corpus de palabras desambiguadas. Dado esto uno puede entrenar a un clasificador para que reconozca los sentidos en ejemplos no vistos.

La mayor parte del trabajo para la desambiguación de sentidos del español no es específicamente para verbos. Gran parte del trabajo se basa en el uso de bases de conocimiento [Agirre et al., 2014, Agirre and Soroa, 2009]. Hay algunos trabajos que utilizan técnicas de aprendizaje supervisado o técnicas de aprendizaje no supervisado [Mihalcea et al., 2004]. Pero en el momento de escribir esta tesis, el trabajo hecho específicamente para la desambiguación de sentidos verbales del español usando las técnicas semisupervisadas exploradas en este trabajo es inexistente para mi conocimiento. Además, el trabajo de esta tesis está utilizando un recurso que, salvo las publicaciones que derivaron de esta tesis, no ha sido explorado. Utilizo como recurso básico el corpus de verbos desambiguados SenSem [Alonso et al., 2007].

Este recurso, aunque valioso, ya que fue anotado por expertos de dominio, es muy pequeño y tiene una cantidad limitada de ejemplos. Pero hay una gran cantidad de corpus no etiquetado disponible en Internet para el español (por ejemplo, el corpus SBWCE [Cardellino, 2016]). Esto me da la oportunidad de estudiar técnicas semisupervisadas aplicadas a un problema que realmente podría beneficiarse de él, en lugar de un problema genérico basado en un conjunto de datos de juguetes con propiedades de las que podrían carecer los datos del mundo real. Asimismo, al estudiar el impacto de las técnicas de aprendizaje automático semisupervisado para desambiguación de sentidos verbales del español en lugar de para la desambiguación de sentidos verbales del inglés, contribuyo a la mejora del procesamiento de lenguaje natural para un idioma cuyos recursos están menos desarrollados. La principal contribución de esta tesis se puede resumir en el estudio de cómo los diferentes sistemas semi-supervisados superan los problemas de la desambiguación de sentidos verbales para el español.

Los dos capítulos siguientes presentan los conceptos fundamentales y el trabajo relacionado de esta tesis. Los capítulos 2 y 3 presentan una revisión de los antecedentes de las dos áreas principales de esta tesis: *aprendizaje automático* y *procesamiento de lenguaje natural*. Específicamente el capítulo 2 introduce los conceptos relevantes para el trabajo en los siguientes capítulos, y presenta una breve descripción y análisis del trabajo previo para las técnicas de aprendizaje semisupervisado estudiadas en esta tesis. El capítulo 3 es un capítulo de fondo sobre procesamiento de lenguaje natural y desambiguación de sentidos. Introduce los conceptos fundamentales del área y hace una revisión de los trabajos anteriores sobre técnicas de aprendizaje supervisado y semisupervisado aplicadas específicamente a las áreas de procesamiento de lenguaje natural y desambiguación de sentidos verbales. Los cinco capítulos centrales de esta tesis exploran diferentes hipótesis para el desambiguación de sentidos verbales del español y experimentan y analizan los resultados para probar las hipótesis propuestas. En el último capítulo se recapitulan los hallazgos y las lecciones aprendidas de esta tesis y se describen las líneas para el trabajo futuro.

Resumen del capítulo 4: Aprendizaje supervisado Este capítulo explora técnicas de aprendizaje supervisado de máquinas para la desambiguación de sentidos verbales. El objetivo principal es establecer una base. El capítulo explora desambiguación de sentidos verbales tanto en español como en inglés. Esto último es necesario como base de comparación para garantizar que no haya sesgos relacionados con el idioma. El capítulo explora diferentes algoritmos de clasificación y representaciones dadas por *ingeniería de features* para descartar técnicas de bajo rendimiento. También presento una discusión profunda sobre la visión que proporcionan las métricas y sus sesgos. Esto es necesario para comprender a fondo los modelos y sus puntos débiles a fin de orientar adecuadamente la evolución futura. Por último, en el capítulo se señalan las principales deficiencias del enfoque supervisado: falta de cobertura y tendencia al sobreajuste. Intento superar estos problemas con las diferentes técnicas semisupervisadas que se exploran en la tesis.

Resumen del capítulo 5: Word embeddings Este capítulo explora el uso de *word embeddings* (o vectores de palabras) como representaciones no supervisadas dentro de clasificadores supervisados. Este enfoque se conoce como *aprendizaje semisupervisado disjunto*: se utiliza una técnica no supervisada previo a una supervisada para mejorarla. El capítulo explora cómo el dominio de los word embeddings afecta al rendimiento de los clasificadores supervisados, y cómo los word embeddings ayudan a superar la falta de cobertura y la tendencia a sobreajustar, incluso a costa de perder algo de rendimiento en otros aspectos, como la precisión o la métrica F1.

Resumen del capítulo 6: Autoaprendizaje Este capítulo explora el algoritmo de autoaprendizaje, un método bien establecido y uno de los primeros algoritmos

semisupervisados existentes. El autoaprendizaje es un *algoritmo de envoltura*. El algoritmo utiliza un clasificador supervisado, entrenado en un conjunto inicial de datos etiquetados, para seleccionar instancias de un grupo de datos no etiquetados. Usando alguna medida de certeza, anota aquellas instancias para las cuales el algoritmo tiene la mayor confianza y las usa para expandir los datos etiquetados para volver a entrenar al clasificador supervisado. Los experimentos en este capítulo se centran principalmente en aumentar la cantidad de datos etiquetados disponibles para capacitar a un clasificador. El objetivo final es disponer de un modelo supervisado con una mayor cobertura integrando parte de los datos no etiquetados en el proceso de formación, asignándoles automáticamente una etiqueta. El autoaprendizaje es la primera de las técnicas de *aprendizaje semisupervisado conjunto* exploradas en la tesis. En este tipo de algoritmos los datos etiquetados y no etiquetados se utilizan juntos en el proceso de aprendizaje. Se descubre que el autoaprendizaje sufre de conjuntos de datos desequilibrados. Amplifica el sesgo hacia la clase más frecuente, por lo que el aumento esperado en la cobertura mediante la agregación de nuevos ejemplos a los datos de entrenamiento queda oscurecida debido a que los límites de decisión entre las clases son borrosos.

Resumen del capítulo 7: Aprendizaje activo Este capítulo explora el aprendizaje activo como un enfoque semisupervisado. El aprendizaje activo se basa en la misma idea de autoaprendizaje. Es un algoritmo de envoltura que utiliza un modelo supervisado para seleccionar instancias tomadas de un conjunto de datos no etiquetados. Sin embargo, en lugar de anotar las instancias automáticamente, como el autoaprendizaje, utiliza un *enfoque inteligente* para seleccionar los datos y se lo da a un humano experto de dominio para anotar. Los datos se seleccionan en función del impacto que tendrán en el modelo anotarlos correctamente (e.g. dando más información al modelo). Como la anotación manual es costosa, la idea es reducir al mínimo los recursos invertidos en la anotación. Los experimentos del capítulo muestran que la información obtenida a través del aprendizaje activo, aunque poco en comparación, puede superar el enfoque de autoaprendizaje, que generalmente está sesgado por la naturaleza del lenguaje y la ley de Zipf.

Resumen del capítulo 8: Redes neuronales en escalera Finalmente, este capítulo explora *la redes neuronales en escalera*. Este es un enfoque novedoso en el campo del aprendizaje semisupervisado, que presenta una arquitectura de red neuronal basada en la idea de entrenar simultáneamente los pesos de la red minimizando una función de costo combinado. Esta función de coste es la suma de una función de coste supervisado y una función de coste no supervisado. La red consta de dos vías: una vía de codificación que optimiza la función de coste supervisado y una vía de decodificación que reconstruye la entrada de la codificación capa por capa. La idea es que la información recopilada por el camino no supervisado ayudará al enfoque supervisado

a generalizarse mejor, superando así la tendencia a sobreajustar de los modelos supervisados. Además, se ha demostrado que el rendimiento del método es comparable o mejor que el resto de los métodos presentados en esta tesis, haciendo de las redes en escalera un candidato interesante para aprendizaje semisupervisado.

Capítulo 2

Contexto de aprendizaje automático

El núcleo de este trabajo de tesis se basa en dos áreas: **aprendizaje automático** y **procesamiento de lenguaje natural**. Más específicamente, esta tesis sigue la aplicación de técnicas de aprendizaje semisupervisado, una subárea de aprendizaje automático, para la desambiguación de sentidos verbales del español, una subárea del procesamiento de lenguaje natural. Este capítulo revisa el trabajo hecho en *aprendizaje automático*, específicamente el aprendizaje semisupervisado. El capítulo comienza con un conjunto de definiciones básicas. Continúa revisando los dos algoritmos de envoltura: autoaprendizaje y aprendizaje activo. Y termina con una breve introducción al algoritmo de la red neuronal en escalera.

2.1 Conceptos fundamentales

Tom Mitchell [Mitchell, 1997] define el **aprendizaje automático** como "el campo de la inteligencia artificial que se ocupa de la pregunta de cómo construir sistemas que mejoren automáticamente basándose en la experiencia". La experiencia, en este caso, se adquiere a través del análisis de datos con el objetivo final de generar un modelo que trabaje en tareas específicas. Ejemplos de estas tareas pueden ser sistemas para la detección de anomalías, como transacciones fraudulentas con tarjetas de crédito; sistemas de recomendación para películas, música, libros, etc.; o visión por computador.

El aprendizaje automático se divide en tres áreas principales: *Aprendizaje supervisado*, *aprendizaje no supervisado* y *aprendizaje de refuerzo*. Como este trabajo se basa principalmente en técnicas de aprendizaje supervisado con la ayuda de técnicas de aprendizaje no supervisado pasaré a describirlas.

El aprendizaje supervisado tiene el objetivo de construir un modelo predictivo a partir de datos etiquetados. Los **datos etiquetados** consisten en un conjunto de *ejemplos anotados*, donde cada ejemplo es un par de datos de entrada, generalmente un vector; y datos de salida que, dependiendo de la tarea, pueden ser un número real, para regresión, o un valor categórico, para clasificación. Por ejemplo, el filtrado de spam cuando un correo electrónico está clasificado como spam o no. En términos más formales, un algoritmo de aprendizaje supervisado es un mapeo desde la entrada

$\mathbf{x}$ a la salida $\mathbf{y}$ con el objetivo t, entrenado a partir de un conjunto de N pares: $\{(\mathbf{x}(n), t(n))|1 \leq n \leq N\}$.

El **aprendizaje no supervisado** tiene el objetivo de describir o descubrir una estructura oculta a partir de datos no etiquetados. Los **datos no etiquetados** consisten en un conjunto de *ejemplos no anotados*: datos sin ninguna información sobre la pertenencia a alguna clase o que tengan un valor asociado. Por ejemplo, agrupación de artículos de noticias. En términos más formales, un algoritmo de aprendizaje no supervisado es la inferencia de una función que describe algún patrón o estructura a partir de un conjunto de puntos de datos no etiquetados: $\{\mathbf{x}(m)|1 \leq m \leq M\}$.

Hay un conjunto de técnicas que están entre el aprendizaje automático supervisado y no supervisado: **aprendizaje automático semisupervisado**. Estas son las técnicas que más exploro en esta tesis. Zhu y Goldberg [Zhu and Goldberg, 2009] lo describen como "la tarea que consiste en usar ejemplos anotados y no anotados para aprender mejor una tarea supervisada o no supervisada". Para el objetivo de esta tesis, me referiré a las técnicas semisupervisadas que exploran el uso de datos no etiquetados para ayudar a un modelo supervisado. En términos formales, dado un modelo supervisado de un conjunto de pares de entrenamiento $\{\mathbf{x}(n), t(n)|1 \leq n \leq N\}$; el aprendizaje semi-supervisado, en el ámbito de esta tesis, estudia cómo los datos auxiliares no etiquetados $\{\mathbf{x}(n)|N+1 \leq n \leq M\}$ pueden ayudar en la formación del modelo supervisado. A menudo, los datos etiquetados son escasos, mientras que los no etiquetados son abundantes, es decir $N \ll M$. La idea es que los datos no etiquetados pueden ayudar a expandir el modelo supervisado ya que tienen más información que un conjunto de datos etiquetados con respecto a las instancias (ya que tiene mayor cantidad de instancias). Esa información subyacente ayuda a expandir la información presente en el modelo. La información puede sintetizarse como nuevas características para una clase, variables latentes provenientes de las representaciones, asociaciones más fuertes entre características y clases, etc.

En este trabajo exploro dos tipos de aprendizaje semisupervisado: *aprendizaje semisupervisado disjunto* y *aprendizaje semisupervisado conjunto*. En el **aprendizaje semisupervisado disjunto** los algoritmos de aprendizaje supervisado y no supervisado son entrenados por separado, y luego la información de uno es dada al otro para mejorarlo [Weston et al., 2008]. Por ejemplo, el uso de *word embeddings* entrenados en un corpus sin etiquetar para una tarea supervisada. En contraste, el **aprendizaje conjunto** tanto los datos supervisados como los no supervisados afectan al procedimiento de entrenamiento del modelo simultáneamente. Puede haber otra subdivisión en los métodos de aprendizaje conjunto. El aprendizaje conjunto basado en *algoritmos de envoltura* parte de un clasificador supervisado y entrenado en datos etiquetados, y usa la información de ese modelo para expandirlo con retroalimentación de datos no etiquetados (por ejemplo, un enfoque de aprendizaje activo que hace una selección "inteligente" de candidatos reunidos a partir de un corpus no supervisado para anotar manualmente y posteriormente utilizar para alimentar un modelo supervisado). Otros algoritmos de aprendizaje conjunto utilizan los conjuntos de datos etiquetados y no

etiquetados en el procedimiento de formación del algoritmo (por ejemplo, las *redes en escaleras* aprenden mediante la minimización de una función de objetivo conjunta, compuesta por un objetivo supervisado y otro no supervisado).

Una categorización que se puede hacer también en métodos semi-supervisados, pero más generalmente en métodos supervisados también lo es: *aprendizaje superficial* y *aprendizaje profundo*. El *aprendizaje superficial* puede ser descrito como un proceso que trata de aprender por aproximación. Es más bien una memorización que una comprensión de los conceptos o estructuras subyacentes de los datos. El **aprendizaje profundo** (deep learning) es un enfoque de aprendizaje automático cuyo objetivo es permitir que las computadoras aprendan de la experiencia y entiendan el mundo en términos de una jerarquía de conceptos, con cada concepto definido en términos de su relación con conceptos más simples. Al recopilar conocimientos a partir de la experiencia, este enfoque evita la necesidad de que los operadores humanos especifiquen formalmente todo el conocimiento que necesita el ordenador. La jerarquía de conceptos permite que la computadora aprenda conceptos complicados construyéndolos a partir de otros más simples [Goodfellow et al., 2016]. El modelo de aprendizaje profundo más básico es un **perceptrón multicapa** (multilayer perceptron), un modelo de red neuronal artificial hacia adelante, que mapea un conjunto de valores de entrada a un conjunto de valores de salida.

2.2 Trabajo previo

2.2.1 Aprendizaje semisupervisado disjunto

El **aprendizaje semisupervisado disjunto** fue definido por Weston et al. [Weston et al., 2008], como el uso de word embeddings para ayudar en tareas de aprendizaje profundo en procesamiento de lenguaje natural. El concepto puede entenderse como el uso de un recurso o de una tarea auxiliar que se obtiene a partir de datos no etiquetados (por ejemplo, word embeddings) y cómo ayuda en una tarea supervisada entrenada en datos etiquetados que están desarticulados de los datos no etiquetados de la tarea auxiliar. Tal como se definió, no hay otros usos del término salvo dentro del área del procesamiento de lenguaje natural. Por lo tanto, lo discutiré más adelante en el capítulo 3.

2.2.2 Autoaprendizaje

El *autoaprendizaje* (también conocido como *autoentrenamiento*, *autoenseñanza* o *bootstrapping*) es una técnica comúnmente usada para el aprendizaje semisupervisado. Es probablemente la primera idea sobre el uso de datos no etiquetados en la clasificación [Chapelle et al., 2010]. Este es un *algoritmo envolvente* que usa repetidamente un método de aprendizaje supervisado. El clasificador inicial se entrena primero con un conjunto de datos etiquetado como semilla (generalmente pequeño) y luego se utiliza para clasificar datos de un conjunto de datos sin etiquetar. En general, los puntos de

datos en los que el clasificador confía más, junto con las etiquetas predichas, se añaden al conjunto de entrenamiento. Se vuelve a entrenar al clasificador y se repite el procedimiento. Nótese que el clasificador usa sus propias predicciones para enseñarse a sí mismo [Zhu, 2005], por lo que en este esquema es posible que el error de clasificación se refuerce a sí mismo. Existen diferentes métodos para tratar de controlar este fenómeno. Scudder [Scudder, 1965] presenta una máquina de reconocimiento de patrones adaptativa no entrenada hecha de una simple máquina de reconocimiento de patrones entrenada para detectar un patrón desconocido, fijo, que ocurre aleatoriamente (derivado usando el enfoque de Bayes) y su probabilidad de error es analizada usando su propia salida en lugar de un profesor. En visión artificial, Rosenberg et al. [Rosenberg et al., 2005] aplican el autoentrenamiento a sistemas de detección de objetos a partir de imágenes, y muestran que la técnica semi-supervisada se compara favorablemente con un detector de última generación. Culp y Michailidis [Culp and Michailidis, 2008] ofrecen un algoritmo de autoaprendizaje iterativo que extiende al aprendedor de un entorno supervisado a uno semisupervisado. Analizan las propiedades de convergencia del algoritmo, ya que el autoaprendizaje es un algoritmo difícil de analizar en general.

2.2.3 Aprendizaje activo

Aprendizaje activo es otro ejemplo de una tarea de aprendizaje conjunto. También es un algoritmo envolvente que se entrena a partir de un conjunto de datos etiquetado como semilla inicial, pero que utiliza otro enfoque para incorporar instancias del conjunto de datos no etiquetado. La hipótesis clave es que si se permite que el algoritmo de aprendizaje elija los datos de los que aprende, funcionará mejor con menos entrenamiento [Settles, 2009]. El algoritmo inspecciona un conjunto de ejemplos sin etiquetar, y los clasifica según cuánto podrían mejorar el rendimiento del algoritmo si estuvieran etiquetados. Luego, un anotador humano (llamado *oráculo* o *experto de dominio*) etiqueta los ejemplos de más alto rango, los cuales se añaden al conjunto de ejemplos de entrenamiento a partir de los cuales el algoritmo infiere su modelo de clasificación, y el bucle comienza de nuevo. En algunos enfoques de aprendizaje activo, el oráculo puede anotar las características que describen las instancias, y no (sólo) las instancias mismas. Este último enfoque proporciona un aprendizaje aún más rápido en algunos casos [Druck et al., 2009].

Se han aplicado diferentes estrategias para determinar las instancias más útiles a ser anotadas por el oráculo, incluyendo el cambio de modelo esperado, la reducción de errores esperada o los métodos ponderados por densidad [Settles, 2009]. La estrategia más intuitiva y popular es el **muestreo por incertidumbre** [Lewis and Catlett, 1994], que elige aquellas instancias o características en las que el algoritmo es más incierto, de entre un gran conjunto de datos etiquetados automáticamente. Como en *autoaprendizaje*, este es también un método envolvente, siendo el algoritmo supervisado para hacer el paso de clasificación elegido abiertamente. Esta estrategia se ha aplicado con éxito a las tareas de extracción de información [Culotta and McCallum, 2005, Settles and Craven, 2008]. La incertidumbre puede ser calculada por diferentes

métodos dependiendo del algoritmo de aprendizaje. Especialmente populares son los métodos que explotan los márgenes de las support vector machines (SVM), como en [Tong and Koller, 2002]. Los métodos más sencillos aprovechan directamente la certeza que el clasificador proporciona para cada instancia que se clasifica automáticamente.

2.2.4 Redes neuronales en escalera

La **red neuronal en escalera** es la última técnica de aprendizaje semisupervisado que revisaré en este trabajo. También es una técnica de aprendizaje conjunto. Pero a diferencia del autoaprendizaje o del aprendizaje activo, no se trata de un algoritmo envolvente, sino de una arquitectura especial para una red neuronal artificial. El enfoque semisupervisado estudiado por la red de escaleras es reciente, siendo introducido en el trabajo de Rasmus et al. [Rasmus et al., 2015a]. Discutiré más a fondo este trabajo y los detalles del algoritmo de la red de escaleras en el capítulo 8, ya que la explicación del mismo es núcleo del trabajo de esta tesis. La obra de Rasmus amplía la obra original de Valpola [Valpola, 2014] que introduce el concepto de redes de escaleras para el aprendizaje no supervisado. El trabajo de Rasmus que estudio en esta tesis utiliza la base de otro trabajo de Rasmus et al. [Rasmus et al., 2015b] que añade conexiones laterales a los *autoencoders* para ayudar al aprendizaje supervisado (e.g. clasificación) con una tarea auxiliar de aprendizaje no supervisada (e.g. reconstrucción de un dato de entrada). La combinación de trabajos derivó en el diseño de una red de escaleras como algoritmo semi-supervisado.

Capítulo 3

Contexto de procesamiento de lenguaje natural

Además de aprendizaje automático, otra área principal de estudio en esta tesis es **procesamiento de lenguaje natural**. En particular, me concentro en una subárea del procesamiento de lenguaje natural: **desambiguación de sentidos** con especial atención en la **desambiguación de sentidos verbalesdel español**. En este capítulo se revisan dichas áreas. Más específicamente, el capítulo explora algunos de los trabajos relevantes sobre procesamiento de lenguaje natural y desambiguación de sentidos utilizando las técnicas de aprendizaje semisupervisado descritas en el capítulo anterior. El capítulo empieza describiendo los conceptos fundamentales del área de procesamiento de lenguaje natural y desambiguación de sentidos. Luego explica los detalles de los word embeddings y sus aplicaciones en procesamiento de lenguaje natural. Finalmente continúa con la revisión de algoritmos de autoaprendizaje y aprendizaje activo en el área de procesamiento de lenguaje natural. No hay trabajo previo para redes de escaleras específicamente para tareas procesamiento de lenguaje natural ya que la técnica es muy reciente, por lo que no lo discutiré aquí.

3.1 Conceptos fundamentales

El procesamiento de lenguaje natural es el campo de la inteligencia artificial que trabaja en la interacción hombre-máquina a través de los lenguajes naturales (i.e. los idiomas, e.g. inglés, español, chino). En esta área, el aprendizaje automático ha ido ganando popularidad desde la "revolución estadística" para hacer frente a tareas que solían realizarse con métodos más ad-hoc (por ejemplo, basados en complejos conjuntos de reglas escritas a mano) [Johnson, 2009]. Este tipo de aprendizaje automático aplicado a procesamiento de lenguaje natural es también conocido como *procesamiento de lenguaje natural estadístico*.

La **desambiguación de sentidos** intenta desambiguar automáticamente las palabras. Ide y Véronis [Ide and Véronis, 1998] establecieron que esta tarea ha sido un objetivo central desde la concepción del procesamiento del lenguaje natural como una tarea de inteligencia artificial. La desambiguación del sentido de la palabra ha sido categorizada como una *tarea intermedia* por Wilks y Stevenson [Wilks and Stevenson, 1996], es decir, no es una tarea final en sí misma, sino más bien un paso necesario

a realizar para proceder con la mayoría de las tareas de procesamiento del lenguaje natural. Es esencial para las aplicaciones de comprensión del lenguaje (por ejemplo, la comunicación hombre-máquina) y útil (y a veces necesaria) para las aplicaciones que no tienen como objetivo la comprensión del lenguaje (por ejemplo, la traducción automática, la recuperación de información, la extracción de información, etc.).

Este trabajo se centra principalmente en una subtarea de desambiguación de sentidos, la **vsd**. En particular, este trabajo se centra en la aplicación de la desambiguación de sentidos verbales para el español. Tanto la desambiguación del sentido de la palabra como del verbo pueden verse como problemas de aprendizaje automático supervisado. Se han estudiado diferentes enfoques en estas áreas.

3.2 Trabajo relacionado

3.2.1 Aprendizaje automático para desambiguación de sentidos

Para desambiguación de sentidos el trabajo de referencia en el momento de escribir esta tesis es *It Makes Sense* [Zhong and Ng, 2010]. Es un sistema para desambiguación de sentidos de palabras en inglés. El sistema puede ser ajustado a las necesidades del usuario, pero originalmente está empaquetado con un clasificador vectorial de soporte lineal con múltiples características basadas en el conocimiento.

McCarthy y Carroll [McCarthy and Carroll, 2003] trabajaron en la desambiguación de sustantivos, verbos y adjetivos utilizando preferencias selectivas adquiridas a partir de texto automáticamente preprocesado y analizado. Las preferencias selectivas se adquieren para relaciones gramaticales (sujeto, objetos directos y sustantivo-adjetivo) que involucran sustantivos y adjetivos o verbos relacionados gramaticalmente. Utilizan synsets de WordNet para definir el inventario de sentidos. Su método explota los enlaces hipónimos que se dan para los sustantivos (por ejemplo, *queso* es un hipónimo de *comida*), los enlaces troponímicos para los verbos (por ejemplo, *murmurar* es un tropónimo de *caminar*), y la relación "similar a" dada para los adjetivos (por ejemplo, una sentido de *barato* es similar a *endeble*). A partir del documento, no está claro si las preferencias selectivas tienen un impacto positivo en desambiguación de sentidos verbales.

Ye y Baldwin [Ye and Baldwin, 2006] usan preferencias selectivas extraídas con un etiquetador de roles semánticos para desambiguación de sentidos verbales. Su marco de desambiguación de sentidos verbales se basa en tres componentes: la extracción de las características de desambiguación, la selección de las mejores características de desambiguación con respecto a datos nunca antes vistos y la sintonización de los parámetros del aprendedor. Para su estudio utilizan un algoritmo de Máxima Entropía [Berger et al., 1996]. Las características de desambiguación de sentidos verbales que usaron incluyen preferencias selectivas y características sintácticas, por ejemplo, bolsa de palabras, bolsa de etiquetas PoS; características basadas en árboles analizados usando diferentes niveles del árbol como fuente de información; y características sintácticas basadas en árboles no analizados, por ejemplo, voz del verbo. Muestran un

mejor rendimiento de su sistema cuando se tienen en cuenta las preferencias selecctivas.

Otro trabajo sobre desambiguación de sentidos verbales del inglés es el de Chen y Palmer [Chen and Palmer, 2009], que presenta un sistema amplia cobertura de alto rendimiento de desambiguación de sentidos verbales para el inglés que utiliza características motivadas lingüísticamente y un modelo de aprendizaje de máxima entropía suavizado. Kawahara y Palmer [Kawahara and Palmer, 2014] presentaron un método supervisado para desambiguación de sentidos verbales basado en VerbNet. Contrariamente a los métodos desambiguación de sentidos verbales más comunes, que crean un clasificador para cada verbo que alcanza un umbral de frecuencia, crearon un único clasificador para ser aplicado a verbos raros o no vistos en un texto nuevo. Su clasificador también explota las características semánticas generalizadas de un verbo y sus modificadores para tratar mejor los verbos raros o no vistos.

El trabajo de Sudarikov et al. [Sudarikov et al., 2016] muestra una aplicación directa de desambiguación de sentidos verbales en otro campo de estudio. Presentan experimentación en traducción automática usando información de desambiguación de sentidos verbales. Evalúan varias opciones para utilizar los sentidos verbales en el idioma de origen como un factor adicional para el sistema de traducción automática estadística de Moses. Sus resultados muestran una mejora de la calidad de la traducción estadísticamente significativa.

Muchas de las características que se usan para desambiguación de sentidos verbales del inglés no están disponibles para desambiguación de sentidos verbales del español porque las herramientas de preprocesamiento y los corpus anotados están menos desarrollados.

Para desambiguación de sentidos en el español, Màrquez et al. [Màrquez et al., 2007] se enfoca principalmente en desambiguación de sentidos para sustantivos. Utilizaron un enfoque de tres vías: si la palabra tiene más de un número umbral de ocurrencias, se clasifica con un clasificador SVM; si la palabra tiene menos ocurrencias que el umbral, se le asigna el sentido más frecuente (MFS) en el corpus de capacitación; si la palabra no se presenta en el corpus de capacitación, se le asigna el MFS en WordNet. Las características del clasificador SVM eran una bolsa de palabras, n-gramas de etiquetas pos y lemas parciales del habla, y una etiqueta sintáctica y función sintáctica del constituyente que tiene el sustantivo objetivo como cabeza.

Anther work in WSD with applications in Spanish is the work of Montoyo et al. [Montoyo et al., 2011] where the task of WSD consists in assigning the correct sense to words using an electronic dictionary as the source of word definitions. They present a knowledge-based method and a corpus-based method. In the knowledge-based method the underlying hypothesis is that the higher the similarity between two words, the larger the amount of information shared by two of their concepts. The corpus-based method is based on conditional maximum-entropy models, it was implemented using a supervised learning method that consists of building word-sense classifiers using a semantically annotated corpus. Among the features for the classifier they used word

forms, words in a window, part-of-speech tags and grammatical dependencies.

Otro trabajo en desambiguación de sentidos con aplicaciones para el español es el trabajo de Montoyo et al. [Montoyo et al., 2011] donde la tarea de desambiguación de sentidos consiste en asignar el sentido correcto a las palabras utilizando un diccionario electrónico como fuente de definiciones de palabras. Presentan un método basado en el conocimiento y un método basado en el corpus. En el método basado en el conocimiento, la hipótesis subyacente es que cuanto mayor es la similitud entre dos palabras, mayor es la cantidad de información compartida por dos de sus conceptos. El método basado en corpus se basa en modelos de máxima entropía condicional, se implementó utilizando un método de aprendizaje supervisado que consiste en construir clasificadores de sentido de palabra utilizando un corpus semánticamente anotado. Entre las características del clasificador se utilizaron formas de palabras, palabras en una ventana, etiquetas de parte de voz y dependencias gramaticales.

Es notable que las características para desambiguación de sentidos del español son más superficiales que las disponibles para el inglés. En esta tesis exploraremos más combinaciones de características dirigidas específicamente al desambiguación de sentidos verbales del español.

3.2.2 Word embeddings (Vectores de Palabras)

Word embeddings (vectores de palabras) son representaciones distribuídas de palabras en forma de vectores densos o números reales. El concepto detrás de este método es la integración matemática de un vector ralo de alta dimensión con una dimensión por palabra (también conocido como *one-hot encoding*) a un espacio vectorial continuo de una dimensión mucho menor. Existen diferentes métodos para obtener word embeddings, pero todos ellos tienen el objetivo de generar representaciones de palabras a partir de un corpus no etiquetado.

The use of word embeddings for supervised natural language processing tasks is an example of a disjoint learning task, and due to the nature of this work, one of my main areas of study. The idea behind word embeddings representations is to find a compact vectorial representation. Ideally, in this representation, each dimension captures underlying, latent properties of the word (syntactic or semantic). In this way, the embedding is superior to a representation of the words that stays at a shallow level, capturing word co-occurrences only.

El uso de word embeddings para tareas de procesamiento de lenguaje natural supervisado es un ejemplo de tarea de aprendizaje disjunto, y debido a la naturaleza de este trabajo, una de mis principales áreas de estudio. La idea detrás de las representaciones mediante word embeddings es encontrar una representación vectorial compacta. Idealmente, en esta representación, cada dimensión captura las propiedades subyacentes y latentes de la palabra (sintácticas o semánticas). De esta manera, los embeddings son superiores a una representación de las palabras que permanece en un nivel poco profundo, capturando sólo las co-ocurrencias de palabras.

Collobert y Weston [Collobert and Weston, 2008] diseñaron una arquitectura de red neuronal convolucional para el aprendizaje multitarea que, basada en un modelo de lenguaje, proporciona muchas predicciones diferentes sobre el procesamiento del lenguaje para una frase dada: etiquetas pos, etiquetas de entidad con nombre, roles semánticos, palabras semánticamente similares y la probabilidad de que una frase tenga sentido (tanto desde el punto de vista sintáctico como semántico). Cada tarea es entrenada usando corpus etiquetados excepto el modelo de lenguaje que se obtiene de una manera completamente no supervisada. Este modelo de lenguaje está proyectad a un espacio más denso y continuo, lo que mejora la representación de las palabras para todas las tareas posteriores. Con este enfoque, conocido como de principio a fin, alcanzan un rendimiento de vanguardia en cada tarea.

Turian et al. [Turian et al., 2010] hacen una introducción general a algunas de las representaciones de palabras no supervisadas más comunes: representaciones distribucionales (p. ej., análisis semántico latente), representaciones basadas en clústers (p. ej., Brown clusters) y representaciones distribuidas (p. ej., word embeddings). Presentan word embeddings generalmente derivadas de *modelos neuronales de lenguaje*, que es un modelo de lenguaje basado en redes neuronales, explotando la capacidad de aprender representaciones distribuidas para reducir el impacto del problema de la dimensionalidad [Bengio, 2008]. El problema con estos modelos, históricamente, era el lento entrenamiento con escalamiento basado en el vocabulario para el cálculo de cada modelo [Bengio et al., 2003]. Esto se ha abordado posteriormente y se ha reducido la dependencia lineal del tamaño del vocabulario [Morin and Bengio, 2005, Collobert and Weston, 2008, Mnih and Hinton, 2009]. En su trabajo mejoran la precisión de los diferentes sistemas de procesamiento de lenguaje natural existentes mediante el uso de representaciones de palabras sin supervisión como características adicionales. Evalúan tres diferentes representaciones de palabras sin supervisión: Brown clústers [Brown et al., 1992], representaciones de Collobert y Weston [Weston et al., 2008], y representaciones HLBL de palabras [Mnih and Hinton, 2009]; y evaluarlas en el reconocimiento de entidades nombradas y en el chunking. Utilizando estas representaciones, muestran efectivamente una mejora del desempeño en líneas de base casi de última generación.

3.2.2.1 Word2Vec

El trabajo de Mikolov et al. [Mikolov et al., 2013a] presentó dos arquitecturas para calcular representaciones vectoriales continuas de palabras a partir de conjuntos de datos muy grandes: el *modelo de skip-grams continuos* y *modelo de bolsas de palabras continuos*.

Estos modelos utilizan una arquitectura para el aprendizaje de representaciones distribuidas de palabras que intenta minimizar la complejidad computacional. Exploran otros métodos para crear un modelo de lenguaje usando redes neuronales como el trabajo de Bengio [Bengio et al., 2003], y concluyen que en tales modelos la mayor parte de la complejidad es causada por la capa oculta no lineal en el modelo. Por lo

tanto, exploran modelos más sencillos que podrían no ser capaces de representar los datos con la misma precisión que las redes neuronales, pero que posiblemente puedan ser entrenados en muchos más datos de manera más eficiente.

El modelo de bolsa de palabras continua (CBOW) consiste en una red neuronal con una capa oculta lineal donde todas las palabras de entrada se proyectan en la misma posición y su vector es medio. La idea es representar la probabilidad de que una palabra ocurra dado un contexto de palabras como entrada.

El modelo de skipgrams continuos es similar al modelo CBOW, pero en lugar de predecir la palabra actual basada en el contexto, trata de maximizar la predicción de una palabra basada en otra palabra en el mismo contexto. Más precisamente, utilizan cada palabra actual como entrada a un clasificador log-lineal con capa de proyección continua, y predicen palabras dentro de un cierto rango antes y después de la palabra actual. Descubrieron que aumentar el rango mejora la calidad de los vectores de palabras resultantes, pero también aumenta la complejidad computacional. Dado que las palabras más distantes suelen estar menos relacionadas con la palabra actual que las más cercanas a ella, dan menos peso a las palabras distantes al muestrear menos de esas palabras en sus ejemplos de capacitación.

En resumen, ambos modelos se basan en una red neuronal con una capa oculta lineal que da ciertas palabras (un promedio del contexto en el caso de CBOW y una sola palabra en el caso de skipgram) que predice las palabras que están cerca en el texto. La red es entonces entrenada en un corpus grande y las proyecciones son los word embeddings que el modelo de Word2Vec produce.

3.2.2.2 El modelo de skipgrams con muestreo negativo

El modelo de skipgrams es la arquitectura con la que trabajé en esta tesis. Es capaz de aprender representaciones vectoriales distribuidas de alta calidad que capturan relaciones sintácticas y semánticas de palabras latentes. Como dije antes, es una red neuronal cuyo objetivo de entrenamiento es encontrar representaciones de palabras que sean útiles para predecir las palabras que la rodean en una oración o en un documento. Más formalmente, dada una secuencia de palabras de entrenamiento $w_1, w_2, w_3, \ldots, w_T$, el objetivo del modelo Skip-gram es maximizar la probabilidad media de registro.

$$\frac{1}{T}\sum_{t=1}^{T}\sum_{-c\leq j\leq c, j\neq 0}\log p(w_{t+j}|w_t)$$

donde c es la ventana de contexto. En la formulación básica de skipgrams el valor de $p(w_{t+j}|w_t)$ está definido como una función de softmax:

$$p(w_O|w_I) = \frac{e^{{v'_{w_O}}^{\top} v_{w_I}}}{\sum_{w=1}^{W} e^{{v'_w}^{\top} v_{w_I}}}$$

donde v_w y v'_w son las proyecciones vectoriales (de "entrada" y "salida" respectivamente) de w, y W es el número de palabras en el vocabulario. Esta representación se escala con el tamaño del vocabulario, que puede ser de millones o miles de millones.

En otro trabajo de Mikolov et al. [Mikolov et al., 2013b], presentan extensiones a la arquitectura original del modelo de programa de salto que mejoran tanto la calidad de los vectores como la velocidad de entrenamiento. En primer lugar, consiguen una aceleración significativa del proceso mediante el submuestreo de las palabras frecuentes. Esto también ayuda al aprender representaciones de palabras más regulares. Sin embargo, el valor añadido real de este documento es la presentación del modelo de muestreo negativo. Para ello, exploran la técnica de Estimación Contrastante de Ruido (NCE) [Gutmann and Hyvärinen, 2012]. Mientras que se puede demostrar que NCE maximiza aproximadamente la probabilidad de registro del softmax, el modelo Skipgram sólo se ocupa de aprender representaciones vectoriales de alta calidad, por lo que son libres de simplificar NCE siempre y cuando las representaciones vectoriales conserven su calidad. Definen el muestreo negativo (NEG) por el objetivo:

$$\log \sigma({v'_{w_O}}^\top v_{w_I}) + \sum_{i=1}^{k} \mathbb{E}_{w_i \sim P_n(w)} \Big[\log \sigma(-{v'_{w_i}}^\top v_{w_I}) \Big]$$

el cual se usa para reemplazar cada término $\log p(w_O|w_I)$ en el objetivo del modelo Skip-gram. En este caso, el $P_n(w)$ es una distribución de ruido a partir de la cual el algoritmo extrae ejemplos aleatorios de palabras que probablemente no sucederán en el contexto de la palabra que están tratando de modelar. La distribución es un hiperparámetro del modelo. En el documento original, está modelado a partir de una distribución unigrama.

3.2.2.3 Word embeddings para desambiguación de sentidos

El trabajo de Taghipour y Ng [Taghipour and Ng, 2015] investiga dos maneras de incorporar los word embeddingsun entorno de desambiguación de sentido de palabra y evalúa algunas tareas de muestra léxica y de todas las palabras de SensEval/SemEval y también una tarea de muestra léxica específica del dominio. Los resultados muestran que tales representaciones mejoran consistentemente la precisión del sistema de desambiguación de sentidos supervisado seleccionado.

Rothe y Schütze [Rothe and Schütze, 2015] presentaron un sistema para aprender embeddings articulares para synsets y lexemas. Las embeddings de synset/lexeme viven en el mismo espacio vectorial que las embeddings de palabras que no están en WordNet y por lo tanto no tienen ningún synset asociado. El sistema logra un rendimiento de última generación en tareas de desambiguación de similitud de palabras y de sentido común.

Un trabajo muy completo sobre word embeddings como características para desambiguación de sentidos es el estudio de evaluación de Iacobacci et al. [Iacobacci et al., 2016]. Proponen diferentes métodos a través de los cuales se pueden aprovechar los

word embeddings en una arquitectura de sistema WSD supervisada de última generación, y realizan análisis de cómo los diferentes parámetros afectan al rendimiento. Muestran cómo un sistema de desambiguación de sentidos que hace uso de word embeddings por sí solo, si se diseña correctamente, puede proporcionar una mejora significativa en el rendimiento en comparación con un sistema para desambiguación de sentidos de última generación que incorpora varias características estándar de desambiguación de sentidos. Exploran diferentes disposiciones de los embeddings para representar las frases, como concatenación, sumas medias y ponderadas en función de la distancia. También comparan diferentes métodos para obtener embeddings de palabras.

Para desambiguación de sentidos del español, en el momento de escribir esta tesis, el único trabajo que pude encontrar fue el mío (sobre el que se extiende esta tesis) en Cardellino y Alonso [Cardellino and Alonso i Alemany, 2017].

En un área relacionada, el trabajo de Gella et al. [Gella et al., 2016] explora un tipo diferente de embeddings para ayudar en desambiguación de sentidos verbales. Introducen una tarea novedosa definida como *desambiguación de sentidos visual*: dada una imagen y un verbo, asignan el verbo correcto del sentido, es decir, el que describe la acción representada en la imagen. El área es el aprendizaje multimodal, se trata de un área que combina diferentes áreas de aprendizaje automático en una tarea que las comprende, en este caso procesamiento de lenguaje naturalcon la visión artificial. El trabajo de Gella et al. introduce una extensión en los conjuntos de datos multimodales para añadir etiquetas de sentido. Proponen un algoritmo no supervisado que realiza la desambiguación del sentido visual utilizando embeddings textuales, visuales o multimodales.

3.2.3 Autoaprendizaje

El autoaprendizaje ha sido utilizado en muchas tareas de procesamiento de lenguaje natural. Por ejemplo, Riloff et al. [Riloff et al., 2003] utilizan el autoaprendizaje para identificar sustantivos subjetivos. Maeireizo et al. [Maeireizo et al., 2004] clasifican los diálogos como "emocionales" o "no emocionales" con un procedimiento que involucra dos clasificadores. Weld et al. [Weld et al., 2009] presentan el uso de un algoritmo de autoaprendizaje como caso de estudio de extracción de información abierta. Utilizan el algoritmo de Wikipedia y muestran cómo el proceso de su algoritmo utiliza un método de bootstrapping para permitir la extracción de información de un conjunto más amplio de páginas web generales.

En el área de la desambiguación del sentido de la palabra, el trabajo de referencia sobre el autoaprendizaje es la publicación Yarowsky de 1995 [Yarowsky, 1995]. En su trabajo, Yarowsky construye un modelo de desambiguación basado en las palabras co-ocurrentes con ejemplos etiquetados manualmente. Luego, este modelo se aplica a ejemplos no etiquetados. Los ejemplos a los que el modelo puede asignar un sentido se incorporan como ejemplos de capacitación y se capacita a un nuevo modelo. Este proceso se aplica iterativamente hasta que se alcanza una condición de terminación, es

decir, no se puede asignar un sentido a ningún ejemplo nuevo o la confiabilidad de la evidencia encontrada por el modelo es demasiado baja. Después de cada iteración, el modelo resultante tiene una cobertura posiblemente mayor que las versiones anteriores. Por lo tanto, este método es útil para construir una herramienta de la vida real a partir de un número limitado de ejemplos.

Mihalcea [Mihalcea, 2004] investiga la aplicación del co-entrenamiento y el autoentrenamiento a desambiguación de sentidos. En particular, investiga en la selección de hiperparámetros con varios grados de reducción de errores. En su trabajo selecciona a los mejores candidatos devueltos por el algoritmo supervisado para añadirlos a los ejemplos de entrenamiento (en lugar de usar algún umbral y añadirlo todo como hago en mi trabajo). Finalmente, presenta un método que combina la co-formación con el voto por mayoría para suavizar las curvas de aprendizaje de bootstrapping y mejorar el rendimiento medio.

En un área relacionada, Yuan et al. [Yuan et al., 2016] exploran un *clasificador de propagación de etiquetas* para desambiguación de sentidos semi-supervisada con modelos neuronales. Estudian desambiguación de sentidos con una red de memoria recurrente a largo y corto plazo. Buscan capturar mejor los patrones secuenciales y sintácticos del texto. La tarea semi-supervisada es un extra que tienen para aliviar la falta de datos de entrenamiento en desambiguación de sentidos. Demuestran resultados de última generación, especialmente en verbos para inglés. El clasificador de propagación de etiquetas expande las etiquetas de los datos no supervisados basándose en la similitud que tienen los datos no etiquetados con algunos de sus vecinos etiquetados. Un enfoque muy similar es utilizar un algoritmo de autoaprendizaje como envoltorio para un algoritmo vecino más cercano.

3.2.4 Aprendizaje activo

Las técnicas de aprendizaje activo no se utilizan tan ampliamente en el procesamiento del lenguaje natural como cabría esperar de la buena aceptación de las técnicas de aprendizaje automático en general, y de la necesidad de desarrollar datos etiquetados. Además de la dificultad técnica que plantean muchos sistemas de aprendizaje activo, y el riesgo de que un método de aprendizaje activo sea peor que el aprendizaje pasivo, existe también el problema del desequilibrio de clases a tener en cuenta, frecuente en el procesamiento del lenguaje natural.

El desequilibrio de clase es uno de los problemas más difíciles en el aprendizaje activo, y uno en el que el muestreo de incertidumbre puede muy probablemente ser peor que el muestreo aleatorio. Se han propuesto muchos enfoques para tratar el problema del desequilibrio de clase dentro del aprendizaje activo [Tomanek and Hahn, 2009, Bloodgood and Vijay-Shanker, 2009, Zhu and Hovy, 2007, Ertekin et al., 2007, He and Carbonell, 2007], pero no se ha encontrado una solución definitiva para el problema.

En la línea de las técnicas semi-supervisadas para la desambiguación del sentido de la palabra, el aprendizaje activo se ha aplicado con éxito a la desambiguación del

sentido del verbo en Dligach y Palmer [Dligach and Palmer, 2011], donde exploran los beneficios de usar un modelo de lenguaje no supervisado para seleccionar ejemplos semilla como un corpus inicial para un enfoque de aprendizaje activo iterativo. Implica la formación de un modelo lingüístico a partir de un corpus de ejemplos de candidatos no etiquetados y la selección de los ejemplos con baja probabilidad de modelo lingüístico. Este punto de partida inteligente parece proporcionar un mejor rendimiento para los sentidos verbales, donde hay una distribución sesgada de clases, porque es capaz de seleccionar ejemplos representativos de clases minoritarias. En mi línea de trabajo exploré el uso del aprendizaje activo a lo largo del autoaprendizaje para tareas de desambiguación de sentidos verbales del español en Cardellino et al. [Cardellino et al., 2015].

Capítulo 4

Aprendizaje supervisado para desambiguación de sentidos verbales

4.1 Estructura

Introduciré en el presente capítulo las bases sobre las cuales construiré el trabajo de la tesis. En los capítulos siguientes iré construyendo gradualmente sobre lo que se desarrolla en este capítulo. Los resultados de este capítulo se utilizarán como base de referencia para evaluar el impacto de las adiciones trabajadas en los capítulos subsiguientes.

Recapitulando del Capítulo 1, Sección 1.1, expongo el problema en cuestión. Tengo disponible un conjunto de datos etiquetado de sentidos verbales desambiguados que es pequeño. El desarrollo de un recurso manual anotado es costoso porque la anotación manual es realizada por expertos en el dominio. Debido a la naturaleza del lenguaje, la distribución de las clases (es decir, los sentidos) en el conjunto de datos es Zipfian [Zipf, 1949]. Por lo tanto, tengo un recurso donde la distribución de las anotaciones está altamente sesgada hacia una clase más frecuente. Este recurso plantea dos desafíos: el número de ejemplos anotados y la distribución de las etiquetas.

Para obtener un algoritmo para desambiguación de sentidos verbales el enfoque más simple es entrenar un clasificador automático supervisado. Sin embargo, con un enfoque puramente supervisado y como consecuencia de los retos mencionados anteriormente, tengo que lidiar con dos problemas: el sobreequipamiento de los datos y el sesgo del algoritmo para etiquetar todo como la clase más frecuente. Demostraré la existencia de estos problemas y discutiré con los resultados y conclusiones de este capítulo lo que puedo hacer para mejorar el proceso de aprendizaje para superar estos problemas.

Para explorar estos desafíos, se formalizan con la siguiente hipótesis:

Hipótesis I *El temaño del conjunto de datos afecta la calidad final del modelo.*

Para poder aceptar o rechazar la hipótesis, se divide en subhipótesis más específicas:

Subhipótesis 1.1 *Con conjuntos de entrenamiento más grandes el desempeño mejora.*

Subhipótesis 1.2 *Con conjuntos de datos más grandes el clasificador tiende menos a sobreajustar.*

Subhipótesis 1.3 *La cantidad de classes afecta la tendencia a sobreajustar.*

Subhipótesis 1.4 *Los modelos lineales tienen menos tendencia a sobreajustar que los modelos no lineales.*

Estas hipótesis se aceptarán o rechazarán usando el siguiente esquema:

- El experimento 4.2 informa del rendimiento de un modelo a medida que aumenta el número de ejemplos. El desempeño es medido por la macro y la F1 media ponderada (Métrica 1). Resultados mostrados en la Sección 4.4.3 sirven para aceptar Hipótesis 1.1, que el rendimiento mejora con un conjunto de entrenamiento más grande.

- El experimento 4.3 reporta la variación de diferentes modelos entrenados sobre diferentes subconjuntos de los datos de entrenamiento. Esta varianza se mide por Métrica 3 que refleja la tendencia de un modelo a adaptarse, con la ayuda de la curva de aprendizaje. A partir de este experimento y métrica, utilizando diferentes visualizaciones según la tarea, muestro que el clasificador es menos propenso a sobredimensionar con conjuntos de datos más grandes (Hipótesis 1.2, en Sección 4.4.4), con menos clases (Hipótesis 1.3, en Sección 4.4.5). También muestro que los modelos lineales tienen menos tendencia a sobreajustarse (Hipótesis 1.4, en Sección 4.4.6).

En la sección 4.2 Recapitulo sobre el trabajo previo realizado en el aprendizaje supervisado tanto para el inglés como para desambiguación de sentidos verbalesdel español, con un enfoque especial en el problema del aprendizaje con muy pocos ejemplos y desequilibrio de clases.

En la Sección 4.3 explico todos los puntos relevantes relacionados con lo que se utiliza para llevar a cabo la experimentación de este capítulo. Primero, en la Sección 4.3.1 Presento los recursos con los que trabajo. El corpus anotado de SenSem sobre el que se realiza el resto del trabajo de tesis y SemEval Corpus con el fin de comparar con el estándar y asegurar que no haya sesgos relacionados con el lenguaje. Estos recursos están representados en esta etapa con lo que denomino *ingeniería de atributos* tomado de los recursos mismos, esto es lo que presento en la Sección 4.3.2. También exploro técnicas para reducir la dimensionalidad de las representaciones y veo cómo este es un paso necesario para poder entrenar a algunos clasificadores. Introduzco los algoritmos de clasificación usados para la tarea de desambiguación de sentidos automática en

Seccón 4.3.3. La sección 4.3.4 enumera los experimentos realizados. Y la Sección 4.3.5 lista el conjunto de métricas que uso para medir los experimentos.

La sección 4.4 informa de los resultados de los experimentos y los analiza para aceptar o rechazar las hipótesis planteadas en el capítulo.

Finalmente la Sección 4.5 extrae las conclusiones de este capítulo, recapitulando las Hipótesis y aceptándolas o rechazándolas según las pruebas recogidas en los resultados. En él se exponen las deficiencias de los métodos que se examinan en este capítulo y lo que quiero lograr en el siguiente y termina enumerando el trabajo futuro.

4.2 Trabajo relevante

I recall from Chapter 3, in Section 3.2.1, there is some work done for English desambiguación de sentidos verbales and some for Spanish desambiguación de sentidos, but not much for Spanish desambiguación de sentidos verbales. To my knowledge the work done exclusively for Spanish desambiguación de sentidos verbales is almost non-existent. Moreover the features used in English desambiguación de sentidos verbales, like selectional preferences and abstract semantic features, are not available for Spanish.

Recuerdo del capítulo 3, en la sección 3.2.1, hay algo de trabajo hecho para el desambiguación de sentidos verbales del inglés y algo para el desambiguación de sentidos del español, pero no mucho para el desambiguación de sentidos verbales del español. Hasta donde yo sé, el trabajo realizado exclusivamente para desambiguación de sentidos verbales del español es casi inexistente. Por otra parte, las características utilizadas en desambiguación de sentidos verbales del inglés, como las preferencias selectivas y las características semánticas abstractas, no están disponibles para el español.

En particular, para diseñar el sistema de aprendizaje supervisado de la máquina me inspiré en el sistema de *It Make Sense* [Zhong and Ng, 2010], y usé un esquema similar a la de ellos.

Primero, hubo un paso de preprocesamiento de los corpus etiquetados donde se analizó automáticamente. Luego utilizo la información disponible para generar características que representen las instancias a utilizar como datos de entrenamiento. Finalmente se probaron diferentes clasificadores supervisados en la fase de experimentación.

Para diseñar las funcionalidades me basé en el trabajo ya mencionado en Section 3.2.1. Especialmente de Ye y Baldwin [Ye and Baldwin, 2006], Màrquez et al. [Màrquez et al., 2007], y Montoyo et al. [Montoyo et al., 2011].

4.3 Metodología

En esta sección, explico los métodos generales utilizados para llevar a cabo los experimentos de este capítulo. Una vez más, para el alcance de esta tesis, las palabras

dataset (o conjunto de datos) y *corpus* son intercambiables. Por otro lado, cuando uso la palabra modelo me refiero al resultado de entrenar a un *clasificador* con una *representación* específica para un *corpus* específico.

4.3.1 Recursos

English. Como el proyecto se centra principalmente en desambiguación de sentidos verbales, hay dos corpus con los que he trabajado: SenSem [Alonso et al., 2007] etiquetada como corpus de sentidos verbales en español y SemEval-2007 Task 17 [Pradhan et al., 2007] corpus para inglés.

El enfoque principal es el trabajo en SenSem, ya que las áreas de desambiguación de sentidos, en general, y de desambiguación de sentidos verbales, en particular, pueden para el idioma español beneficiarse enormemente de los resultados que puedo encontrar; especialmente porque el recurso básico existe pero es pequeño y necesita mejoras. Además, como este es el trabajo de un hispanohablante, seguramente es un interés de investigación para mí.

SemEval, al ser de lengua inglesa, se utiliza sobre todo como comparación con el trabajo en castellano de este capítulo, ya que el trabajo en inglés en la zona es más sólido que el de castellano.

4.3.1.1 SenSem

SenSem [Alonso et al., 2007] es un corpus anotado manualmente tanto para castellano como para catalán. Sirve como recurso principal para los experimentos en español. Contiene los 248 verbos más comunes del español, anotados con sentidos definidos en un léxico provisto, algunos de ellos con mapeos a la Ontología Española de Wordnet [Montraveta et al., 2008].

Una versión del corpus SenSem tiene etiquetas pos anotadas automáticamente con Freeling [Padró and Stanilovsky, 2012]. Sin embargo, estas etiquetas están anotadas en un nivel basado en palabras, por lo que hay una gran proporción de ellas anotadas con la etiqueta incorrecta (por ejemplo, verbos anotados como sustantivos). Además, el español tiene algunas palabras que son multipalabras (es decir, palabras formadas por dos o más términos diferentes) cuya etiqueta no es la misma que la de cada una de las palabras que componen la multipalabra. Por ejemplo, "más_allá_de" se etiqueta como una multi-palabra con la etiqueta PoS "SP", es una preposición, sin embargo, las palabras "más" y "allá" son por sí mismas adverbios y sólo "de" es una preposición.

Con el fin de recopilar información más útil para la extracción de características, se realizó un preprocesamiento en dos pasos del corpus SenSem. Primero, una anotación automática usando un analizador estadístico. En este paso las frases de SenSem, que son tokenizadas, son dadas a Freeling. Utilizando el analizador de dependencias estadísticas, las frases se anotan con: lema, etiqueta de parte del habla, información morfosintáctica y triples de dependencias. Además, hay detección de multi-palabras y reconocimiento de entidades nombradas (tratadas por Freeling como multi-palabras).

Sin embargo, la anotación automática no es suficiente ya que los errores no sólo provienen de Freeling, sino también de otros problemas que tiene SenSem: frases sin sentido definido, frases en las que no está presente el verbo desambiguar y frases truncadas antes de terminar. Por esta razón, el segundo paso del preprocesamiento era manual, donde cada una de las frases anotadas automáticamente donde se perdía el lema principal a desambiguar (por equivocación, no estar correctamente marcado en el recurso original, etc.), se encuentra manualmente. Además de esto, todos los casos que son erróneos en el corpus original (por ejemplo, oraciones truncadas o oraciones sin un sentido definido) fueron descartados.

Después del preprocesamiento del texto, el corpus SenSem se dividió en tren/prueba. Para ello se filtran todos aquellos sentidos con menos de dos ocurrencias en el corpus y se dividen los restantes sentidos con muestreo estratificado utilizando un 80 % para el entrenamiento y un 20 % para las pruebas, donde el corpus del tren tiene al menos dos ocurrencias de cada sentido y el corpus de prueba tiene al menos una ocurrencia de cada sentido. Estas divisiones también preservan la distribución de las clases observadas en todo el conjunto de datos. Esto implica que en cada división no se pueden encontrar más ejemplos de una clase de los que se podrían encontrar en una muestra estratificada del conjunto de datos, es decir, una muestra que conserve la distribución de todo el conjunto de datos. Sin embargo, esto no es necesariamente válido para las clases minoritarias, porque al menos un ejemplo debe encontrarse en cada escisión, incluso si eso implica una representación excesiva de la clase.

El corpus de prueba es siempre el mismo para todos los experimentos de la tesis, y se lleva a cabo desde el entrenamiento hasta el uso para medir el rendimiento de los diferentes algoritmos estudiados en esta tesis. Necesito dos ocurrencias en el corpus de entrenamiento de cada sentido porque en los experimentos futuros (los que comprenden el enfoque semi-supervisado), se necesita un corpus de validación para ayudar al criterio de parada temprana de los algoritmos y por lo tanto se usa el mismo enfoque para dividir el tren y el corpus de prueba (en este caso con al menos un ejemplo por sentido para cada división).

La tabla 4.1 muestra algunas de las estadísticas del corpus SenSem tras el preprocesamiento del texto y la eliminación de frases erróneas. Quiero centrarme especialmente en el número medio de instancias para el sentido más frecuente por lema en comparación con el número medio de instancias para el segundo sentido más frecuente por lema. Es claro ver el desequilibrio de los sentidos en los corpus ya que la clase más frecuente tiene más de 3 veces más ocurrencias que la siguiente.

En resumen, la versión original del SenSem se analizó automáticamente con Freeling para recopilar más datos para utilizarlos en la construcción de las características, y luego se revisó manualmente para corregir errores y descartar frases incorrectas (por ejemplo, frases truncadas). Después de esto, se dividió en conjuntos de datos de tren y de prueba.

Estadísticas	**Valores**
No. de instancias (antes de filtrar)	23938
No. de instancias (después de filtrar)	20138
No. de lemas (antes de filtrar)	248
No. de lemas (después de filtrar)	208
No. de sentidos (antes de filtrar)	772
No. de sentidos (después de filtrar)	732
No. promedio de sentidos por lema	3.52
No. promedio de instancias por lema	96.82
No. promedio de instancias por sentido	27.51
No. promedio de instancias para el sentido más frecuente por lema	67.08
No. promedio de instancias para el segundo sentido más frecuente por lema	19.67

Cuadro 4.1: Estadísticas del SenSem

4.3.1.2 SemEval

Para comparar los resultados con un escenario experimental estándar, replico los experimentos usando el corpus SemEval-2007 Task-17 corpus [Pradhan et al., 2007]. El corpus tiene información de los sentidos de 100 lemas diferentes, 65 verbos y 35 sustantivos (a diferencia de SenSem que sólo tiene información sobre los sentidos verbales). Dado que este trabajo quiere evaluar los resultados de la desambiguación del sentido del verbo en español, se decidió omitir los 35 sustantivos de la configuración experimental del inglés y trabajar sólo con los verbos.

A diferencia de SenSem, el corpus de SemEval ya estaba dividido en cuerpos de tren y de prueba. Hay sentidos que sólo ocurren una vez en todo el corpus, ya sea en el tren o en los datos de prueba. Para no alterar SemEval de manera que siga siendo útil para la comparación, no se realizó ningún filtrado del conjunto de datos. Sin embargo, dada la naturaleza de los clasificadores utilizados aquí, aquellos casos en los que el sentido sólo existe en el corpus de pruebas no tendrán forma de ser predecibles por los clasificadores.

Otras diferencias entre SenSem y SemEval es la forma en que se presentan los ejemplos. En SenSem cada ejemplo representa sólo una frase en la que un token específico es el que está siendo desambiguado. SemEval, por otro lado, puede estar conformado por múltiples oraciones, por ejemplo, especificando nuevamente el token a ser desambiguado. Estas frases no tienen ninguna otra información aparte de estar ya tokenizadas.

Para obtener información adicional, el corpus SemEval fue preprocesado utilizando las herramientas Stanford CoreNLP [Manning et al., 2014]. Similar a SenSem, la información recopilada fue: lema, etiqueta de parte del habla (con información morfosintáctica) y triples de dependencia. También había reconocimiento y clasificación de entidades con nombre, pero la herramienta (a diferencia de Freeling) trata cada palabra de una entidad con nombre como un valor único. Además de esto, cada ejemplo fue recortado para que sólo quedara la oración que sostenía la palabra meta.

Como el corpus de SemEval también se ocupa de casos como la jerga americana o algunos otros usos informales de la parte inglesa del corpus que necesitaban ser revisados manualmente para encontrar el token objetivo después de preprocesar las frases.

La tabla 4.2 muestra algunas de las estadísticas del corpus SemEval tras el preprocesamiento del texto y la eliminación de frases erróneas. Como puedes ver, el número de lemas e instancias es mucho menor que SenSem, pero el promedio de instancias por lema y por sentido duplica el de SenSem. Sin embargo, lo que le pasó a SenSem le pasa a SemEval en cuanto al sentido más frecuente en comparación con el segundo sentido más frecuente. Es claro ver el desequilibrio de los sentidos en los corpus ya que la clase más frecuente tiene más de 5 veces más ocurrencias que la siguiente. Como puedo ver, el desequilibrio de los datos es aún peor que con SenSem.

Estadísticas	**Valores**
No. de instancias (antes de filtrar)	11280
No. de instancias (después de filtrar)	10167
No. de lemas (antes de filtrar)	65
No. de lemas (después de filtrar)	51
No. de sentidos (antes de filtrar)	230
No. de sentidos (después de filtrar)	194
No. promedio de sentidos por lema	3.8
No. promedio de instancias por lema	199.35
No. promedio de instancias por sentido	52.41
No. promedio de instancias para el sentido más frecuente por lema	148.78
No. promedio de instancias para el segundo sentido más frecuente por lema	28.75

Cuadro 4.2: Estadísticas SemEval

En resumen, al igual que SenSem, SemEval fue automáticamente anotado usando Stanford CoreNLP, y revisado manualmente más tarde para corregir errores y otros errores. A diferencia de SenSem, el corpus ya está dividido en subconjuntos de tren y de prueba.

4.3.2 Atributos

Para entrenar un algoritmo de clasificador supervisado, necesito definir algunas características para representar las instancias del conjunto de datos. Tales instancias, particularmente para desambiguación de sentidos verbales, son definidas por la palabra (específicamente el verbo) para ser desambiguada en una oración. Con esa palabra como foco se utilizan las siguientes características para representar la instancia:

- La palabra principal.
- El lema de la palabra principal.
- La etiqueta pos de la palabra principal: para el caso del español, sólo se usa la forma abreviada (generalmente las 2 o 3 primeras letras).

- Para el caso del español, la información morfosintáctica de la palabra principal es dada por separado de la etiqueta pos.
- La bolsa de palabras de una ventana simétrica de 5 palabras (i.e. 5 palabras antes y después de la palabra principal): este atributo representa el número de ocurrencias de cada palabra que rodea a la palabra principal sin importar su posición.
- Las palabras, lemas y etiquetas pos de las palabras alrededor del verbo en una ventana de 5 palabras.
- Los bigramas y trigramas formados por las palabras que rodean al verbo.
- Las triplas de dependencias formada por el verbo, la relación y las palabras que dependen de la palabra principal o de las palabras de las que depende el verbo.

Ejemplo 4.1. La extracción de atributos se ejemplifica con la siguiente frase extraída del corpus SenSem. Muestra todas las características extraídas de él.

Sentence and verb to disambiguate

"Una serie de controles para que las industrias y los bancos se **abran** a un control efectivo desde EE.UU.".

Extracted features

Main word abran

Main word's lemma abrir

Main word's part-of-speech tag VMS

Main word's morphosyntactic info

Part of speech Verb

Type Main

Mood Subjunctive

Tense Present

Person Third

Number Plural

Bag-of-words in 5-word window { a, bancos, control, desde, efectivo, industrias, los, se, un, y }

Collocational words, lemmas and tags in a 5-word window

Main word position - 5 { industrias, industria, NC }

Main word position - 4 { y, y, CC }

Main word position - 3 { los, el, DA }

Main word position - 2 { bancos, banco, NC }

Main word position - 1 { se, se, P0 }

Main word position + 1 { a, a, SP }

Main word position + 2 { un, uno, DI }

Main word position + 3 { control, control, NC }

Main word position + 4 { efectivo, efectivo, AQ }

Main word position + 5 { desde, desde, SP }

Bigrams/Trigrams

Left bigram { bancos, se }

Left trigram { los, bancos, se }

Right bigram { a, un }

Right trigram { a, un, control }

Inbound dependency triples { (que - conj - target) , (industria - suj - target), (se - pass - target), (a - cc - target), (desde - cc - target) }

Outbound dependency triple { (target - S - para) }

4.3.2.1 Selección de atributos

La representación obtenida por las características previamente presentadas es muy escasa, ya que muchas de ellas aparecerán una o dos veces en todo el conjunto de datos. Además, la cantidad de combinaciones posibles para ello terminará con una gran cantidad de características para representar cada instancia. Esto es algo que, como explicaré más adelante, no todos los clasificadores supervisados pueden manejar.

Las técnicas de reducción de la dimensionalidad mejoran la representación de los datos al reducir el número de variables consideradas. Representan una instancia a través de las variables principales. En particular utilizo técnicas de selección de características en las que las principales variables son las que retienen más información sobre los datos representados.

La selección de características se basa en una suposición subyacente: las características utilizadas para representar los datos tienen mucha redundancia, ya que la creación se hace *ad-hoc* y puede no ser perfecta (por ejemplo, desconocimiento del dominio, imposibilidad de obtener automáticamente características más relevantes, etc.). Por lo tanto, hay mucho ruido ya que hay muchas características para representar los datos y muchas de ellas pueden ser causadas por el ruido en lugar de la información útil real. Este ruido y la redundancia de las características también afecta a la tendencia de sobreajuste de un modelo, y la aplicación de alguna selección puede ayudar a reducir esto a medida que la representación se vuelve más general, eliminando el ruido y suavizando la representación.

Para la experimentación uso la *selección de características univariadas* con el valor F ANOVA de Fisher [Fisher, 1921] para elegir las 10 mil características más representativas. Esta técnica funciona seleccionando las mejores características basándose en pruebas estadísticas univariadas. Puede ser visto como un paso de preprocesamiento

a un estimador. El método estima el grado de dependencia lineal entre dos variables aleatorias.

4.3.2.2 Hashing de atributos

La selección de características, como una forma de reducir la dimensionalidad del vector de entrada de un clasificador, alcanza algunas propiedades interesantes como se describió anteriormente. Sin embargo, añade al coste computacional de la formación de un modelo. Para filtrar las características uno necesita explorar todo el conjunto de datos y luego seleccionar las características desde allí. Además, no hay garantía de que los nuevos ejemplos para anotar serán representados por esas características seleccionadas.

Hashing de atributos [Weinberger et al., 2009], también conocido como hashing trick, es una forma eficiente de vectorizar características. A diferencia de la selección de características, no se basa en la suposición de que algunas características contienen más información que otras. Consiste en aplicar una estructura de datos específicamente diseñada para optimizar la dimensionalidad asumiendo representaciones dispersas. En este método, los rasgos se vectorizan en una matriz de longitud fija mediante la aplicación de una función hash a los rasgos y el uso del valor como índice de la matriz. La cuenta de la característica entonces se almacena en la posición correspondiente del arsenal.

Exploro esta técnica como una forma de representar datos con una cantidad limitada de memoria sin fijar la representación sólo a algunas características seleccionadas. Esto es útil en el caso de tener ejemplos en los que las características seleccionadas usando el método mostrado anteriormente no están presentes.

4.3.3 Clasificadores

En las siguientes secciones se están probando diferentes clasificadores para encontrar los que mejor se adaptan al problema. Existen principalmente dos tipos de clasificadores: lineales y no lineales.

4.3.3.1 Clasificadores Lineales

Un algoritmo clasificador lineal realiza el proceso de clasificación utilizando una combinación lineal de las características. Para problemas de clasificación binaria se puede ver la operación como dividir un espacio de entrada de alta dimensión con un hiperplano: todos los puntos de un lado del hiperplano se consideran de una clase y todos los puntos del otro lado son de la otra clase. Estos clasificadores tienen la ventaja de ser más fáciles y rápidos de entrenar, a la vez que alcanzan niveles de precisión comparables a los de los clasificadores no lineales.

Naive Bayes Multinomial Los métodos de Naive Bayes [Russell and Norvig, 2003] son un conjunto de algoritmos de aprendizaje supervisado basados en la aplicación del

teorema de Bayes con la suposición "ingenua" de independencia entre cada par de características. El método ingenuo multinomial Bayes, implementa el ingenuo algoritmo Bayes para datos distribuidos multinomiales.

Clasificador de Regresión Logística La regresión logística [Walker and Duncan, 1967], a pesar de su nombre, es un modelo lineal para la clasificación más que para la regresión. La regresión logística también se conoce en la literatura como regresión logarítmica, clasificación de máximaentropía (MaxEnt) o clasificador logarítmico lineal. En este modelo, las probabilidades que describen los posibles resultados de un único ensayo se modelan mediante una función logística.

Support Vector Machines (con Kernel Lineal) Las máquinas vectoriales de soporte [Cortes and Vapnik, 1995] son modelos de aprendizaje supervisado con algoritmos de aprendizaje asociados que analizan los datos utilizados para la clasificación y el análisis de regresión. Dando un conjunto de ejemplos de entrenamiento, cada uno marcado como perteneciente a una u otra de dos categorías, un algoritmo de entrenamiento SVM construye un modelo que asigna nuevos ejemplos a una u otra categoría, convirtiéndolo en un clasificador binario no probabilístico.

Como en capítulos futuros necesitaré un clasificador probabilístico, ya que algunos de los algoritmos semi-supervisados que uso están basados en esto, y también una red neuronal, ya que otros algoritmos semi-supervisados están basados en eso; este clasificador se usa principalmente como una comparación.

4.3.3.2 Clasificadores no lineales

Un clasificador no lineal utiliza una función más compleja para aproximarse mejor al problema. Las características pueden combinarse de forma no lineal. Así se pueden encontrar patrones más complejos en los datos. Además, algunos problemas son estrictamente no lineales separables. Un problema común con este tipo de clasificadores es la tendencia a sobredimensionar los datos.

Árboles de Decisión Los árboles de decisión [Quinlan, 1986] son un método de aprendizaje supervisado utilizado para la clasificación y regresión. El objetivo es crear un modelo que prediga el valor de una variable objetivo mediante el aprendizaje de reglas de decisión simples inferidas a partir de las características de los datos.

Perceptrón multicapa Un perceptrón multicapa (MLP) [Rosenblatt, 1962] es un modelo de red neuronal artificial que asigna conjuntos de datos de entrada a un conjunto de salidas apropiadas. Un MLP consiste en múltiples capas de nodos en un gráfico dirigido, con cada capa completamente conectada a la siguiente. A excepción de los nodos de entrada, cada nodo es una neurona (o elemento de procesamiento) con una función de activación no lineal. MLP utiliza una técnica de aprendizaje supervisado

llamada backpropagation para la formación de la red. MLP es una modificación del perceptrón lineal estándar y puede distinguir datos que no son linealmente separables.

4.3.4 Experimentos

Los experimentos, junto con las métricas, son fundamentales para probar las hipótesis de éste y cada uno de los siguientes capítulos de la tesis. Los experimentos y las métricas están estrechamente relacionados, ya que un experimento sólo es útil a través de una métrica que da una idea de su rendimiento.

El experimento 4.1 es un experimento general que se aplica utilizando una arquitectura split tren/prueba, que ya está presente en los recursos. La idea de este esquema es comparar el rendimiento de diferentes clasificadores y representaciones para todo el conjunto de datos. Este tipo de experimento es necesario como comparación de referencia para los posibles modelos.

Experimento 4.1.

4.1a Entrenar un modelo con el subconjunto de entrenamiento del corpus para cada lema.

4.1b Clasificar el corpus de evaluación, de cada lema, con el modelo entrenado.

4.1c Comparar los resultados predichos por el modelo, de cada lema, con el estándar usando alguna métrica.

El experimento 4.2 está diseñado para comprobar el rendimiento de un modelo a medida que aumenta el número de ejemplos, suponiendo que un mayor número de ejemplos mejorará el rendimiento del clasificador. La idea es dividir el corpus original en partes más pequeñas y formar nuevos modelos añadiendo gradualmente nuevos ejemplos.

Experimento 4.2.

4.2a Elegir un número de *particiones* para el corpus.

4.2b Para cada lema, dividir el corpus usando el número de particiones, asegurándose que hay una partición con todas las etiquetas disponibles y usar ese para el entrenamiento inicial.

4.2c Entrenar un modelo.

4.2d Test the model using the test set and store the results.

4.2e Evaluar el modelo usando el conjunto de evaluación y guardar los resultados.

4.2f Agregar la siguiente partición a los datos de entrenamiento y reentranar el modelo.

4.2g Finalizar el modelo cuando todas las particiones fueron usadas.

Finally, Experiment 4.3 is similar to Experiment 4.2 as it also tries to check the performance of a model by gradually adding new data. However, this experiment objective is to measure the variance of different models trained over different subsets of the training data to assess the impact of the amount of data on how the model overfits the data.

Finalmente, el experimento 4.3 es similar al experimento 4.2 ya que también intenta comprobar el rendimiento de un modelo añadiendo gradualmente nuevos datos. Sin embargo, el objetivo de este experimento es medir la varianza de diferentes modelos entrenados sobre diferentes subconjuntos de los datos de entrenamiento para evaluar el impacto de la cantidad de datos sobre cómo el modelo se sobrepone a los datos.

Experimento 4.3.

4.3a Para cada lema, particionar aleatoriamente todo el corpus en un *número de particiones* seleccionada. El temaño de las particiones debe ser uniforme. Asegurarse que hay una partición con todas las clases del conjunto de datos y utilizarlo para la iteración inicial.

4.3b Tomar el conjunto de datos inicial y particionarlo para entrenamiento y evaluación.

4.3c Entrenar el modelo con el conjunto de entrenamiento obtenido en el paso previo y guardar las predicciones sobre los conjuntos de entrenamiento y evaluación del paso anterior.

4.3d Agregar la siguiente *partición* al conjunto de datos y repetir desde el paso 4.3b.

4.3e Cuando todas las *particiones* sean agregadas, repetir todo el algoritmo n veces con un nuevo conjunto de particiones aleatorias.

La idea detrás de la repetición de este algoritmo n veces es representar cómo el entrenamiento del mismo modelo sobre diferentes conjuntos de datos varían, y encontrar un estimador de la tendencia del modelo a sobredimensionar esos conjuntos de datos.

4.3.5 Métricas

Los experimentos definidos en la sección anterior requieren métricas para ser evaluados. Las métricas funcionan junto con las visualizaciones para mostrar diferentes vistas de un resultado. Sin embargo, ambas herramientas están sujetas a interpretación, ya que cada métrica y visualización destaca algunos aspectos de la evaluación, mientras que oscurece otros. Trabajo con tres tipos de métricas en esta tesis: rendimiento, significación y tendencia al sobreajuste. Cada tipo de métrica muestra una vista parcial de los resultados. Es difícil (si no imposible) combinar diferentes métricas sin perder el objetivo que tiene la métrica específica.

4.3.5.1 Métricas de desempeño

Las métricas de desempeño son las que miden qué tan bien lo hace un experimento con respecto a un corpus de pruebas, realizado a partir del corpus de entrenamiento, que aproximadamente representa, en menor escala, las mismas clases. Estas métricas son las que se usan en Experimentos 4.1 y 4.2, y comparan un modelo con otro.

Cuando se trata de tareas de clasificación automática en PNL necesito una métrica para determinar el rendimiento de un clasificador para todas las clases. En precisión, una métrica estándar, el desequilibrio de una clase puede afectar el resultado final. En precisión y recordación, también métricas estándar, necesito comprobar el valor para cada clase, difícil cuando se trata de muchas clases. En particular, desambiguación de sentidos es una tarea que trata con una distribución Zipfian [Zipf, 1949]. Esto hace que una métrica sesgada hacia la clase más frecuente afecte la percepción de qué tan bien lo está haciendo un algoritmo.

Exactitud La exactitud se define como la proporción de instancias correctamente clasificadas. Este es un ejemplo de una métrica sesgada, ya que con muchas instancias que pertenecen a una clase, cualquier clasificador sesgado a la clase más frecuente (MFC) puede tener un buen desempeño.

Precision, recall and F1-score The first two metrics generally go side by side as one of them is not enough to measure the performance by itself. Although originally defined for information retrieval [Rijsbergen, 1979], they have become standard in many classification tasks. Particularly, these are the metrics used by SemEval competition to compare results. **Precision** is defined as the number of instances classified with certain class that are effectively instances of that class. **Recall** is defined as the number of instances of a class that were effectively classified with such class. These metrics have the advantage over accuracy of showing results for each sense, which can be a good indicator when dealing with a small amount of classes. However problems with many classes, like in desambiguación de sentidos verbales, need simpler values to measure the performance, thus making it easier to compare to an established baseline. The **F1-score**, a harmonic mean between precision and recall does solve the problem of having a single value, but still deals with having one value for every possible class.

Precisión, exhaustividad y F1-score Las dos primeras métricas generalmente van una al lado de la otra, ya que una de ellas no es suficiente para medir el rendimiento por sí sola. Aunque originalmente se definieron para la recuperación de información [Rijsbergen, 1979], se han convertido en estándar en muchas tareas de clasificación. En particular, estas son las métricas utilizadas por la competencia SemEval para comparar resultados. **Precisión** se define como el número de instancias clasificadas con cierta clase que son efectivamente instancias de esa clase. **Exhaustividad** se define como el número de instancias de una clase que fueron efectivamente clasificadas con dicha clase. Estas métricas tienen la ventaja sobre la precisión de mostrar resultados para cada

sentido, lo que puede ser un buen indicador cuando se trata de una pequeña cantidad de clases. Sin embargo, los problemas con muchas clases, como en desambiguación de sentidos verbales, necesitan valores más simples para medir el rendimiento, lo que facilita la comparación con una línea de base establecida. El **F1-score**, una media armónica entre precisión y memoria resuelve el problema de tener un único valor, pero aún así se trata de tener un valor para cada clase posible.

Promedios de precisión, exhaustividad y F1-score Los promedios tratan con el problema de tener resultados de precisión y recuperación para múltiples clases comprimidas en una sola clase. Hay diferentes tipos de promedios:

Promedio Macro [Manning et al., 2008] se define como el media no ponderada de los valores de la métrica para cada clase. Con este promedio las clases menos frecuentes son tan importantes como las más frecuentes, sin embargo, también significa que un desequilibrio de clase extremo impacto en los resultados finales.

Promedio Micro [Manning et al., 2008] se calcula obteniendo cada verdadero positivo (tp), falso positivo (fp), verdadero negativo (tn) y falso positivo (fp) negativo (fn) en general, en lugar de utilizar una base por clase. Con eso información se calcula la precisión y la recuperación. Esto significa que los con más ocurrencias son más relevantes ya que las clases tp/tn/fp/fn pertenecen en su mayoría a las clases más frecuentes.

Promedio Ponderado se calcula en base a cada clase, pero luego cada una de ellas es el promedio ponderado por el número de ocurrencias que tiene. Similar a lo que sucede con el micro promedio, las clases con más ocurrencias tienen más relevancia en los resultados finales.

Precisión de la clase más frecuente (PMFC) y media de exhaustividad de clases menos frecuentes (RMLFC) . Estas dos métricas fueron exploradas como una posible adición a las métricas explicadas anteriormente. El PMFC tiene la idea de revelar la tendencia de un clasificador a categorizar todas las instancias como parte del MFC, cuanto más bajo sea el valor, más sesgado estará el clasificador. RMLFC es el promedio macro de la recuperación en las clases que omiten las más frecuentes, mostrando así cuán bueno es el clasificador para clasificar correctamente las instancias de las clases menos frecuentes. Aunque algunos experimentos se realizaron siguiendo estas métricas, no hubo un impacto real o diferencia visible para elegir estas métricas sobre los promedios habituales.

Métricas elegidas Después de una serie de experimentos se concluye que ninguna de estas métricas puede mostrar el desempeño de los clasificadores y el sesgo de los mismos por sí mismo, sino que la combinación de ellos muestra un mejor patrón.

Métrica 1. El desempeño de un experimento es mostrado usando la comparación de las siguientes métricas:

1a Promedio Macro F1-score.

1b Promedio Ponderado F1-score.

Como las clases desequilibradas son un reto al que tengo que enfrentarme, es importante mostrar cómo esto afecta al rendimiento general de los modelos. Metric 1 es útil para ver si un modelo está teniendo un sesgo grande hacia la clase más frecuente.

4.3.5.2 Métricas de significancia

Las métricas de significación se utilizan para comparar diferentes modelos y comprobar si la diferencia en el rendimiento entre ellos es estadísticamente significativa o no. Estos son útiles para descartar dos, o más, modelos que funcionan de manera similar y cuya diferencia no mejora realmente los resultados sino que es simplemente un resultado del azar. Se usan en el experimento 4.1.

Coeficiente kappa de Cohen El coeficiente kappa de Cohen ha sido concebido originalmente como una métrica para medir el acuerdo entre anotadores en tareas de etiquetado manual. Puede ser utilizado de alguna manera como una métrica de rendimiento comparando los resultados de las pruebas de un modelo con la verdad del terreno y normalizando así la precisión, ajustando el resultado para el acuerdo esperado con el clasificador perfecto por casualidad [Cohn and Cohn,]; sin embargo, encuentro útil aplicarlo como una medida de cuán significativamente diferentes son los resultados de dos clasificadores. Para comprobar si el rendimiento de los diferentes clasificadores es estadísticamente significativo se utiliza el Metric 2.

Métrica 2.

2a Tomar un par de clasificadores, un conjunto de datos y una representación.

2b Entrenar el modelo para cada clasificador.

2c Evaluar el modelo sobre el conjunto de evaluación para el clasificador entrenado.

2d Usar el coeficiente kappa de Cohen para analizar el acuerdo entre anotadores para el par de clasificadores sobre los resultados predichos en el corpus de evaluación.

Otras métricas de significación como la Kappa de Fleiss [Fleiss et al., 1971] y la alfa de Krippendorff [Krippendorff, 1970] también se han utilizado para evaluar la significación y la reproducibilidad, pero en este trabajo nos limitamos a la Kappa de Cohen porque es una métrica más estándar.

4.3.5.3 Curva de aprendizaje y tendencia a sobreajustar

Otra medida importante en este trabajo es la tendencia de un modelo a sobredimensionar los ejemplos de capacitación. Esto puede ser medido analizando el *error debido a la alta varianza*. Manning et al. [Manning et al., 2008] lo define como la variación de la predicción de los clasificadores aprendidos: mide cuán inconsistentes son las predicciones entre sí, sobre diferentes conjuntos de datos, no si son exactos o no.

Esta métrica está estrechamente relacionada con Experiment 4.3 porque la necesita para ver la variación del modelo para diferentes conjuntos de datos. Para calcularlo baso el algoritmo en el de Experiment 4.3:

Métrica 3. La curva de aprendizaje se calcula con los siguientes pasos:

3a Para cada lema, particionar aleatoriamente el total del corpus en un *número de particiones*. El tamaño de las particiones debe ser uniforme. Asegurarse de que una partición tenga todas las clases del conjunto de datos y utilizarla para la iteración inicial.

3b Tomar el conjunto de datos inicial y dividirlo en entrenamiento y evaluación.

3c Entrenar un modelo con el conjunto de entrenamiento del paso anterior y guardar las predicciones sobre los datos de entrenamiento y evaluación.

3d Agregar la siguiente *partición* a los datos y repetir desde el paso 3b.

3e When all the *splits* are added proceed to repeat the whole algorithm n times with a new set of random splits.

3f Cuando todas las *particiones* hayan sido agregadas, repetir todo el algoritmo n veces con un nuevo conjutno de particiones aleatorias.

3g Calcular la media y el error estándar de la media del error de clasificación para las predicciones de los conjuntos de entrenamiento y evaluación para cada paso del algoritmo.

La curva resultante muestra el error de varianza de un modelo a medida que aumenta el número de ejemplos de entrenamiento. Los modelos con mayor varianza son más propensos a sobredimensionar los datos.

4.4 Análisis de resultados

En la siguiente sección se reportan los resultados obtenidos a través de los diferentes experimentos. Estos resultados se perfilan con el uso de algunas herramientas de visualización que elegí para obtener un mejor entendimiento de acuerdo a las métricas asignadas a cada experimento para cada hipótesis. El objetivo final es sacar conclusiones sobre los resultados y utilizarlas para aceptar o rechazar las hipótesis.

Aunque miré y experimenté con diferentes técnicas de visualización y métricas, es importante recordar que cada métrica puede oscurecer algunos resultados o favorecer otros. Quería ser lo más objetivo posible y elegí esas métricas y visualizaciones para mostrar la mayor cantidad de información posible. Pero hay un equilibrio entre la cantidad de resultados que puedo mostrar y la cantidad de información que puedo obtener de ellos antes de que empiece a ser demasiado difícil de manejar.

En particular, para desambiguación de sentidos verbales mientras entrenaba un modelo por cada lema, tuve que seleccionar algunas visualizaciones, como gráficos de cajas y bigotes, que son buenas para dar una idea general de muchos resultados diferentes (en este caso, los resultados de cada lema) pero oscurecen los resultados más específicos.

4.4.1 Selección de representación

Como se explica en la sección 4.3.2, el problema de utilizar una representación con todas las características es su escasa y alta dimensionalidad. Esto afecta al rendimiento del clasificador perceptivo multicapa, ya que el gran número de combinaciones entre la capa de entrada y la capa oculta hace que sea demasiado grande para que la memoria de la máquina pueda manejarlo en el momento de escribir esta tesis. El clasificador perceptrón multicapa es necesario ya que es la base para el método semi-supervisado discutido en el capítulo 8. Esta es la razón para intentar algunas técnicas para limitar el número de características como se explicó anteriormente. Un primer enfoque fue utilizar la selección de características, sin embargo, esto tenía un inconveniente, ya que toma tiempo entrenar el algoritmo para seleccionar las características en base a alguna métrica (por ejemplo, varianza o ganancia de información). Por lo tanto, también experimenté con el *hashing trick* para ver cómo funcionaba esta técnica.

Para descartar que la reducción de la dimensionalidad afecte al resultado final entrené diferentes modelos utilizando los diferentes enfoques (reduciendo la dimensionalidad y no reduciéndola) y, utilizando la macro de la puntuación F1 y la media ponderada (Metric 1), comparé los resultados.

La figura 4.1 muestra los diferentes resultados de los diferentes modelos cambiando la representación de las características usando un gráfico de cajas y bigotes. La trama se estructura de la siguiente manera:

- Cada fila muestra los resultados de un corpus: SenSem y SemEval.
- Cada columna representa un clasificador: árbol de decisión, perceptrón multicapa (con dos capas ocultas una de 250 neuronas y la otra con 100 neuronas), naive Bayes, regresión logística y SVM.
- Cada grupo de gráficos de caja en cada gráfico representa una métrica: Promedio macro y ponderado del F1-score.

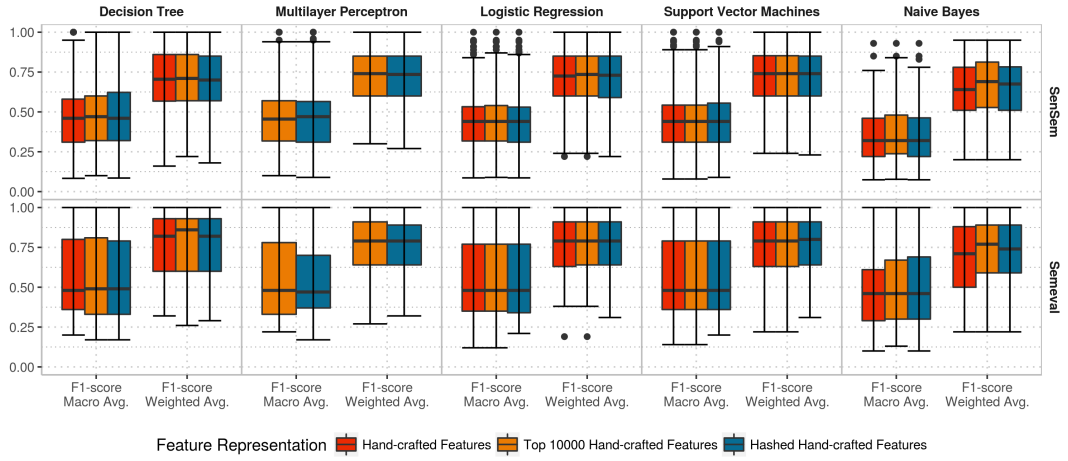


FIGURA 4.1: Comparación de representaciones de atributos para distintos clasificadores usando promedio macro y ponderado del F1-score

- Cada diagrama de caja de diferente color dentro de un grupo es el tipo de representación: todas las características, top 10 mil características seleccionadas por el método de selección de características y hashing trick.

- El gráfico de cajas representa la distribución de los valores de las métricas a través de sus cuartiles. Cada valor es el rendimiento para un lemma del corpus. La línea negra gruesa en medio de una trama de caja representa la mediana y los extremos al final de cada cuadrícula representa el valor máximo y mínimo (excepto eventuales valores atípicos) representado por puntos negros fuera de la trama de la caja).

Lo primero que hay que notar en la Figura es la ausencia de un diagrama de caja de representación en la columna correspondiente al perceptrón multicapa. Esto se debe a lo que se dijo en el párrafo anterior, acerca de que un perceptrón multicapa es incapaz de manejar todas las características posibles.

Del gráfico puedo deducir que no hay diferencia distintiva entre una representación y otra. En términos generales, los cuartiles son similares, y el rendimiento depende más del clasificador que de la representación. Sin embargo, es cierto que en muchos (si no en todos) los casos hay una mejora mínima en el rendimiento mediante el uso de la reducción de la dimensionalidad, probablemente porque hace que los clasificadores se ajusten menos cuando se elimina el ruido de las características raras. Aunque también es cierto que el uso de una técnica de selección de características tiene un poco más de rendimiento (aunque todavía mínimo) que el truco del hash, también hubo una compensación debido al costo computacional de hacer la selección de características para la tarea de clasificación, que en experimentos futuros usando técnicas semi-supervisadas se convertiría en un problema más pesado a medida que el número de nuevas características crezca a un alto ritmo.

Por estas razones, *el resto de los experimentos se llevaron a cabo utilizando la representación de hashing trick*, ya que no muestra una gran caída en el rendimiento sobre el equipo de prueba y también es el más barato de trabajar en términos computacionales.

4.4.2 Selección del clasificador

En los capítulos siguientes me centraré en el trabajo con los clasificadores de perceptrón multicapa. Esto, una vez más, debe tener un punto de comparación, particularmente en el capítulo 8, ya que las redes de escaleras se basan en un clasificador de percepción multicapa. Sin embargo, es importante comparar el perceptrón multicapa con otros métodos de clasificación para descartar la posibilidad de que sea una mala elección con la que trabajar desde el principio.

4.4.2.1 Selección de arquitectura

Uno de los principales hiperparámetros de una red neuronal es la arquitectura. En un perceptrón multicapa esto es la selección del número de capas y el tamaño de cada capa (es decir, el número de neuronas). Como la cantidad de datos es pequeña, existe un alto riesgo de que la red memorice los conjuntos de datos. Con suficientes neuronas disponibles, cada instancia puede ser mapeada a una ruta en la red. Por lo tanto, no puedo trabajar con una red neural muy profunda sin caer en este problema. Por eso sólo busco hasta tres capas ocultas.

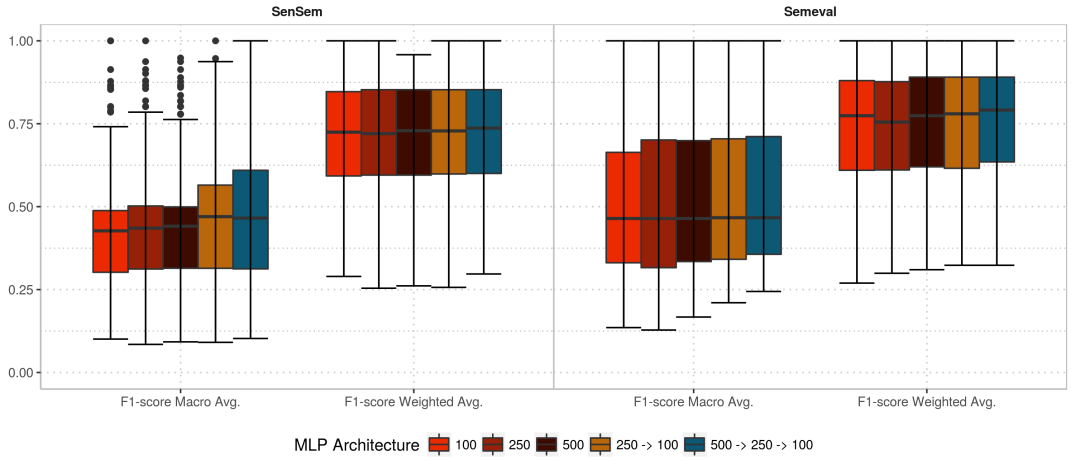


Figura 4.2: Comparación de arquitecturas del perceptrón multicapa

La figura 4.2 compara diferentes arquitecturas para un perceptrón multicapa usando las representaciones de trucos de hash (como se seleccionó antes). Esto se hace sólo para las siguientes arquitecturas:

1. Una capa oculta de 100 neuronas.
2. Una capa oculta de 250 neuronas.

3. Una capa oculta de 500 neuronas.
4. Dos capas ocultas: La primera de 250 neuronas y la segunda con 100 neuronas.
5. Tres capas ocultas: la primera con 500 neuronas, la segunda con 250 neuronas, y la tercera con 100 neuronas.

La Figura tiene una estructura similar a la de la Figura 4.1 ya que también es una trama de cajas y bigotes para mostrar el rendimiento de cada lemma:

- Cada columna representa un corpus: SenSem y SemEval.
- Cada grupo de gráficos de caja en cada gráfico representa una métrica: Promedio macro y ponderado del F1-score.
- Cada gráfico de caja de color distinto representa una arquitectura de la red, descrita anteriormente.
- El gráfico de caja representa la distribución de desempeño para cada lema como se describió para la Figura 4.1.

En este caso puedo ver que el número de neuronas no afecta el resultado sino el número de capas. La arquitectura de tres capas muestra los mejores resultados evitando una alta tendencia al overfit. Hubo otros experimentos añadiendo más capas, pero la mejora de los resultados no fue mucho mayor que con una arquitectura de tres capas y, a medida que el número de hiperparámetros aumentaba, las redes eran más propensas a sobredimensionar los datos memorizándolos. Además, el tiempo de formación aumentó considerablemente para arquitecturas más profundas.

4.4.2.2 Comparación de clasificadores

La figura 4.3 muestra la comparación de los clasificadores descritos en la Sección 4.3.3, con la adición de un clasificador de línea de base que asigna el sentido más frecuente a cada instancia.

Dado que se estableció que la representación supervisada a utilizar en el resto de los experimentos de esta tesis está utilizando el truco del hash, la comparación aquí es sólo para dicha representación. La figura también tiene un gráfico de cajas y bigotes con una estructura similar a la mostrada anteriormente:

- Cada columna representa un corpus: SenSem y SemEval.
- Cada grupo de gráficos de caja en cada gráfico representa una métrica: Promedio macro y ponderado del F1-score.
- Cada gráfico de caja de color distinto representa un clasificador: baseline, árbol de decisión, perceptrón multicapa, Naive Bayes, regresión logística y SVM.

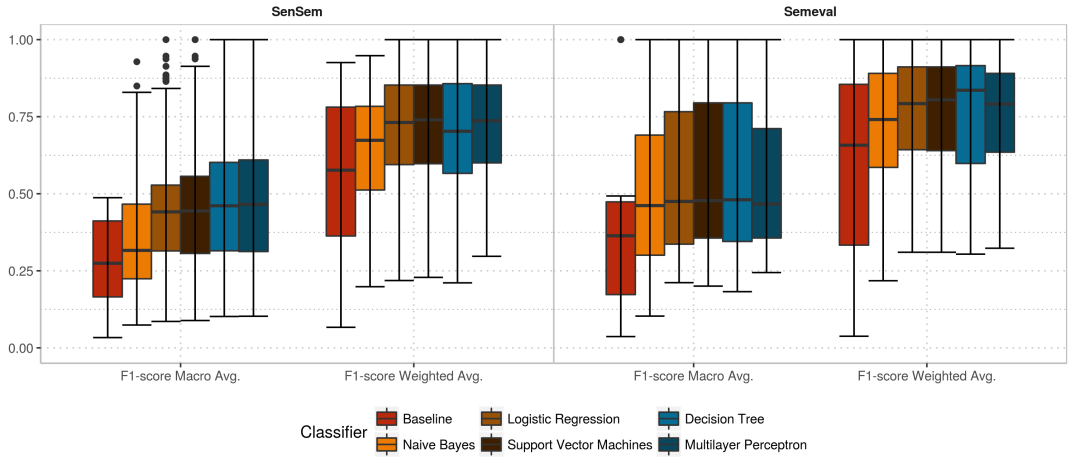


FIGURA 4.3: Comparación de clasificadores

- El gráfico de caja representa la distribución de desempeño para cada lema como se describió para la Figura 4.1.

Lo primero que hay que eliminar de la trama es que todos los clasificadores superan al clasificador de línea de base descrito anteriormente. En particular, la ingenuidad de Bayes es la que muestra los peores resultados entre los otros clasificadores, muy cerca del desempeño del clasificador de línea base, claramente sesgado por el sentido más frecuente. Por otro lado, tanto el clasificador de árbol de decisión como el clasificador de perceptrón multicapa (con la arquitectura definida en el apartado anterior) muestran los mejores rendimientos para el corpus SenSem si nos centramos en las clases minoritarias. Sin embargo, en un promedio ponderado el árbol de decisión tiene un peor desempeño que SVM, LR o MLP.

Para SemEval hay menos disparidad entre el desempeño de los clasificadores (siendo todavía ingenuo Bayes el de peor desempeño, además de la línea de base), siendo la mediana similar para todos ellos, y siendo el árbol de decisión el que tiene el mejor desempeño en un promedio ponderado.

Viendo que los árboles de decisión y las percepciones multicapa son los que tienen mejores resultados, hay una fuerte indicación de que el problema de desambiguación de sentidos verbales no es lineal.

4.4.2.3 Significancia

Los resultados anteriores son lo suficientemente buenos como para afirmar que un perceptrón multicapa es un buen enfoque para modelar el problema de desambiguación de sentidos verbales. Sin embargo, para ver si la diferencia de rendimiento es significativa o no, debería probarla usando Metric 2 y viendo el coeficiente kappa de Cohen entre los diferentes clasificadores.

La figura 4.4 muestra con un mapa térmico la media del coeficiente kappa del Cohen entre los resultados de clasificación de cada clasificador por lema sobre el corpus de prueba. Cuanto más alta es la kappa, más similar es la clasificación, por lo tanto,

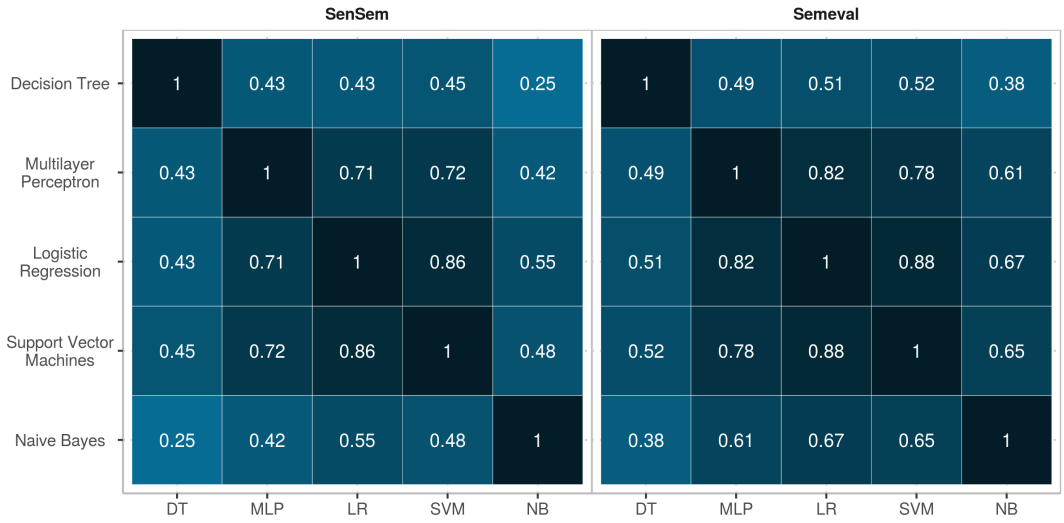


FIGURA 4.4: Coeficiente kappa de Cohen entre clasificadores

menos significativa es la diferencia entre los clasificadores. El color de la carta térmica define el valor del kappa interclasificador entre el clasificador de la fila y el clasificador de la columna. El mapa térmico es simétrico.

Los clasificadores de regresión logística y SVM son los que están más de acuerdo, probablemente porque ambos son clasificadores lineales. Por lo tanto, la diferencia de rendimiento entre ellos no es realmente significativa. El perceptrón multicapa es el más parecido a estos dos. Los árboles de decisión y los ingenuos Bayes son los que muestran menos acuerdo en comparación con los otros clasificadores. Es probable que el ingenuo Bayes tenga un bajo acuerdo, ya que es uno de los clasificadores con peor desempeño. Aunque no es trivial establecer un umbral para el cual la estadística kappa es buena para denotar significación estadística, puedo inferir de este gráfico que el aprendizaje del árbol de decisión está teniendo claramente un buen desempeño con diferencias estadísticamente significativas con otros clasificadores que también están funcionando bien, lo que hace que sea más interesante continuar explorando en el trabajo futuro.

4.4.3 Hipótesis 1.1

Una vez seleccionado el modelo base con el que trabajar para los siguientes experimentos, puedo empezar a probar las hipótesis establecidas en Sección 4.1.

Comienzo mirando los resultados para probar Hipótesis 1.1. Recordemos la hipótesis de que los conjuntos de datos de entrenamiento más grandes ayudan a mejorar el rendimiento. Para comprobar la validez de esto sigo los pasos del experimento 4.2. Utilizo el perceptrón multicapa con tres capas que he seleccionado en los párrafos anteriores.

La figura 4.5 muestra el rendimiento en el corpus de pruebas para diferentes tamaños del corpus de entrenamiento (como porcentaje del corpus de entrenamiento completo). La figura muestra un gráfico de cajas y bigotes siguiendo una estructura similar a la mostrada anteriormente:

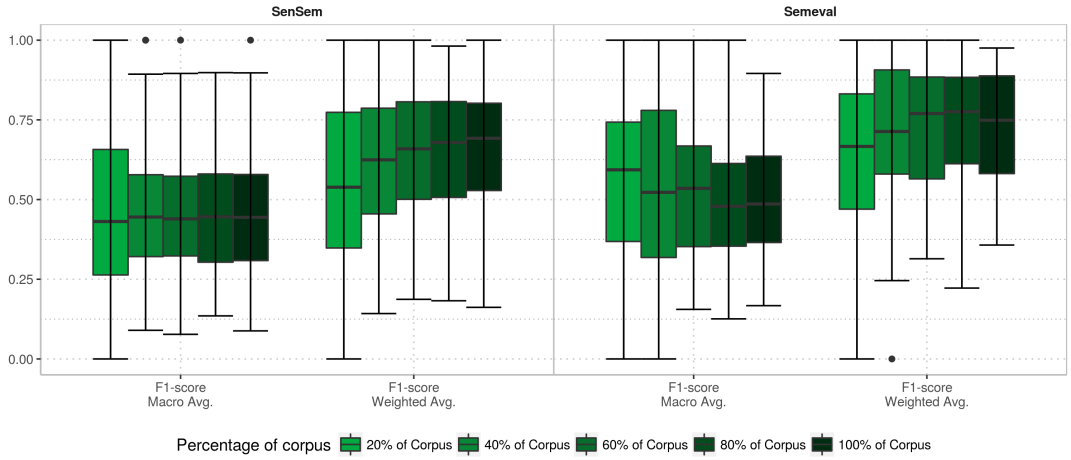


FIGURA 4.5: Desempeño por lema para el corpus de evaluación para distintos tamaños del corpus de entrenamiento

- Cada columna representa un corpus: SenSem y SemEval.
- Cada grupo de gráficos de caja en cada gráfico representa una métrica: Promedio macro y ponderado del F1-score.
- Cada gráfico de caja de tonalidad distinta representa el desempeño para distintos tamaños del conjunto de entrenamiento (como porcentaje del total de datos de entrenamiento).
- El gráfico de caja representa la distribución de desempeño para cada lema como se describió para la Figura 4.1.

Lo primero que hay que señalar de esta figura es cómo el aumento del número de ejemplos no mejora necesariamente el rendimiento del clasificador. Esto se debe a que en el proceso de ampliación progresiva se agregan los ejemplos de corpus con el objetivo de que cada clase esté representada. Cuando el corpus es muy pequeño, las clases minoritarias están comparativamente bien representadas. Pero a medida que aumenta el número de ejemplos, también aumenta el desequilibrio entre clases. Esto resulta en ninguna ganancia de rendimiento (SenSem) o incluso degradación del rendimiento (SemEval) para las clases minoritarias, como se puede ver bien en la métrica de promedio macro. El promedio ponderado mejora con el número de ejemplos en el caso de SenSem, que es un corpus muy pequeño, pero no tanto en el caso de SemEval, que es mucho mayor. Por lo tanto, aumentar el número de ejemplos parece bueno para mejorar un corpus pequeño, pero no necesariamente un corpus más grande. La razón principal de ello parece ser el problema del desequilibrio de clases, que se agudiza cada vez más a medida que aumenta el número de ejemplos.

Sin embargo, esto es difícil de ver claramente, ya que los gráficos de caja son útiles para dar una idea general del rendimiento de muchos modelos diferentes, pero oscurecen el rendimiento particular de un modelo específico. Como tengo un modelo

por lema, necesitaría examinar más de cerca 200 modelos diferentes para ver cómo está funcionando para cada lema, algo fuera del alcance de esta tesis.

Sin embargo, hay un patrón claro de cómo un modelo mejora el rendimiento con más datos de entrenamiento, especialmente en los casos de modelos con un rendimiento extremadamente bajo: hay modelos que inicialmente tienen un F1-score de 0 para los modelos entrenados con menos datos, algo que no ocurre con más datos añadidos.

4.4.4 Hipótesis 1.2

Para comprobar Hipótesis 1.2 dije que lo haría con la ayuda del experimento 4.3. La hipótesis establece que el tamaño del conjunto de datos no sólo afecta el desempeño (medido por métrica 1), sino también la tendencia al sobreajuste (medido por métrica 3) de un modelo.

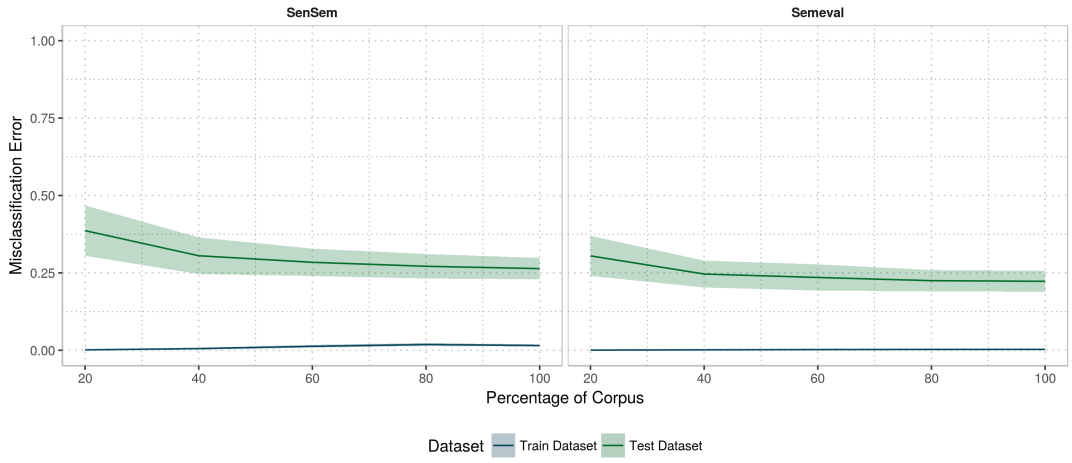


FIGURA 4.6: Curva de aprendizaje para distintos tamaños del corpus de entrenamiento

La figura 4.6 muestra la curva de aprendizaje para diferentes tamaños de los datos de entrenamiento. La estructura de la curva de aprendizaje es la siguiente:

- Cada columna representa un corpus: SenSem y SemEval.
- La coordenada x muestra el tamaño de los datos de entrenamiento, como un porcentaje del total de datos de entrenamiento disponibles, comenzando desde el 20 % del corpus (el corpus fue particionado en 5 partes de acuerdo a lo establecido en el experimento 4.3).
- La coordenada y muestra el error de clasificación de un modelo.
- Hay dos colores que representan los conjuntos de datos: entrenamiento y evaluación.
- Las líneas sólidas más oscuras representan la media del error de clasificación a través de las diferentes particiones de los conjuntos de datos a lo largo de todos los modelos.

- El área sombreada, que tiene un color más claro, representa el error estándar de la media para el error de clasificación.

Recordar que el experimento 4.3 primero dividió los corpus en partes de tamaño uniforme y gradualmente agregó esas nuevas partes a la formación y evaluación del modelo. Entrenó un modelo en una parte y lo evaluó en la otra, lo aguantó, la parte y registró los resultados. Repitió el algoritmo un número de veces sobre diversas maneras de dividir los datos para ver cómo se comporta el mismo modelo en diversos datasets. La figura 4.6 muestra la media y el error estándar de la media sobre el error de clasificación erróneo que cada modelo tiene en los conjuntos de datos de entrenamiento y pruebas.

Los datos de la prueba tienen una variación visible del error de clasificación errónea, visto en el área sombreada amplia, mientras que los datos de entrenamiento no tienen casi ningún error de clasificación errónea, ya que el área sombreada es indistinguible de la línea que representa la media del error de clasificación errónea.

En el gráfico hay dos indicadores de un error debido a la varianza:

- Un área sombreada más amplia: significa que el error de clasificación de diferentes modelos tiene una alta varianza, lo que hace que los modelos más inconsistente sobre diferentes conjuntos de datos.

- Una media más alta del error de clasificación de la unidad de prueba en comparación con el error medio del conjunto de entrenamiento, significa que el modelo está sobredimensionando a los datos de entrenamiento.

Como puedo ver claramente en la figura 4.6, esta variación es menor cuanto más datos de entrenamiento tengo. Esta es una fuerte indicación de que la Hipótesis 1.2 es cierta, ya que hay menos sobreajustes en los diferentes modelos mientras más datos se usen para el entrenamiento.

4.4.5 Hipótesis 1.3

Para comprobar Hipótesis 1.3 Lo haría con la ayuda de Experiment 4.3. Recuerdo que la Hipótesis decía que el número de clases afecta la tendencia al sobreajuste (medido por la métrica 3). Para mostrar esto, trazo la curva de entrenamiento midiendo la media de los lemas que tienen un número diferente de etiquetas totales.

La figura 4.7 muestra la curva de aprendizaje para diferentes números de clases. La estructura del gráfico es similar a la de la Figura 4.6 con algunos cambios menores para mostrar lo que dice la Hipótesis:

- Cada fila del gráfico representa un corpus: SenSem y SemEval.

- Las columnas del gráfico representan la cantidad de clases del modelo: 2, 3, 4, 5, 6, 7, 8, y 10 clases.

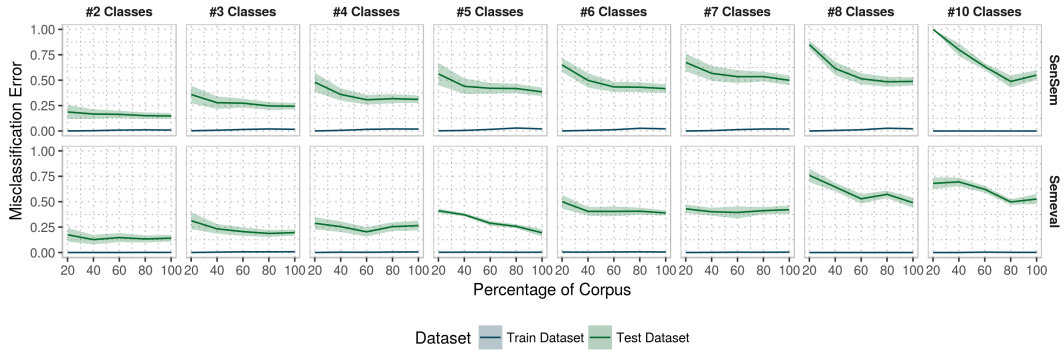


FIGURA 4.7: Curvas de aprendizaje para distintos números de clases

- La coordenada x es el tamaño del conjunto de entrenamiento como se explica en la sección 4.4.4.
- La coordenada y muestra el error de clasificación.
- Los colores, líneas y áreas sombreadas representan lo mismo que se describe en la sección 4.4.4.

Los modelos son uno para cada lema, por lo que la Figura 4.7 muestra en cada columna la media de todos los modelos que tienen ese número de clases (sentidos). El patrón de la Figura es claro, ya que la media del error debido a la varianza es mayor cuantas más clases tenga el modelo. El error debido a la alta varianza es mayor en los modelos con más clases como consecuencia de la distribución de las clases, que es Zipfian. Como la tendencia a clasificar todos los datos como parte de la clase más frecuente cuando no hay suficientes datos de las otras clases, el error de clasificación de los datos de la prueba es mayor que el error de formación. Estos resultados dan evidencias para no rechazar la Hipótesis 1.3.

4.4.6 Hipótesis 1.4

Para comprobar Hipótesis 1.4 hago uso del experimento 4.3. Esta Hipótesis establece que los modelos lineales tienen menos tendencia a sobredimensionarse que los modelos no lineales. Esta Hipótesis busca verificar si la no linealidad de un modelo afecta su tendencia a sobredimensionar los datos.

La figura 4.8 muestra la curva de aprendizaje para diferentes tipos de clasificadores (los que se presentan en la Sección 4.3.3). La figura utiliza una estructura similar a la de la figura 4.7, pero cambia lo que representan las columnas:

- Las columnas del gráfico representan la cantidad de clases del modelo: 2, 3, 4, 5, 6, 7, 8, y 10 clases.
- La coordenada x es el tamaño del conjunto de entrenamiento como se explica en la sección 4.4.4.

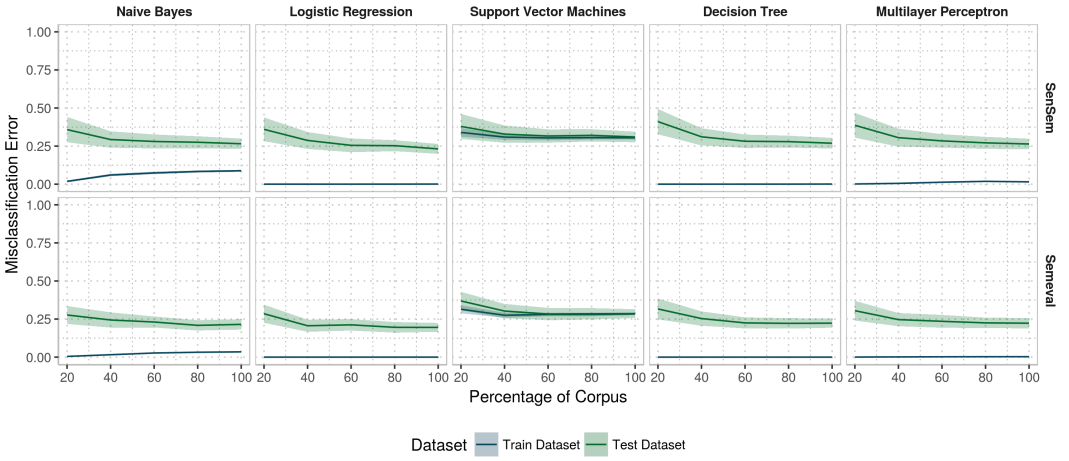


FIGURA 4.8: Curva de aprendizaje para distintos tamaños del corpus de entrenamiento por clasificador

- La coordenada y muestra el error de clasificación.
- Los colores, líneas y áreas sombreadas representan lo mosmo que se describe en la sección 4.4.4.
- Cada fila del gráfico representa un corpus: SenSem y SemEval.
- Las columnas del gráfico representan los clasificadores: Naive Bayes, Regresión Logística, Support Vector Machines, Árboles de Decisión y Perceptrón multicapa. Nótese que los clasificadores lineales son los primeros tres de izquierda a derecha y los clasificadores no lineales son los últimos dos de izquierda a derecha.
- La coordenada x es el tamaño del conjunto de entrenamiento como se explica en la sección 4.4.4.
- La coordenada y muestra el error de clasificación.
- Los colores, líneas y áreas sombreadas representan lo mosmo que se describe en la sección 4.4.4.

Lo primero que se destaca en el gráfico son los resultados de Ingenua Bayes y Máquinas Vectoriales de Soporte. En ambos casos, el error de entrenamiento no es tan cercano a 0 como en los otros clasificadores.

En las Bayes ingenuas, el error de los datos de formación aumenta con el número de instancias de formación. Recordad que en la Sección 4.4.2.2, vi que Naive Bayes era el clasificador con el menor rendimiento y me di cuenta de que esto se debía a que tenía sesgos hacia la clase más frecuente. La figura 4.8 da aún más evidencia para apoyar esa afirmación. El ingenuo Bayes entonces muestra menos diferencia entre el error de entrenamiento y el error de prueba, ya que sacrifica el rendimiento del modelo por una mayor generalidad del mismo.

Support Vector Machines muestra un área sombreada más amplia del error de clasificación errónea del conjunto de datos de formación, lo que significa una mayor varianza del rendimiento en los datos de formación. SVM con un núcleo lineal separa los datos con un hiperplano de márgenes máximos, al igual que Bayes ingenuo, sacrifica el buen rendimiento de los datos de entrenamiento para un mejor rendimiento sobre los datos generales.

Los otros tres métodos muestran resultados muy similares, con árboles de decisión ligeramente superiores al resto y regresión logística ligeramente inferior. Sin embargo, de los resultados visuales veo que la importancia de esta diferencia en el rendimiento no es significativa.

Los resultados mostrados apoyan la Hipótesis 1.4, ya que los clasificadores lineales muestran menos tendencia a sobreajustarse, aunque a expensas de un mayor error debido al sesgo.

4.5 Conclusiones

Este capítulo presenta el problema de desambiguación de sentidos verbales utilizar métodos puramente supervisados. La hipótesis que quiero probar es que el tamaño del equipo de entrenamiento afecta al rendimiento de un modelo. Esta hipótesis inicial se dividió en subhipótesis más específicas, que la experimentación y el análisis de los resultados intentaron aceptar o rechazar.

Diseñé algunos experimentos para ver si había una diferencia significativa entre los diferentes modelos. Un primer paso fue descartar que las técnicas para reducir la dimensionalidad de las representaciones afectaran el rendimiento de los modelos. Esto se demostró con los resultados de la Sección 4.4.1.

Para el clasificador de la red neural, necesitaba establecer la arquitectura de la red. Los resultados reportaron que una arquitectura con más capas mejoró el desempeño del modelo. Este no fue el caso para el número de neuronas por capa. La sección 4.4.2.1 muestra estos resultados.

Finalmente, comparé el rendimiento del clasificador de la red neuronal con los otros clasificadores definidos en la Sección 4.3.3. La sección 4.4.2.2 muestra que la red neuronal tuvo uno de los mejores resultados para el corpus SenSem. Sin embargo, el coeficiente de Kappa no reportó significación en esta mejora sobre los otros clasificadores. En cualquier caso, el objetivo era comprobar si la red neural tenía un rendimiento superior al de otros clasificadores, lo que no era el caso. La red neural, como mostraré, es el clasificador que finalmente elegí porque su desempeño no era peor que los otros y es comparable a los clasificadores explorados en capítulos futuros.

Los experimentos realizados para aceptar o rechazar la Hipótesis 1.1 de que un mayor conjunto de datos de entrenamiento mejora el rendimiento son observables en la Sección 4.4.3. Los resultados reportados dan fuertes indicaciones sobre la validez de la Hipótesis, ya que hay una mejora visible en los resultados generales a medida que

aumenta el número de ejemplos de entrenamiento, por lo que la Hipótesis no puede ser rechazada.

Hipótesis 1.2 no puede ser rechazado ya que los resultados de la Sección 4.4.4 dan una fuerte evidencia que lo respalda. De hecho, la tendencia al sobreajuste y el error debido a la varianza disminuyen mientras más datos de entrenamiento tenga el modelo.

De la misma manera la Hipótesis 1.3 no puede ser rechazada debido a la evidencia en los resultados de la Sección 4.4.5. El número de clases (sentidos) influye en el ajuste del modelo. Cuantas más clases haya, más difícil le resultará al modelo no adaptarse.

Por último, los experimentos para probar Hipótesis 1.4, que espera que los modelos lineales tengan menos tendencia a sobredimensionarse que los modelos no lineales, no arrojan resultados concluyentes, aunque en una observación superficial los resultados pueden interpretarse a favor de no rechazar la Hipótesis. Dos de los clasificadores lineales tienen el error de entrenamiento más cercano al error de validación. Los métodos sacrifican la precisión en sus datos de entrenamiento para mejorar la generalización. El problema, sin embargo, es que el error de clasificación errónea del conjunto de pruebas no tiene un aspecto significativamente mejor que el de los otros clasificadores.

El objetivo de este capítulo era sentar las bases sobre las que se compararán los capítulos siguientes y tratar de superar sus deficiencias. En particular, los modelos puramente supervisados presentan desafíos en dos aspectos principales: la tendencia a adaptarse cuando se entrena un modelo con un conjunto de datos pequeño y la cobertura que estos modelos pueden tener sobre ejemplos nuevos e invisibles. Los enfoques semi-supervisados que exploraré en los capítulos siguientes tratan de resolver estos desafíos desde diferentes ángulos.

Una de las causas latentes del sobreequipamiento en modelos puramente supervisados viene dada por la propia naturaleza de tales modelos. Generan una representación de los ejemplos de formación basada en características obtenidas a partir de los mismos datos anotados que los clasificadores intentan representar. La forma en que pretendo atacar este fallo es mediante el uso de características que generalizan mejor, no vinculadas a un conjunto de datos etiquetado en particular, sino obtenidas a partir de una muestra de lenguaje más general. Esto se explora en el capítulo 5. Los métodos no supervisados utilizan representaciones más suaves de los datos etiquetados con embeddings de palabras.

Otro de los problemas latentes en los modelos puramente supervisados ocurre en la cobertura de tales modelos. Esto es difícil de medir y cuantificar, ya que para ello se necesita un número mucho mayor de ejemplos anotados.

El problema de cobertura sólo puede medirse a través de la métrica del silencio. El silencio captura aquellos ejemplos que pertenecen a una de las clases de datos, pero que no están disponibles en el corpus anotado, por lo que los modelos no pueden aprender de estos datos. Los datos comentados sólo pueden considerar un universo limitado de características, dejando fuera otras características de las etiquetas que pueden mejorar el modelo sobre nuevos candidatos a clasificar.

Otros métodos semi-supervisados, estudiados con más detalle en capítulos posteriores, estudian formas de superar esta deficiencia de los enfoques puramente supervisados, por ejemplo, mediante la anotación de nuevos datos (automática o manualmente) que contribuyen a que los modelos supervisados dispongan de más información latente para mejorar el rendimiento con respecto a los nuevos datos.

Parte II

Aprendizaje semisupervisado disjunto

Capítulo 5

Vectores de palabras (word embeddings)

5.1 Estructura

En este capítulo exploro el uso de embeddings de palabras no supervisadas para ayudar a la tarea de desambiguación de sentidos verbales del español. El trabajo sobre desambiguación de sentidos verbales del inglés del capítulo 4 sirvió como punto de comparación con el trabajo realizado para el español, sin embargo para simplificar el estudio de las técnicas semi-supervisadas me estoy centrando a partir de ahora en el español solamente.

El capítulo anterior nos mostró dos de los principales problemas con los enfoques puramente supervisados con pequeños datos etiquetados: el exceso de datos y la poca cobertura. El modelo aprende a adaptarse a las funcionalidades disponibles, y aunque consigue hacerlo bien, se generaliza mal, por lo que acaba sobredimensionándose. La cobertura de los datos, por otro lado, depende de la cantidad de datos etiquetados disponibles, que se estableció que era pequeña.

El capítulo anterior también mostró que el tamaño del conjunto de datos de entrenamiento afecta el rendimiento final de un modelo. Con base en estos resultados es posible argumentar que la razón detrás de este fenómeno es que mientras más datos haya, más características pueden ser incluidas en el modelo. Por lo tanto, los nuevos ejemplos tienen una mejor representación porque es más probable que las características que los representan ya estén incluidas en el conjunto de datos de formación.

Para abordar este problema de cobertura, quiero ampliar los datos etiquetados con nuevos datos de fuentes no etiquetadas, al tiempo que mantengo el esfuerzo de la anotación humana al mínimo. Sin embargo, para hacerlo, primero necesito encontrar una manera para que los datos disponibles disminuyan la tendencia al sobreajuste, ya que necesito un modelo que se generalice bien a nuevos ejemplos para que éstos se sumen a la cobertura del modelo. Exploraré el impacto de agregar más ejemplos con bajo costo de anotación en el siguiente capítulo. En este capítulo me centraré en la reducción de la tendencia a la sobreadaptación de los enfoques supervisados clásicos.

La ingeniería de atributos plantea un problema para la generalización porque se toman literalmente del conjunto de datos etiquetado, y por lo tanto no juegan bien con el cambio de dominio. Esto es importante porque los datos etiquetados están

contenidos en un dominio en particular, el dominio periodístico, porque están tomados de un periódico. Sin embargo, el modelo también debería adaptarse a otros ámbitos.

Por último, hay otros dos problemas latentes de las representaciones de características artesanales descritas en el capítulo anterior, que se suman a la complejidad de cualquier problema que uno quisiera representar: la alta dimensionalidad y la escasez de las representaciones. La dimensionalidad, como vimos en la Sección 4.3.2 puede ser manejada a través de alguna técnica de reducción de dimensionalidad (por ejemplo, el *hashing trick*). No obstante, la escasez sigue siendo un problema, ya que la mayoría de las características diseñadas se producen en una sola instancia, por lo que los vectores de representación tienen poca información. Esta alta dimensionalidad y la escasez de los datos se suman al costo computacional de resolver el problema con los clasificadores.

Para tratar estos problemas de representaciones dispersas, aplico métodos de representación de palabras sin supervisión, a saber, Word2Vec [Mikolov et al., 2013a]. Estas técnicas han ganado recientemente atención en la comunidad de procesamiento del lenguaje natural, especialmente con el resurgimiento de las redes neuronales y el aprendizaje profundo. Se denominan embeddings de palabras y representan palabras con menos dimensiones y vectores densos.

Este capítulo toma el dominio y los experimentos del capítulo anterior, es decir, la desambiguación del sentido verbal para el español, y desarrolla nuevos experimentos sobre esta base, con la adición de embeddings de palabras como características para los modelos.

Este capítulo evalúa las siguientes hipótesis:

Hipótesis II *Las representaciones no supervisadas mejoran los modelos supervisados al evitar el sobreequipamiento causado por las características tomadas del mismo conjunto de datos en el que se entrena el modelo.*

Haré esto probando las siguientes subhipótesis:

Subhipótesis 2.1 *El rendimiento de una representación no supervisada depende del dominio del conjunto de datos sin etiquetar en el que están formados.*

Subhipótesis 2.2 *El uso de embeddings de palabras produce menos sobreajustes sobre modelos supervisados que las características hechas a mano.*

Estas hipótesis se probarán utilizando el siguiente esquema:

- El experimento 5.1 reporta el desempeño de diferentes representaciones, las cuales pueden ser supervisadas o no. El rendimiento se mide por el promedio macro y ponderado del F1-score (métrica 1). Los resultados mostrados en la Sección 5.4.1 sirven para aceptar Hipótesis 1.1, que el dominio donde se obtienen los embeddings de palabras afecta al rendimiento final del modelo.

- El experimento 5.2 reporta la varianza de diferentes modelos entrenados sobre diferentes subconjuntos de los datos de entrenamiento. Esta varianza se mide

por la métrica 3 que refleja la tendencia de un modelo a adaptarse, con la ayuda de la curva de aprendizaje. A partir de este experimento y métrica, usando diferentes visualizaciones de acuerdo a la tarea, muestro que el clasificador es menos propenso a sobredimensionar usando embeddings de palabras en lugar de características hechas a mano como se indica en Hipótesis 2.2.

En la sección 5.2 Recapitulo el trabajo anterior sobre representaciones no supervisadas y embeddings de palabras. Se hace especial hincapié en el algoritmo de word2vec. También señalo el trabajo realizado en desambiguación de sentidos usando word embeddings.

En la sección 5.3. Explico todos los puntos relevantes relacionados con lo que se utiliza para llevar a cabo la experimentación de este capítulo. Primero, en la sección 5.3.1 Presento los recursos con los que trabajo. Se hace una recapitulación rápida del corpus SenSem, seguido de los corpus no etiquetados usados para entrenar los embeddings de palabras: SBWCE, cuerpos periodísticos y cuerpos reglamentarios. En la Sección 5.3.2 Describo los dos tipos de características utilizadas para la experimentación: características supervisadas del capítulo anterior y embeddings de palabras formadas a partir de los corpus no etiquetados. La sección 5.3.4 lista los experimentos. La sección 5.3.5 lista el conjunto de métricas que uso para medir los experimentos.

La sección 5.4 informa de los resultados de los experimentos y los analiza para aceptar o rechazar las hipótesis expuestas en el capítulo.

Finalmente la sección 5.5 extrae las conclusiones de este capítulo, recapitulando las Hipótesis y las implicaciones de aceptarlas o rechazarlas según las evidencias recogidas en los resultados. Establece las deficiencias de los métodos explorados en este capítulo y lo que quiero lograr en el siguiente. Termina esbozando el trabajo futuro.

5.2 Trabajo relevante

El método que más he explorado para este capítulo es el propuesto por Mikolov et al. [Mikolov et al., 2013b]: el modelo de programa de omisión con muestreo negativo. Consiste en un modelo de lenguaje que maximiza la probabilidad de que un par de palabras (es decir, un programa de salto) ocurran en algún texto en lenguaje natural, mientras que minimiza la probabilidad de que un par de palabras al azar ocurran al mismo tiempo. Para más detalles sobre este proceso, consulte el capítulo 3, sección 3.2.2.

Existen, sin embargo, otras aproximaciones a las representaciones de palabras no supervisadas, que son muy bien exploradas en el trabajo de Turian et al. [Turian et al., 2010]. En este trabajo, los autores mejoran la precisión de los diferentes sistemas de procesamiento de lenguaje natural existentes mediante el uso de representaciones de palabras sin supervisión como características.

En sistemas de desambiguación de sentidos, como se explica en el capítulo 3, sección 3.2.2, Taghipour y Ng [Taghipour and Ng, 2015] muestran que el uso de word embeddings mejora de forma consistente la precisión de las tareas de muestra léxica y

de todas las palabras de SemEval, y también en una tarea de muestra léxica específica del dominio. Rothe y Schütze [Rothe and Schütze, 2015] presentaron un sistema para aprender embeddings para synsets/lexemas. Finalmente, hay un buen trabajo sobre word embeddings como características para desambiguación de sentidos por Iacobacci et al. [Iacobacci et al., 2016].

Para desambiguación de sentidos del español, que yo sepa, hay poco o nada de trabajo hecho. Sólo pude encontrar la mía sobre la cual esta tesis se expande en [Cardellino and Alonso i Alemany, 2017].

5.3 Metodología

Este capítulo explora cómo los embeddings de palabras ayudan a un clasificador de aprendizaje supervisado (p. ej., el perceptrón multicapa) reemplazando los elementos artesanales supervisados. A partir de este capítulo, hay una observación importante: las frases *word embeddings* y *vectores de palabras* pueden utilizarse indistintamente.

5.3.1 Recursos

Hay dos tipos de recursos necesarios para la experimentación realizada en este capítulo: los corpus etiquetados para capacitar al clasificador supervisado, y los corpus no etiquetados para capacitar a las representaciones no supervisadas (word embeddings).

A diferencia del capítulo anterior, la experimentación se realiza únicamente con los datos españoles, ya que esta es mi principal preocupación de estudio y el capítulo anterior proporcionó suficiente información del conjunto de datos en inglés para afirmar la validez de los modelos supervisados.

5.3.1.1 SenSem

SenSem sirve como el recurso principal para entrenar el modelo de desambiguación de sentidos verbales como los datos etiquetados. Se describe en detalle en la Sección 4.3.1.1. Por favor refiérase a la sección para más información.

5.3.1.2 SBWCE

El principal recurso de donde provienen los embeddings de palabras utilizadas en los experimentos de este capítulo es *Spanish Billion Word Corpus and Embeddings* (SBWCE) [Cardellino, 2016], que es una compilación de más de 1.400 millones de palabras en bruto de la lengua española, extraídas de diferentes fuentes disponibles en Internet, la mayoría de ellas procedentes de corpus utilizados para tareas de traducción automática estadística, así como de corpus de la fundación Wikimedia, lo que lo convierte en un corpus de dominio general.

El corpus tiene más de 45 millones de frases con más de 3 millones de fichas únicas. También es preprocesado para eliminar todos los símbolos de puntuación y reemplazar todos los dígitos con la etiqueta "DIGITO".

5.3.1.3 Corpus periodístico

SenSem ofrece un corpus comentado basado en dos periódicos de la región de Cataluña en España: "El Periódico" y "La Vanguardia". Esto hace que el recurso esté fuertemente basado en sentidos que tienen más que ver con el dominio periodístico. Por lo tanto, para comprobar Hipótesis 2.1 necesitábamos entrenar algunas representaciones no supervisadas de un corpus de dominio periodístico.

Extraje los documentos procedentes de fuentes periodísticas disponibles en la SBWCE y en otros periódicos disponibles en línea. En comparación con el corpus de la SBWCE, el corpus que pudimos reunir para esta tarea era mucho más pequeño, con casi 71 millones de palabras disponibles, que se convirtieron en 70 millones después de filtrar todas las palabras con menos de 3 ocurrencias. Había una lista final de aproximadamente 240 mil palabras únicas para generar los embeddings de palabra cuya dimensión era 50.

5.3.1.4 Corpus normativo

Finalmente, para probar la hipótesis 2.2 necesito un corpus de dominio específico, pero que no sea del mismo dominio en el que se basa SenSem. Originalmente mi intención era utilizar un corpus basado en documentación de software libre, ya que el aspecto técnico de tales corpus sería un buen contraste con el de un corpus de dominio periodístico. Sin embargo, el corpus disponible que pude encontrar en línea era demasiado pequeño para generar buenas representaciones de embeddings de palabras.

Además de los textos de dominio periodístico, el otro dominio disponible con suficiente información disponible era un corpus normativo, basado en corpus como el Parlamento Europeo, las Naciones Unidas, etc. donde el texto es más formal ya que trata de leyes, normativas, etc.

La cantidad de datos disponibles es aproximadamente la misma que en el Corpus Periodístico, con 72 millones de fichas, pero con un vocabulario de 100 mil palabras únicas y embeddings de la dimensión 50.

5.3.2 Atributos

5.3.2.1 Supervisados

Para los experimentos puramente supervisados, los datos se representan usando todas las características descritas en la Sección 4.3.2 y el *hashing trick* presentado en la Sección 4.3.2.2.

5.3.2.2 Word embeddings

Para los experimentos puramente supervisados, los datos se representan usando todas las características. Para las representaciones de embeddings de palabras, usé el algoritmo Word2Vec [Mikolov et al., 2013a] para entrenar los embeddings de palabras de los corpus no etiquetados. El primer conjunto de embeddings de palabras,

para los experimentos de dominio general, son los pre-entrenamientos disponibles en la SBWCE.

Los word embeddings pre-entrenadas a partir de este recurso fueron creadas usando el modelo de Word2Vec *skip-gram* y la biblioteca gensim [Řehůřek and Sojka, 2010]. Filtra palabras con menos de 5 ocurrencias dejando fuera aproximadamente 1 millón de palabras únicas. La dimensión final de los vectores de palabra es 300.

La idea general de usar embeddings pre-entrenados es la disponibilidad de los mismos. En términos generales, los embeddings formadas en una gran cantidad de datos funcionan relativamente bien para tareas generales, sin embargo, queríamos ver el impacto de los embeddings de formación específicamente para los datos disponibles y qué efecto tiene esto en los resultados.

Los embeddings de la palabra son fáciles de usar. La idea es representar cada instancia (es decir, la oración con el verbo desambiguar) como una concatenación de vectores de palabras. Utilizo la ficha del verbo para desambiguar como vector central en la concatenación, y elijo una ventana simétrica de 5 fichas a cada lado de la palabra central haciendo del vector final una concatenación de 11 palabras. De esta manera, la representación final no sólo capta la semántica de las palabras a través de los embeddings, sino también a través de la posición relativa de cada palabra con respecto a los embeddings del verbo.

Si el token no está disponible en el modelo de embeddings de palabras, probamos el token con todos los caracteres en minúsculas y mayúsculas (el primer carácter en mayúsculas y el resto en minúsculas). Si ninguna versión del token está disponible usamos un vector de ceros de la misma dimensión que los embeddings de palabras.

En el caso de que la palabra central esté cerca del principio o del final de la oración, rellenamos la cantidad de palabras que quedan para completar todo el vector con ceros. Por ejemplo, el verbo se ubica como la tercera palabra desde el principio de la oración, luego para completar la ventana derecha usamos la palabra vectores para la primera y segunda ficha de la oración y rellenamos con tres vectores de valor cero antes de los vectores de dos fichas.

Después de este ajuste, el vector de entrada cuando se utiliza el corpus SBWCE es de la dimensión 3300 y los vectores para el dominio periodístico son de la dimensión 550.

5.3.3 Clasificadores

En el capítulo anterior exploré diferentes tipos de clasificadores. La conclusión fue que no había diferencias significativas entre ninguno de los clasificadores. Por lo tanto, por razones de comparación, selecciono el clasificador de perceptrón multicapa con tres capas de tamaño 500, 250 y 100. En este capítulo, los experimentos sólo se realizarán con ese clasificador. La configuración es la misma que para los experimentos de aprendizaje supervisado, con la diferencia de las características a utilizar.

5.3.4 Experimentos

Los experimentos en este capítulo siguen la estructura dada en la Sección 4.3.4. Sin embargo, en este capítulo no exploramos todas las combinaciones posibles sino que nos centramos en las que resultaron más interesantes en la Sección 4.3.4: el corpus es siempre el corpus SenSem y el clasificador es el perceptrón multicapa.

Experiment 5.1 compara representaciones supervisadas y no supervisadas. Dentro de las representaciones no supervisadas, comparo diferentes embeddings de palabras. Este experimento es la base para probar Hipótesis 2.1 que evalúa la diferencia de rendimiento del mismo clasificador entrenado usando embeddings obtenidos de diferentes dominios. Sigue la estructura de Experiment 4.1.

Experimento 5.1.

5.1a Tren de un modelo con el subconjunto de tren del corpus para cada lema del corpus.

5.1b Clasificar el corpus de prueba, para cada lema, con el modelo entrenado.

5.1c Compare los resultados pronosticados del modelo, para cada lema, con el verdadero resultados usando alguna métrica.

Experiment 5.2 sigue la estructura de Experiment 4.3, mide la varianza de diferentes modelos entrenados sobre diferentes subconjuntos de los datos de entrenamiento. Los resultados se utilizan para medir la tendencia de un modelo a sobredimensionarse. Esto es necesario para probar Hypothesis 2.2. La estructura de este experimento es la siguiente:

Experimento 5.2.

5.2a Para cada lema, dividir aleatoriamente todo el corpus en un número seleccionado de divisiones. El tamaño de las divisiones debe ser uniforme. Asegúrese de que hay una división con todas las clases completas del conjunto de datos y tómela para la iteración inicial.

5.2b Tomar el conjunto de datos inicial y dividirlo en entrenamiento y evaluación.

5.2c Entrenar un modelo con el conjunto de datos de entrenamiento obtenido en el paso anterior y almacene las predicciones sobre el tren y los conjuntos de datos de prueba obtenidos en el paso anterior.

5.2d Agregar la partición siguiente al conjunto de datos y repetir desde el paso 5.2b.

5.2e Cuando todas las particiones fueron procesadas, repetir todo el algoritmo n veces con un nuevo conjunto de particiones aleagorias.

5.3.5 Métricas

Las métricas de este capítulo son las utilizadas en el capítulo anterior para medir el rendimiento y la tendencia al sobreequipamiento.

La sección 4.3.5.1 define la métrica 1, es decir, la macro y la media ponderada de la puntuación F1. Esta métrica es útil para tratar el problema de evaluar el rendimiento cuando las clases están desequilibradas. La métrica resalta si el modelo está funcionando bien no sólo en la clase más frecuente sino también en las clases menos frecuentes.

La sección 4.3.5.3 define la métrica 3. Es útil medir la tendencia de un modelo a sobredimensionarse a medida que aumenta el tamaño del conjunto de datos. Lo hace midiendo el error debido a la varianza de un modelo entrenado en un conjunto de entrenamiento, sobre todos los demás conjuntos de entrenamiento disponibles.

5.4 Análisis de resultados

En la siguiente sección se reportan los resultados obtenidos a través de los experimentos presentados anteriormente. Como en la Sección 4.4, quiero recordar que las cifras que se muestran aquí a través de las métricas y las herramientas de visualización son vistas de los resultados. Cualquier vista puede mostrar algo a costa de oscurecer algo más.

En particular, los gráficos de cajas y bigotes muestran el comportamiento de todos los modelos (uno por cada lema) como un todo, ocultando lo que está sucediendo con el caso particular de cada lema. Para ello necesitaría un análisis más detallado, que está fuera del alcance de esta tesis.

5.4.1 Hipótesis 2.1

Los primeros resultados a analizar son los que prueban la hipótesis 2.1, que establece que el rendimiento de una representación no supervisada depende del dominio. Quiero ver si un dominio más específico afecta a los resultados finales. Para ello sigo los pasos del experimento 5.1, donde distingo las representaciones de embeddings de palabras basadas en dominio. Los embeddings de dominio general de SBWCE, embeddings de dominio específicas, compartidas con el dominio del corpus supervisado, utilizando los embeddings de dominio periodístico, y embeddings de dominio específicas fuera del dominio, no compartidas con el dominio del corpus supervisado, utilizando los embeddings de palabra de las regulaciones. Usando cada una de las representaciones de embeddings de palabras entreno un clasificador de perceptrones multicapa para ver el rendimiento de la representación seleccionada.

La figura 5.1 informa de los resultados de rendimiento en el corpus de prueba para cada lema usando estos tres dominios diferentes para entrenar la palabra embeddings. La trama es una trama de cajas y bigotes estructurada de la siguiente manera:

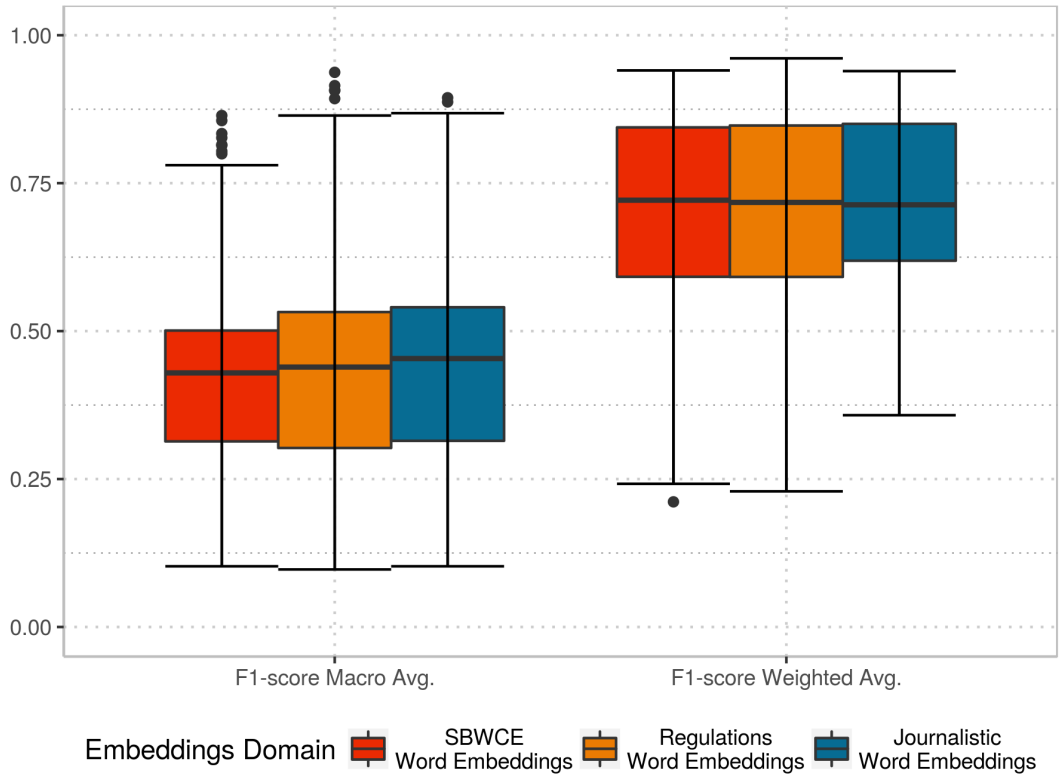


FIGURA 5.1: Desempeño por lema sobre el corpus de evaluación general, específica del dominio y específica fuera del dominio.

- Cada grupo de gráficos de caja muestra las diferentes métricas: F1-score promedio macro y promedio ponderado.
- Cada diagrama de caja de un color diferente muestra el rendimiento para diferentes dominios de formación de embeddings de palabras: dominio general (SBWCE), dominio específico fuera de la tarea (normativo), y dominio específico de la tarea (Periodístico).
- Los gráficos de cajas representan la distribución de los valores de las métricas a través de sus cuartiles. Cada valor es el rendimiento de un lema del corpus. La línea negra gruesa en el centro de una gráfica de caja representa la mediana y los bigotes al final de cada gráfica de caja representan el valor máximo y mínimo (excepto para eventuales valores atípicos representados por puntos negros fuera de la gráfica de caja).

A partir de la figura, se observa el incremento en el rendimiento de la mediana de los experimentos con embeddings de palabras periodísticas en el dominio con el macro promedio de la puntuación F1. Además de la mejor mediana, los valores máximos también son más altos para los embeddings en el dominio que para los embeddings en el dominio general. En el caso de la media ponderada de la puntuación F1 hay menos errores para los embeddings dentro del dominio que para los embeddings generales o fuera del dominio, y la diferencia para la mediana a favor del corpus general es

marginal. Recordemos que el macro promedio es bueno para medir el desempeño de las clases minoritarias. Dado que es claramente mejor para los embeddings de palabras en el dominio periodístico, muestra que los embeddings de palabras específicas en el dominio modelan mejor los sentidos menos frecuentes.

Por otro lado, no está claro si las regulaciones de embeddings fuera del dominio afectan al rendimiento de forma negativa o positiva. Muestran un mejor rendimiento en algunos de los mejores modelos y peor en algunos de los peores, como se refleja en una mayor dispersión de los bigotes. Por lo tanto, encuentro una señal clara de que necesito mirar de cerca cada modelo y ver en particular cuáles son los lemas que están mejorando y cuáles son los que están empeorando. En particular, necesito investigar los valores atípicos. El gráfico muestra que hay 4 (o tal vez 5) lemas que tienen mejor rendimiento con embeddings fuera del dominio que con embeddings dentro del dominio. En cualquier caso, las embeddings de palabras periodísticas son aún mejores en términos generales para ambas métricas, porque reducen el error.

Las razones detrás de estos resultados pueden ser muchas. Como primera explicación, el corpus normativo puede no ser tan diferente del ámbito periodístico como hubiera esperado a priori. Un mejor corpus para probar esta hipótesis podría ser uno basado en documentación que puede ser muy técnica. Sin embargo, el corpus disponible para formar tal embeddings, en esta etapa, no era lo suficientemente grande. Una línea de trabajo futuro es obtener más documentación a través del desguace de la web y formar una integración a partir de ello.

Otra posible explicación, que requiere el análisis desde el punto de vista de un experto lingüista, es que hay algunos lemas (y por lo tanto algunos modelos) que muestran un mejor desempeño en el dominio particular de las regulaciones y por lo tanto esto se refleja en los resultados. Este mejor desempeño puede deberse a una mayor frecuencia en el corpus de palabras de las regulaciones que son relevantes para distinguir los sentidos de ese lema. Esta línea de trabajo se continuará en futuras investigaciones.

Por último, SBWCE tiene algo de ruido debido al tamaño del corpus, otra línea de trabajo futuro puede ser en esa área, con embeddings capacitadas a partir de una versión más limpia de la SBWCE.

A partir de estos resultados tengo una fuerte evidencia para aceptar la declaración de Hipótesis, ya que el dominio desde el cual se entrenan las embeddings de la palabra afecta efectivamente el rendimiento final de un modelo.

5.4.2 Comparación de desempeños de representaciones supervisadas y no supervisadas

Hipótesis 2.2 establece que el uso de embeddings de palabras produce menos tendencia a sobredimensionar los modelos supervisados. En la siguiente sección se reportan los resultados de esa hipótesis para aceptarla o rechazarla. Sin embargo, hasta ahora sólo he comparado los resultados de diferentes embeddings de palabras para los modelos.

En esta sección, antes de mostrar los resultados de los experimentos de Hipótesis 2.2, quiero mostrar cómo funcionan estos nuevos modelos en comparación con los modelos seleccionados en el capítulo 4. Recordemos que el modelo final que quiero comparar es el clasificador perceptrón multicapa con tres capas.

De nuevo sigo los pasos del experimento 5.1 que entrena un modelo con todo el conjunto de datos y evalúa dicho modelo en un conjunto de datos de prueba. Para estos resultados, las representaciones elegidas son las características hechas a mano para el enfoque supervisado y los embeddings de palabras de dominio específicas (periodísticas) para el enfoque no supervisado.

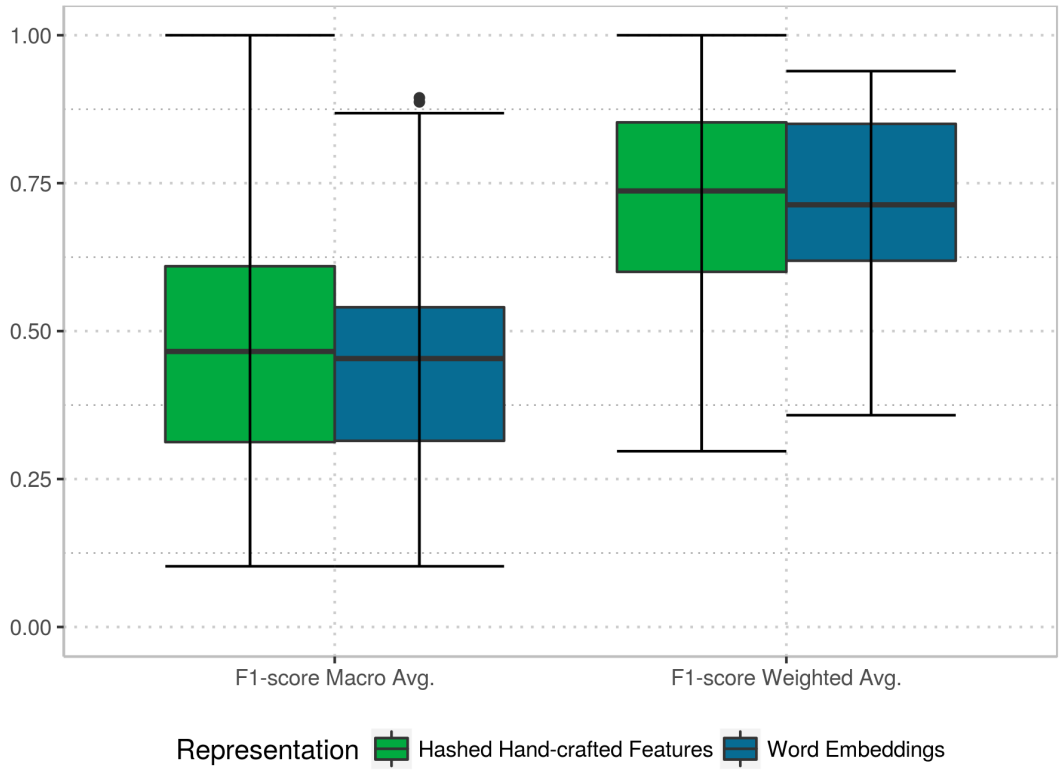


FIGURA 5.2: Desempeño por lema sobre el corpus de evaluación para ingeniería de atributos y word embeddings de dominio específico

La figura 5.2 informa de los resultados de Experiment 5.1 y los prueba en el conjunto de datos de prueba. Como en Figuras anteriores, utiliza un gráfico de cajas y bigotes para mostrar la distribución de los resultados de cada modelo:

- Cada grupo de gráficos de caja muestra las diferentes métricas: F1-score promedio macro y promedio ponderado.
- Cada diagrama de caja de un color diferente muestra el rendimiento para diferentes representaciones. Para la representación supervisada utilizo la técnica artesanal características de hash. Para las representaciones no supervisadas utilizo una dominio de la tarea (periodístico).

- El gráfico de cajas representa la distribución de valores de las métricas como se explica para la figura 5.2.

En la figura 5.2 se puede ver una clara ventaja de las características hechas a mano sobre las embeddings de palabras. El promedio macro del F1-score tiene un rendimiento especialmente mejor, lo que indica que los sentidos con un recuento bajo de ocurrencias están mejor representados.

Los word embeddings sirven como una forma de suavizar las características al reducir la dimensionalidad a una representación más baja y general. Además, la representación de la que se obtienen las embeddings tiene en cuenta menos características que las artesanales. De hecho, las características artesanales codifican la información sintáctica y del punto de venta, mientras que para obtener embeddings de palabras sólo se utilizan co-ocurrencias de palabras.

Teniendo esto en cuenta, se puede interpretar que las características artesanales representan el dominio mejor que el modelo semi-supervisado. Esto puede deberse al hecho de que las características se toman de los propios datos supervisados, a diferencia de los embeddings de palabras periodísticas, que se toman de un corpus que comparte dominio con los datos supervisados. Las características supervisadas tienen un mejor rendimiento porque pueden encajar más estrechamente con los datos, y esa es la razón por la que las características supervisadas son menos capaces de generalizar un modelo, especialmente un modelo impulsado por dominio, en datos de un dominio más general. En la siguiente sección exploraré la tendencia de estos modelos a sobredimensionarse o generalizarse.

La idoneidad de estos dos tipos de modelos para adaptarse a un corpus fuera del dominio se mostrará en el siguiente capítulo. Allí veremos que los embeddings de palabras son capaces de funcionar bien en un enfoque de autoaprendizaje en un corpus grande. Por el contrario, las características hechas a mano no se generalizan y su rendimiento se degrada rápidamente en todas las iteraciones.

5.4.3 Hipótesis 2.2

Los resultados de la sección anterior mostraron que las características supervisadas funcionan mejor que las no supervisadas para la tarea de desambiguación de sentidos verbales. Los resultados de los experimentos de Hipótesis 2.2 dan una pista sobre lo que está sucediendo debajo de estos resultados.

Estos resultados se toman de Experiment 5.2, y reportan la curva de aprendizaje de un modelo a medida que aumenta el número de ejemplos. Muestra la media y el error debido a la varianza de los conjuntos de entrenamiento y prueba en cada iteración. La comparación se realiza utilizando una representación supervisada a través de características hechas a mano y una representación no supervisada a través de embeddings de palabras, en este caso los embeddings de palabras específicas del dominio.

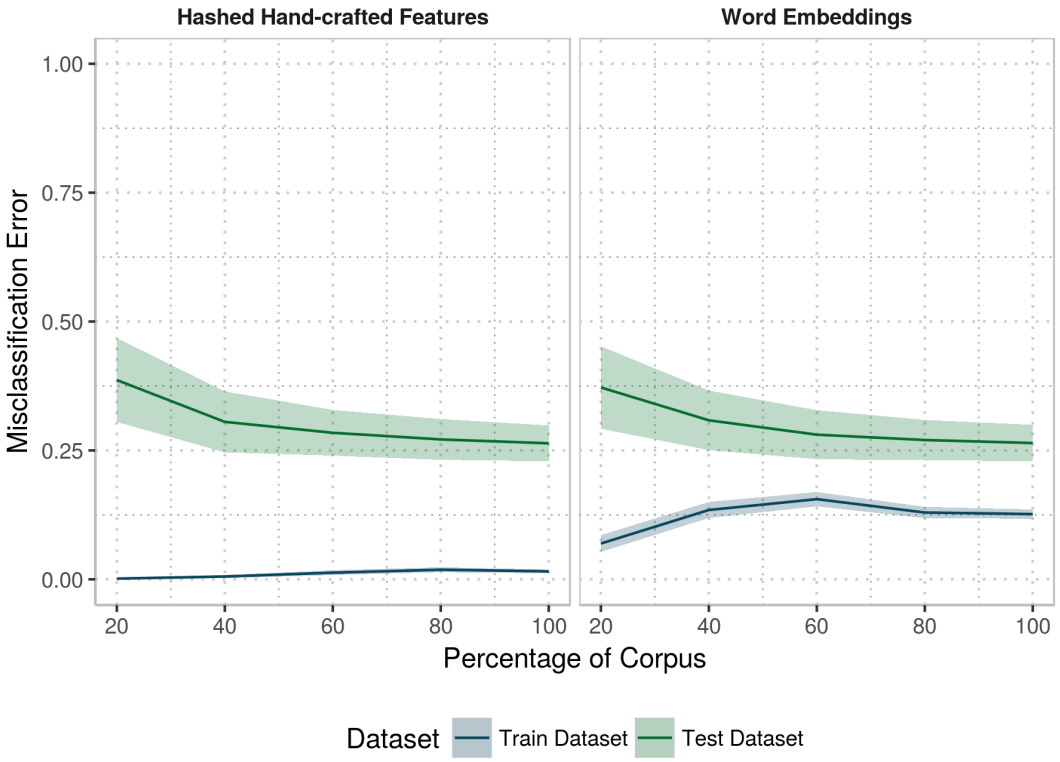


FIGURA 5.3: Curva de aprendizaje para distintos tamaños del corpus de entrenamiento para representaciones supervisadas y no supervisadas

La figura 5.3 muestra la curva de aprendizaje para diferentes tamaños de los datos de formación a través de dos representaciones diferentes de las características de entrada: características artesanales supervisadas y embeddings de palabras no supervisadas. La estructura de la curva de aprendizaje es la siguiente:

- Cada columna representa una representación del corpus: ingeniería de features supervisados y word embeddings no supervisados de un dominio específico (periodístico).
- La coordenada x muestra el tamaño de los datos de entrenamiento, como un porcentaje del total de datos de entrenamiento disponibles, comenzando desde el 20 % del corpus (el corpus fue particionado en 5 partes de acuerdo a lo establecido en el experimento 5.2).
- La coordenada y muestra el error de clasificación de un modelo.
- Hay dos colores que representan los conjuntos de datos: entrenamiento y evaluación.
- Las líneas sólidas más oscuras representan la media del error de clasificación a través de las diferentes particiones de los conjuntos de datos a lo largo de todos los modelos.

- El área sombreada, que tiene un color más claro, representa el error estándar de la media para el error de clasificación.

Es posible ver en la figura la diferencia entre las representaciones cuando se mide la tendencia a sobredimensionar. Los embeddings de palabras muestran menos diferencia entre los datos de formación y los datos de prueba con respecto a la clasificación errónea. Esto también se pudo ver en la Sección 4.4.6, donde SVM e Ingenua Bayes mostraron menos tendencia a sobreajustar el modelo.

En este caso, el clasificador es un clasificador no lineal con una fuerte tendencia a encajar, y los embeddings de palabras le están ayudando a no encajar tanto como lo hace la representación supervisada. Sin embargo, es importante señalar que el error de clasificación errónea en los datos de las pruebas sigue siendo aproximadamente el mismo para una u otra representación, por lo que el modelo no está sacrificando el rendimiento de la formación con el fin de obtener el rendimiento de la prueba. Lo que el modelo está sacrificando en realidad es información anecdótica y no generalizadora. Estos resultados apoyan la evidencia para aceptar la Hipótesis 2.2.

Es importante recordar que los modelos son pequeños y que los modelos de redes neuronales funcionan mejor cuanta más información tengan. Para obtener más información, se necesitan más ejemplos. Sin embargo, la anotación manual de ejemplos es costosa y está más allá de mis posibilidades. Por lo tanto, busco aumentar el número de ejemplos disponibles para el modelo mediante la incorporación de ejemplos de corpus no supervisados. Y para ello necesitamos modelos que se generalizan mejor a corpus fuera de dominio, como es el caso de los modelos con embeddings de palabras.

5.5 Conclusiones

Este capítulo introdujo una técnica semi-supervisada disjunta para el desambiguación de sentidos verbales del español. A partir de los desafíos expuestos en el capítulo 4 Quería superar la deficiencia de tener un modelo con una alta tendencia a sobredimensionar los datos. Declaré una hipótesis principal que quiero probar, que es que las representaciones no supervisadas ayudan a mejorar los modelos puramente supervisados al reducir la tendencia al sobreajuste dada por las características tomadas de los mismos datos en los que se entrena el modelo. La razón para plantear esta hipótesis es que las representaciones no supervisadas dan una versión suavizada como entrada a un clasificador y por lo tanto esto ayuda al clasificador a tener una versión más generalizada de los datos de los que está aprendiendo. Subdividí la hipótesis en subhipótesis más pequeñas y enfocadas.

Hipótesis 2.1 establece que el rendimiento de una representación no supervisada, en este caso un embeddings de palabras, depende del dominio desde el que se toman los datos no etiquetados para entrenar los embeddings. Los resultados para aceptar esta hipótesis están en la Sección 5.4.1. Se demostró que un dominio más específico da mejores resultados para la misma tarea. Sin embargo, algunos de los resultados

relativos al uso de un dominio fuera del dominio del supervisado necesitan un análisis más profundo para el trabajo futuro.

Antes de analizar los resultados de los experimentos para probar Hipótesis 2.2 Necesitaba una comparación entre representaciones supervisadas y no supervisadas. Esto se informa en la sección 5.4.2. En este experimento entreno dos modelos diferentes utilizando el clasificador de red neural con tres capas descrito en Sección 5.3.3, y cambiando la representación de entrada para el clasificador. De esta comparación, las características hechas a mano muestran un mejor rendimiento que las características no supervisadas. Sin embargo, la razón es que las características hechas a mano obtienen una representación más ajustada del conjunto de datos supervisado, ya que las características vienen directamente de allí. Los resultados de los experimentos para probar Hypothesis 2.2 muestran que las características hechas a mano dependen en gran medida del conjunto de datos, y que la representación resultante está muy bien adaptada a los ejemplos de formación.

La sección 5.4.3 muestra los resultados de Experiment 5.2, que mide la tendencia a sobredimensionar un clasificador. El experimento se realiza para probar Hipótesis 2.2, que establece que las representaciones no supervisadas (por ejemplo, los embeddins de palabras) producen menos sobreadaptaciones sobre los modelos supervisados que las representaciones supervisadas (por ejemplo, las características hechas a mano). El experimento comparó las características hechas a mano y los embeddings de palabras de dominio específicas. Como indica la hipótesis, el modelo entrenado con embeddings de palabras mostró una curva de aprendizaje más cercana entre los datos de entrenamiento y los datos de prueba. Esto da suficiente evidencia para apoyar la aceptación de Hipótesis 2.2.

Las representaciones de características no supervisadas, en particular las formadas en el mismo dominio que el conjunto de datos supervisados, muestran resultados prometedores. Sin embargo, el rendimiento todavía está por debajo de lo que puedo lograr utilizando representaciones puramente supervisadas. Y aunque puede haber muchas razones para explicar este resultado, según lo que veo, lo más probable es que tenga que ver con la (sobre)adaptación de las características supervisadas a los datos supervisados, en contraste con la representación más fluida de los embeddings de palabras.

Recordemos que en el Capítulo I anterior se encontraron dos desafíos principales. El primer desafío fue que los modelos entrenados con poca información tienden a encajar demasiado. El uso de clasificadores no lineales como las redes neuronales podría tener un gran impacto para minimizar el error e incluso maximizar el rendimiento de los datos de prueba. Pero el costo es la generalización de tales modelos. Las características no supervisadas ayudan en ese sentido al proporcionar una representación más suave que ayuda a la red neuronal a evitar la tendencia a sobredimensionarse.

El otro desafío fue la cobertura del modelo. La cobertura puede entenderse como la capacidad de predecir sobre ejemplos no vistos que pertenecen a una de las clases

objetivo y el modelo debería ser capaz de etiquetar. Estos ejemplos podrían eventualmente ser incorporados al corpus de capacitación y así agregar información al modelo. Pero como no están anotadas, el modelo aún no dispone de esa información. Dadas nuestras limitaciones para anotar ejemplos manualmente, los nuevos ejemplos se recopilan a partir de datos no etiquetados y se clasifican con el modelo. Si el modelo sólo se rige por la información que ya tiene (es decir, las características extraídas de los datos supervisados), entonces es difícil añadir la información de los datos no vistos y ampliar la cobertura. Los embeddings de palabras contienen información sobre los datos supervisados, utilizados para que el modelo los clasifique, y sobre datos no supervisados que aún no están presentes en el modelo. De esta manera, esta información ayuda a un modelo entrenado a partir de características no supervisadas a generalizarse mejor y ser capaz de recopilar información de nuevos ejemplos ampliando su cobertura.

Siguiendo estas razones, las embeddings de palabras parecen especialmente adecuadas para aplicar un enfoque que permita obtener nuevos ejemplos de corpus nunca antes vistos.

En el próximo capítulo me centraré en un primer acercamiento a un método semisupervisado que añade datos de fuentes no etiquetadas y los utiliza para expandir el modelo. Este método es el autoaprendizaje, un enfoque básico del aprendizaje semisupervisado.

El trabajo futuro de este capítulo incluye el uso de otras representaciones no supervisadas, como las enumeradas por Turian et al. [Turian et al., 2010]: Collobert y Weston [Collobert and Weston, 2008] y Brown clusters [Brown et al., 1992]. Otra línea de trabajo sería hacer un análisis de errores más exhaustivo sobre los diferentes dominios de embeddings de palabras, viendo si un mejor preprocesamiento de los datos puede proporcionar mejores resultados.

Parte III

Aprendizaje semisupervisado conjunto

Capítulo 6

Autoaprendizaje

6.1 Estructura

Con una tarea como desambiguación de sentidos verbales para el español, donde los datos anotados son pequeños, un enfoque puramente supervisado tiene que superar dos desafíos: el modelo se sobrepone a los datos, y la cobertura del modelo es pequeña. Estos fueron los hallazgos del capítulo 4. El capítulo 5 exploró un enfoque semi-supervisado proporcionando embeddings de palabras como entrada a un clasificador supervisado.

Los resultados del capítulo anterior evidenciaron a favor de la hipótesis . Recordemos que la hipótesis afirma que las embeddings de palabras tenían menos tendencia a sobredimensionarse que las características supervisadas. Los resultados sirvieron para aceptar esta hipótesis. Sin embargo, esta mejora en el overfitting vino con una disminución en el rendimiento: las características supervisadas se adaptaron mejor al problema, los embeddings de palabras tuvieron peores resultados. Aún así, los resultados de los embeddings de palabras fueron prometedores, ya que tener menos embeddings implica un mejor rendimiento en los corpus de dominios en general.

El otro desafío para esta tarea fue la cobertura. La cobertura puede ser pensada como los ejemplos invisibles, parte de una clase, que el modelo puede alcanzar (es decir, clasificar a la clase correcta). Los ejemplos que no están en formación podrían añadir información al modelo que todavía no está integrado en el modelo. Una forma de añadir tal información es añadiendo ejemplos nunca vistos al modelo. Los embeddings de palabras ayudan en esta dirección, no añadiendo nuevos ejemplos al modelo, sino añadiendo información de datos no etiquetados. Con esta información adicional, el modelo puede cubrir una gama más amplia de ejemplos.

Para incrementar la cobertura del modelo con nuevos ejemplos, exploro *autoaprendizaje*. Esta es una técnica semi-supervisada que expande los datos etiquetados usando la certeza del modelo sobre los datos no etiquetados. Añade nuevos ejemplos automáticamente, usando un modelo supervisado para etiquetar ejemplos sin etiquetar. Este es un algoritmo de aprendizaje semi-supervisado conjunto. Esto significa que los datos etiquetados y no etiquetados se utilizan conjuntamente en el proceso de aprendizaje. En contraste, el enfoque con embeddings de palabras es *disjunto* porque los embeddings se obtienen independientemente del modelo para desambiguación de sentidos verbales.

El algoritmo de autoaprendizaje funciona básicamente de la siguiente manera. Un algoritmo supervisado se entrena en un conjunto de datos etiquetado como semilla. Luego se utiliza el modelo obtenido para clasificar ejemplos no etiquetados obtenidos de un corpus grande no etiquetado. La certeza del algoritmo en la clasificación se utiliza para añadir automáticamente ejemplos etiquetados al conjunto de datos etiquetado. El algoritmo utiliza este conjunto de datos más grande (proveniente de datos supervisados y no supervisados) para entrenar un nuevo modelo. Este proceso se itera hasta que se cumple un criterio de parada.

En términos generales, el autoaprendizaje es un algoritmo simple que aplica una técnica de bootstrap para mejorar un modelo supervisado que mejora con datos no etiquetados. El objetivo de explorar esta técnica se basa en incrementar la cobertura del modelo como dije antes. El supuesto subyacente es que los nuevos ejemplos ayudarán al modelo a integrar la información que está latente en esos ejemplos.

Sin embargo, hay un desafío que tengo que enfrentar cuando vengo a capacitar a un modelo utilizando un enfoque de autoaprendizaje. Este desafío reside en el desequilibrio de las clases en el corpus. Si el modelo inicial está sesgado para clasificar las instancias como parte de la clase más común, este error será arrastrado por el modelo de autoaprendizaje y anotará automáticamente los nuevos ejemplos como la clase más frecuente sólo porque el modelo original lo hace.

Además, en este caso particular hay otro problema, y es el dominio específico del corpus sobre el que se forma el modelo inicial. Recordemos que SenSem, el recurso en el que estoy basando los experimentos, proviene de un dominio específico: el periodístico. Tener un dominio sesgado también afecta a las clases que aparecen más. Por lo tanto, algunas clases son las más comunes en un dominio general, pero este no es el caso para el dominio específico, como ilustran los siguientes ejemplos:

Ejemplo 6.1. El verbo “llegar” tiene 8 sentidos diferentes en SenSem. De ellos, uno es “alcanzar cierta altura o punto”. Este sentido no es el más común disponible en el corpus SenSem ya que tiene algunos ejemplos etiquetados en el corpus. Sin embargo, en un dominio más amplio se puede utilizar con una prevalencia más alta que en este dominio.

Estas son las razones que han motivado el estudio de diferentes técnicas en los capítulos anteriores para evitar el sobreequipamiento y el sesgo de la clase más frecuente en el algoritmo supervisado. Esto es importante para que el algoritmo de autoaprendizaje evite desviarse demasiado rápido a la clase más frecuente y no añadir suficiente información útil y diversa al modelo.

Este capítulo pondrá a prueba las siguientes hipótesis:

Hipótesis III *Las clases objetivo de un modelo entrenado en datos etiquetados pueden ser mejor representadas integrando más ejemplos, incluso si son automáticamente etiquetados (y por lo tanto contienen errores y sesgos). Aumentar el número de ejemplos para capacitar a un modelo ayuda a evitar el sobreequipamiento del modelo y mejora la cobertura.*

Para aceptar o rechazar tal Hipótesis, la divido en otras más específicas.

Primero hago dos hipótesis acerca de la bondad general de este enfoque para mejorar un enfoque mayormente supervisado:

Subhipótesis 3.1 *El rendimiento del modelo sobre los datos de prueba retenidos mejora después de que el algoritmo de autoaprendizaje añade nuevos ejemplos anotados automáticamente.*

Subhipótesis 3.2 *El modelo obtenido en cada iteración muestra un aumento en la certeza promedio para predecir la clase de ejemplos no vistos. Fin de la subhipótesis.*

En cuanto al sobreajuste, hago la siguiente hipótesis:

Subhipótesis 3.3 *Aumentar el número de ejemplos ayuda a reducir el sobreajuste.*

En cuanto a la cobertura, hago la siguiente hipótesis:

Subhipótesis 3.4 *El algoritmo de autoaprendizaje aumenta el número de características asociadas al modelo, lo que es un indicador de un incremento en la cobertura del modelo.*

Dado el sesgo inicial en la distribución de las clases y el sesgo del algoritmo de autoaprendizaje para amplificar este sesgo inicial y sobrepoblar una clase mayoritaria, se espera que las siguientes hipótesis sean rechazadas:

Subhipótesis 3.5 *El autoaprendizaje ayuda a mejorar el rendimiento del modelo en cada una de las clases de manera uniforme.*

Subhipótesis 3.6 *La representatividad de cada clase en el conjunto de datos se mantiene a través de todas las iteraciones del algoritmo.*

Subhipótesis 3.7 *La cobertura de las características se distribuye uniformemente entre las clases.*

Estas hipótesis serán aceptadas o rechazadas utilizando el siguiente esquema:

- El experimento 6.1 reporta el desempeño de un modelo antes y después de ejecutar el algoritmo de autoaprendizaje. El rendimiento se mide por la macro y la puntuación media ponderada F1 (métrica 1). Los resultados mostrados en la sección 6.4.1 sirven para probar y rechazar ligeramente la Hipótesis 3.1, que el rendimiento sobre un modelo en datos de prueba retenidos mejora después de las iteraciones de autoaprendizaje.

- El experimento 6.2 muestra la certeza del modelo sobre las clases predichas para cada iteración del algoritmo. El promedio de este valor es lo que necesito para probar Hipótesis 3.2 que establece que el modelo tiene un aumento en la certeza promedio a lo largo de las iteraciones de autoaprendizaje (Metric 5). Los

resultados de esta experimentación se muestran en la Sección 6.4.2 y muestran que la certeza no aumenta, por lo tanto esta Hipótesis es rechazada.

- El experimento 6.3 evalúa la sobreadaptación midiendo el error debido a la varianza de un modelo entrenado en un conjunto de datos sobre otros conjuntos de datos. Para medir esto utilizo la curva de aprendizaje (Metric 3) que ya expliqué en capítulos anteriores. Los resultados de los experimentos mostrados en la Sección 6.4.3 sirven para rechazar ligeramente la Hipótesis 3.3 que afirma que añadir nuevos ejemplos al conjunto de datos ayuda a disminuir la tendencia del modelo a adaptarse.

- El experimento 6.4 registra el número de características que tiene un modelo. Este número de características se mide por el recuento bruto. Los resultados mostrados en la Sección 6.4.4 sirven para aceptar la Hipótesis 3.4 de que el algoritmo de autoaprendizaje aumenta el número de características asociadas al modelo.

- El experimento 6.1 también sirve como base para probar Hipótesis 3.5, que el algoritmo de autoaprendizaje mejora el rendimiento de cada clase uniformemente. Para probar esta hipótesis miro los resultados desde una perspectiva diferente. En la sección 6.4.5 se discuten los resultados de esta hipótesis. Esta vez, sin embargo, no utilizo el promedio de la puntuación F1 (Metric 1), sino la puntuación F1 de cada clase antes y después de la iteración de autoaprendizaje. Los resultados muestran que, como se esperaba, la hipótesis es rechazada.

- El experimento 6.5 registra la distribución de las clases en el conjunto de datos de formación para cada una de las iteraciones del algoritmo de autoaprendizaje. Esta distribución se mide por el recuento proporcional de dichas clases. Los resultados de este experimento medidos con la métrica se muestran en la Sección 6.4.6 y sirven para rechazar la Hipótesis 3.6 de las clases que se distribuyen uniformemente a través de iteraciones de autoaprendizaje.

- El experimento 6.4 también se utiliza, medido de forma diferente esta vez, para probar Hipótesis 3.7. El experimento recoge el número de veces que una característica y una clase coexisten en el conjunto de datos de formación. Estos resultados se miden mediante dos métricas diferentes. En primer lugar, el recuento bruto de características por clase. Por otro lado, también me fijo en la asociación que cada clase tiene con cada característica. Para ello utilizo la información mutua puntualmente definida en Metric 4. Los resultados que sirven para rechazar la Hipótesis se muestran en la Sección 6.4.7.

En la sección 6.2 Reviso algunos trabajos anteriores realizados en autoaprendizaje en general y también aplicados específicamente a desambiguación de sentidos verbales. También menciono algunos otros métodos semi-supervisados basados en técnicas de bootstrap.

En la Sección 6.3 Repaso los puntos relevantes que conciernen a la experimentación realizada en el capítulo. La sección 6.3.1 reintroduce los recursos con los que trabajo en la experimentación. La mayoría de ellos ya fueron presentados y sólo hago un resumen rápido con referencias a la sección donde el recurso está mejor explicado. Siguiente en la Sección 6.3.2 Explico las características y representaciones con las que trabajo en este capítulo y, en particular, doy algunas pistas sobre cómo tratar la adición de nuevos ejemplos por parte del modelo y cómo esto afecta a la representación mediante la adición de nuevas características también. La sección 6.3.4 explica el detalle fino del algoritmo de autoaprendizaje, incluyendo el umbral de certeza y el criterio de parada que utiliza el algoritmo. Finalmente, las secciones **??** y **??**, al igual que en los capítulos anteriores, enumeran los experimentos y las métricas que utilizo para medir los resultados respectivamente.

La sección 6.4 informa de los resultados de los experimentos y los analiza para aceptar o rechazar las hipótesis planteadas en el capítulo.

Por último, en la sección 6.5 se extraen las conclusiones de este capítulo, recapitulando las Hipótesis y las implicaciones de aceptarlas o rechazarlas en función de las evidencias recogidas en los resultados. Establece las deficiencias de los métodos explorados en este capítulo y lo que quiero lograr en el siguiente. Termina esbozando el trabajo futuro.

6.2 Trabajo relevante

Como se explica en la Sección **??**, del Capítulo 2, el autoaprendizaje es uno de los primeros enfoques para utilizar una técnica de aprendizaje semi-supervisado con bastante tiempo en la literatura [Scudder, 1965]. La idea se ha aplicado en muchas tareas de lenguaje natural además de desambiguación de sentidos: subjetividad en sustantivos [Riloff et al., 2003], clasificación de diálogos [Maeireizo et al., 2004].

Para el autoaprendizaje aplicado a desambiguación de sentidos, el trabajo de Yarowsky [Yarowsky, 1995] para construir un modelo de desambiguación basado en las palabras co-ocurrentes con ejemplos etiquetados manualmente es fundamental en el área. La exploración realizada en este capítulo toma la mayor parte de sus ideas de este trabajo. Otro trabajo es el de Mihalcea [Mihalcea, 2004], en el que explora los diferentes ajustes de los hiperparámetros que se pueden tener en el algoritmo para mejorar el rendimiento de la tarea supervisada.

6.3 Metodología

Este capítulo explora el algoritmo de autoaprendizaje para expandir un modelo supervisado con datos no etiquetados anotados automáticamente basados en la certeza. El algoritmo de autoaprendizaje es un *algoritmo envolvente*. Esto significa que el algoritmo de autoaprendizaje no clasifica los datos por sí mismo. Lo hace envolviendo un

clasificador supervisado. Luego utiliza la información que el clasificador supervisado obtiene de los datos para aumentar el modelo.

Recordemos que se aprenden clasificadores independientes para cada lema. Describo la metodología en términos generales, pero sigo ejecutando un modelo de algoritmo de autoaprendizaje por cada lema. Es importante tener en cuenta que las métricas de evaluación pueden ocultar las diferencias de rendimiento entre los lemas.

6.3.1 Recursos

El autoaprendizaje requiere dos recursos principales: un conjunto de datos con y sin etiqueta. El conjunto de datos etiquetado es la semilla del modelo inicial. A partir de estos datos, el algoritmo inicia las iteraciones. El conjunto de datos sin etiqueta es la fuente de nuevos ejemplos para anotar automáticamente. La experimentación para este capítulo se hace sólo sobre los corpus españoles.

Inicialmente quise hacer un análisis general de los resultados de desambiguación de sentidos verbales para todos los lemas disponibles. Sin embargo, mientras se trabajaba en el análisis, dando una mirada mucho más cercana a los datos, se concluyó que el mejor método era tomar algunos lemas simbólicos y analizarlos con una mirada mucho más cercana para ver cómo funciona el algoritmo en cada uno de esos casos. Además de esto, la otra razón para hacerlo fue tener algún punto de comparación al hacer el aprendizaje activo en el siguiente capítulo, ya que la disponibilidad de datos etiquetados manualmente era limitada.

Como lemas simbólicos, seleccioné un verbo de comunicación, originalmente "hablar" tenía la intención, pero había ejemplos anotados de un solo sentido, por lo que se eligió "explicar". También seleccioné "pensar", otro verbo de la clase de comunicación, para observar similitudes y diferencias. Dos verbos de movimiento fueron los siguientes: "acceder" y "llegar". Finalmente se seleccionaron dos verbos transitivos: "buscar" y "facilitar".

6.3.1.1 SenSem

El corpus SenSem se utiliza como la semilla anotada en la que se inicia el algoritmo de autoaprendizaje. Para obtener más información sobre este recurso, consulte la Sección 4.3.1.1.

El principal problema con el uso de SenSem como en el autoaprendizaje es cómo se ve afectado por la tendencia del algoritmo a divergir hacia la clase más frecuente. Como mostraré en los resultados más adelante en este capítulo, este es un gran problema con respecto al autoaprendizaje. Un primer enfoque que tuve que seguir para tratar este problema fue el sobremuestreo.

Sobremuestreé al azar los ejemplos de las clases menos frecuentes para que el conjunto de datos inicial (semilla) tuviera un número equivalente de ejemplos para cada clase. Esto no solucionó completamente la tendencia del algoritmo a clasificar todo como parte de la clase más frecuente, pero ayudó a suavizarla. Si usé en los

experimentos el conjunto de datos original sin sobremuestreo, entonces la deriva del algoritmo fue mucho más rápida. Este sobremuestreo no tuvo ningún impacto en el rendimiento del algoritmo supervisado (es decir, el clasificador entrenado sólo en datos etiquetados) sobre el corpus de pruebas, pero ayudó en el caso de las iteraciones de autoaprendizaje para retrasar la deriva a la clase más frecuente.

6.3.1.2 SBWCE

.

La fuente de datos no etiquetada de las nuevas instancias para el algoritmo es el SBWCE. Este recurso ya fue descrito en el capítulo 5, sección 5.3.1.2.

Debido a limitaciones de tiempo y recursos no pude utilizar el recurso completo. El algoritmo toma y clasifica los datos no etiquetados para seleccionar las instancias. Luego, con el conjunto de datos SBWCE completo, cada iteración tomaría una gran cantidad de tiempo para completarse. La solución era muestrear aleatoriamente un número fijo de frases sin etiquetar para cada lema. Para cada lema elegí 1000 frases sin etiquetar y las utilicé como datos sin etiquetar para el autoaprendizaje.

Una vez seleccionadas las sentencias, el corpus es preprocesado para agregar anotaciones PoS y de dependencia. El paso de preprocesamiento se realiza con Freeling. La idea clave es tener en el conjunto de datos sin etiqueta el mismo tipo de información que el conjunto de datos supervisado, tal y como se describe en 4.3.1.1. Esta información se utiliza para construir las características artesanales.

6.3.2 Atributos

El capítulo anterior mostró que los embeddings de palabras son prometedoras, pero las funciones supervisadas siguen teniendo el mejor rendimiento. Decidí mantener ambos tipos de características y comparar el rendimiento de cada una en tareas de aprendizaje semisupervisadas conjuntamente. Los parámetros para ambos tipos de representaciones son los que tienen el mejor rendimiento en los capítulos anteriores.

6.3.2.1 Atributos manuales

Para las funciones artesanales utilizaron los atributos descriptos en el Capítulo 4, Sección 4.3.2.

Recordemos que todo el conjunto de características hechas a mano para los datos supervisados ya era grande a pesar de que el conjunto de datos etiquetado era pequeño. Cuando esto se hace para conjuntos de datos no etiquetados, es muy probable que la cantidad de características sea demasiado grande para caber en la memoria.

Además, el modelo original supervisado del algoritmo de autoaprendizaje se entrena sólo en los datos supervisados. Entonces sólo tiene las características que tiene el corpus supervisado. No es probable que estas características sean las únicas disponibles en el conjunto de datos sin etiquetar.

Como la mayoría de los clasificadores toman una entrada de tamaño fijo, tener nuevas características a partir de datos no etiquetados ofrece dos opciones: ignorar las características o generar una representación con las características de todo el conjunto de datos (es decir, etiquetados y sin etiquetar).

El hashing de características, descrito en detalle en la sección 4.3.2.2, ofrece una solución muy eficaz a este problema. Fue desarrollado con el aprendizaje en línea en la consideración donde los datos etiquetados están en curso.

6.3.2.2 Word embeddings

En el capítulo 5 De la experimentación deduzco que el dominio de los embeddings de palabras afectó al rendimiento del modelo. Para estos experimentos seleccioné los embeddings de palabras que mostraron los mejores resultados en el capítulo anterior. Son las que se obtienen del corpus periodístico descrito en el apartado 5.3.1. Los embeddings fueron entrenadas usando word2vec y las instancias son construidas por concatenación de los embeddings, el proceso se describe en la Sección 5.3.2.2.

6.3.3 Clasificadores

En la siguiente sección describo los detalles del algoritmo de autoaprendizaje. Sin embargo, recuerde que este es un método de envoltura. Entonces todavía necesito un clasificador supervisado para usar como parámetro del modelo de autoaprendizaje. El clasificador es un hiperparámetro del algoritmo de autoaprendizaje.

Based on the results of Chapter 4 I selected the multilayer perceptron with three layers of size 500, 250 and 100. For the chapter all the experiments have this classifier fixed.

Basado en los resultados del capítulo 4 seleccioné el perceptrón multicapa con tres capas de tamaño 500, 250 y 100. Para el capítulo todos los experimentos tienen arreglado este clasificador.

6.3.4 Algoritmo de autoaprendizaje

Como expliqué antes, el algoritmo de autoaprendizaje es un algoritmo de envoltura sobre un clasificador. El algoritmo toma los siguientes parámetros iniciales:

- Un conjunto de datos etiquetados para entrenar el modelo supervisado inicial.
- Un conjunto de datos de evaluación para calcular el desempeño del modelo antes y después de finalizadas todas las iteraciones del algoritmo de autoaprendizaje.
- Un conjunto de datos de validación para llevar registro del algoritmo de autoprendizaje y evitar que las instancias agregadas automáticamente hagan diverger mucho del modelo inicial.
- Un conjunto de datos no etiquetados para obtener los nuevos datos que se anotarán automáticamente.

- Un clasificador probabilístico para usar en el proceso de anotación automática.
- Una tolerancia de error para detener el algoritmo en caso de error.

El conjunto de datos de prueba etiquetado es el mismo que se utilizó en los experimentos de los capítulos anteriores. El conjunto de datos de validación se toma del conjunto de datos de formación siguiendo la misma estructura que se describe en la sección 4.3.1.1. El corpus SenSem se dividió en un 80 % para la formación y un 20 % para la prueba, pero manteniendo al menos una instancia para cada clase (sentido) en el conjunto de datos de la prueba y dos instancias en el conjunto de datos de la formación. El conjunto de datos de validación se tomó del conjunto de datos de formación dividiéndolo de nuevo en 80 %-20 %. Esta división también aseguraba que hubiera al menos un ejemplo de formación y un ejemplo de validación para cada clase en el conjunto de datos.

Al igual que con las divisiones de formación/prueba, las divisiones de formación/validación también conservan la distribución de las clases observadas en todo el conjunto de datos. Esto implica que en cada división no se pueden encontrar más ejemplos de una clase de los que se podrían encontrar en una muestra estratificada del conjunto de datos, es decir, una muestra que conserve la distribución de todo el conjunto de datos. Sin embargo, esto no es necesariamente válido para las clases minoritarias, porque al menos un ejemplo debe encontrarse en cada escisión, incluso si eso implica una representación excesiva de la clase.

El algoritmo comienza entrenando un modelo inicial a partir de los datos de entrenamiento etiquetados. A partir de ese modelo obtengo el rendimiento inicial del clasificador a lo largo de la prueba y el error de clasificación de los datos de validación. Los primeros se utilizan para comparar el modelo antes y después de las iteraciones de autoaprendizaje. Este último se utiliza como criterio de parada para el algoritmo.

El algoritmo de autoaprendizaje consiste en un bucle que utiliza el modelo entrenado para recopilar nuevos datos del conjunto de datos sin etiquetar. En la primera iteración el modelo es el inicial entrenado con los datos de entrenamiento etiquetados.

En cada iteración del bucle el modelo clasifica todo el conjunto de datos sin etiquetar. A partir de esta clasificación, el algoritmo selecciona las instancias que se anotarán automáticamente. Estas instancias se seleccionan en base a la certeza (probabilidad) que tiene el clasificador de que las instancias son parte de la clase en la que fueron clasificadas.

La certeza del modelo sobre las instancias debe estar por encima de un umbral que debe anotarse automáticamente. Este umbral es otro hiperparámetro del algoritmo de autoaprendizaje. En la siguiente sección se explica con más detalle cómo seleccionar este umbral.

Con las instancias y sus etiquetas correspondientes (dadas por el clasificador), además de los datos etiquetados recolectados hasta ahora (que consisten en el conjunto

de datos de entrenamiento original y los nuevos datos añadidos después de cada iteración) el algoritmo entrena un nuevo modelo. El modelo se prueba en el conjunto de datos de validación para obtener el error de clasificación. El error no puede ser mayor que el error de validación más bajo hasta ahora más el hiperparámetro de tolerancia del error, de lo contrario el algoritmo se detiene y el modelo final es el último modelo obtenido hasta ahora. En caso de que el error sea inferior a la tolerancia, el modelo se actualiza para la siguiente iteración. Las instancias seleccionadas se añaden como parte del nuevo conjunto de formación para la siguiente iteración y se eliminan del conjunto de datos sin etiquetar. El algoritmo continúa el bucle hasta que se cumpla el criterio de parada o hasta que no haya más datos que añadir. Esto puede ocurrir en dos ocasiones: todos los datos no etiquetados ya forman parte del modelo o el clasificador no tiene suficiente certeza sobre ninguna de las posibles instancias para añadirlos al nuevo modelo. Una vez que el bucle del algoritmo termina por cualquier razón, el último modelo se utiliza para medir el rendimiento sobre el conjunto de datos de prueba.

El algoritmo tiene dos parámetros importantes: el umbral de certeza para añadir nuevos ejemplos y la tolerancia de error para detener el algoritmo antes de que se consuman todos los datos. Hay muchas maneras de establecerlos. En las próximas dos secciones explicaré más detalladamente mis decisiones sobre estos dos parámetros.

6.3.4.1 Umbral de certeza

En cada iteración del algoritmo después de entrenar un nuevo modelo (es decir, utilizando los datos de entrenamiento inicial más las instancias añadidas de iteraciones anteriores) el modelo se utiliza para recopilar nuevos ejemplos de entrenamiento del conjunto de datos no etiquetados. El clasificador es probabilístico, por lo que naturalmente tiene una certeza sobre los datos que clasifica. Esta certeza es la probabilidad dada a una instancia por el clasificador de pertenecer a una determinada clase. El umbral de certeza es la probabilidad mínima que el clasificador necesita en una instancia para que el algoritmo de autoaprendizaje lo integre como un nuevo ejemplo de formación: cualquier ejemplo en el que el clasificador tenga una certeza superior al umbral se anota automáticamente y se añade para formar un nuevo modelo.

En los experimentos iniciales, el umbral se fijó al principio del algoritmo. Se utilizó el mismo valor inicial para todas las iteraciones. Este valor era el mismo para todos los lemas. Recuerden que tengo un clasificador por lema para desambiguar. En este caso, ejecuto un conjunto de iteraciones del algoritmo por lema como se explicó anteriormente. Tener un umbral fijo para los diferentes lemas puede no ser adecuado, ya que cada lema tiene sus propias propiedades que lo diferencian del resto.

Buscando una solución más adecuada, lo primero que hay que explorar es definir un umbral inicial para la primera iteración y adaptarlo después de cada paso hasta encontrar un buen valor. Sin embargo, hay dos problemas con este enfoque: cómo establecer el valor umbral inicial y cómo definir un "buen valor" automáticamente.

Otra opción es discutida por Mihalcea [Mihalcea, 2004]. Presenta un algoritmo de autoaprendizaje que selecciona las N instancias superiores según la certeza del clasificador en ellas y las integra como nuevos ejemplos de entrenamiento. Sin embargo, la única manera de elegir ese número N para este caso es empíricamente. Por lo tanto, todavía sufrimos el mismo problema que con el umbral, la forma de estimar el parámetro.

De una forma u otra, tengo el mismo reto: no ser capaz de definir con suficiente confianza qué valores establecer para los hiperparámetros. Además, tampoco hay reglas generales para hacerlo. Así que decidí ir con un umbral de certeza para que el algoritmo no se limite a añadir sólo un número fijo de elementos. Utilizando un umbral, el algoritmo de autoaprendizaje puede añadir todos los elementos con gran certeza.

Para evitar seleccionar el umbral manualmente, decidí que el umbral fuera 100 % de certeza en cada iteración. Si no hay casos en este nivel de certeza, el umbral es bajado por un hiperparámetro alfa, en este caso 5 %. Con el nuevo umbral el algoritmo busca instancias. El umbral se reduce hasta que se encuentra al menos una instancia por encima del umbral o la selección de instancias se convierte en aleatoria. La selección aleatoria se identifica siempre que el umbral de certeza es igual a la probabilidad aleatoria de seleccionar una clase en el algoritmo (es decir, si el conjunto de datos tiene 2 clases, el umbral está cerca del 50 %, si el conjunto de datos tiene 3 clases, el umbral está cerca del 33 %, etc.). Cuando este es el caso, el algoritmo se detiene y no selecciona ninguna instancia. En la implementación, restrinjo aún más este criterio de parada deteniendo un 10 % antes de la cifra de selección aleatoria para cada lema.

Un criterio de parada más adecuado sería la distribución de las clases realmente observadas en los datos en lugar de una distribución uniforme. El trabajo adicional explorará esta variación del algoritmo.

6.3.4.2 Criterio de parada para el conjunto de datos de validación

Una vez que las instancias fueron seleccionadas por el clasificador (asumiendo que se hizo una selección y el algoritmo no se detuvo porque no agregó ningún ejemplo), se evalúa su impacto en el modelo. Este es el objetivo del conjunto de datos de validación. El algoritmo entrena un nuevo modelo con las instancias y los datos acumulados hasta el momento, es decir, los datos originales supervisados y las instancias anotadas automáticamente añadidas en iteraciones anteriores. Con el modelo obtenido el algoritmo obtiene el error de clasificación erróneo de la evaluación del modelo en el conjunto de datos de validación. El error se compara con el error mínimo de clasificación obtenido hasta ahora más una tolerancia. Al igual que el umbral de clasificación, la tolerancia de error comienza en cero y se incrementa gradualmente hasta un error de clasificación erróneo máximo que es un 10 % menos que el azar.

La razón para utilizar un conjunto de datos de validación para el criterio de parada es evitar adaptar el modelo al conjunto de datos de prueba. Los parámetros e hiperparámetros del modelo se adaptan así al conjunto de datos de validación. Esto

también se conoce como afinación del modelo. Posteriormente, cuando el modelo se evalúa en el corpus de pruebas, los resultados son realmente representativos.

Un enfoque alternativo a esto sería *k-fold cross-validation* sobre el conjunto de datos de formación en cada iteración (es decir, el conjunto de formación con los datos etiquetados y los datos anotados automáticamente). Entonces, el error medio de clasificación errónea podría utilizarse como criterio de parada. Sin embargo, la tendencia del algoritmo a incluir más ejemplos de las clases más frecuentes socava la idoneidad de este enfoque. De hecho, el error de clasificación errónea puede ser cada vez menor simplemente porque lo estoy evaluando en un conjunto de datos que se ha vuelto desequilibrado. Entonces, cuando el clasificador predice que las nuevas instancias pertenecen a la clase más frecuente, hay una mayor probabilidad de que estén correctamente clasificadas. Por el contrario, la evaluación en un set de datos de prueba retenido conserva la distribución original de clases.

6.3.5 Experimentos

Recordemos que todos los experimentos utilizan una versión sobremuestreada del conjunto de datos inicial etiquetado para evitar que el conjunto de datos se desvíe a la clase más frecuente inmediatamente.

El experimento 6.1 informa del rendimiento del modelo sobre el conjunto de datos de prueba antes y después de las iteraciones de autoaprendizaje. Primero, el experimento reporta el desempeño del modelo entrenado sólo con datos supervisados. Posteriormente se informa del rendimiento del modelo entrenado con los datos originales supervisados más los datos anotados automáticamente añadidos por el algoritmo de autoaprendizaje. Recordemos que todos los experimentos se realizan en base a un lema.

Experimento 6.1.

6.1a Entrear un modelo con los datos anotados de entrenamiento.

6.1b Evaluar el modelo sobre los datos de evaluación.

6.1c Correr el algoritmo de autoaprendizaje hasta que se detenga.

6.1d Evaluar el modelo obtenido con el algoritmo de autoaprendizaje sobre el conjunto de datos de evaluación.

El experimento 6.2 realiza un seguimiento de la certeza media para cada iteración del algoritmo de autoaprendizaje. El experimento registra la certeza que el algoritmo tiene sobre la clasificación de las instancias.

Experimento 6.2.

6.2a Inicial las iteraciones de autoaprendizaje.

6.2b Run the model obtained up to this point into the whole unlabeled dataset (part of the self-learning algorithm).

6.2c Correr el modelo obtenido hasta este punto en la totalidad del conjunto de datos no anotados (esto es parte del algoritmo de autoaprendizaje).

6.2d Before selecting the instances record the predicted classes for each instance and the probability (certainty) they have for the model.

6.2e Antes de elegir las instancias, guardar las clases predichas para cada instancia y su probabilidad (certeza) dentro del modelo.

6.2f Continuar con el algoritmo de autoaprendizaje.

El experimento 6.3 se realiza para evaluar el *error debido a la varianza* del modelo entrenado por el algoritmo de autoaprendizaje. El objetivo de este experimento es medir la tendencia al overfit de un modelo a medida que el número de ejemplos aumenta con el algoritmo de autoaprendizaje. Se basa en los experimentos de los capítulos anteriores, que también informan sobre el sobreequipamiento. Sin embargo, cambia en un aspecto fundamental. El capítulo anterior tuvo que trabajar con un corpus de tamaño limitado y mostró la curva de aprendizaje del experimento dividiendo el conjunto de datos. En este caso, el conjunto de datos añade nuevos ejemplos mediante el diseño del algoritmo de autoaprendizaje. Por lo tanto, mido cómo estos nuevos ejemplos afectan la curva de aprendizaje. Recuerde que esto se hace en base a un lema. La estructura del experimento es la siguiente:

Experimento 6.3.

6.3a Tomar solo una porción de todo el conjunto de datos no anotado para usar como instancias.

6.3b Mezclar los conjuntos de entrenamiento y validación y particionarlos aleatoriamente con muestreo estratificado.

6.3c Start the self-learning algorithm.

6.3d Iniciar el algoritmo de autoaprendizaje.

6.3e Para cada paso, entrenar el clasificador con los datos de entrenamiento obtenidos antes de iniciar el algoritmo de entrenamiento más las instancias anotadas automáticamente.

6.3f Registrar las predicciones sobre los datos de entrenamiento y validación.

6.3g Repeat the whole procedure n times with a different portion of the unsupervised dataset.

6.3h Repetir todo el procedimiento n veces con una porción diferente de los datos no anotados.

La repetición del procedimiento n veces emula lo que en los capítulos anteriores se hizo utilizando la división aleatoria del conjunto de datos supervisado. El cambio del conjunto de datos no supervisados y la reorganización y división de los conjuntos de datos de formación y validación evalúa el clasificador en nuevos conjuntos de datos y muestra en qué medida varía en función de los diferentes conjuntos de datos. Esto se utiliza para generar un estimador de la tendencia del modelo a sobredimensionarse.

El experimento 6.4 informa de las características asociadas al modelo. El experimento registra las características para cada instancia de cada clase. Esto muestra el número de características que cubre todo el modelo, así como el número de características para cada clase.

Experimento 6.4.

6.4a Registrar el número de veces que una tupla $(clase, atributo)$ ocurre para cada clase y cada atributo del conjunto de datos supervisados de entrenamiento.

6.4b Ejecutar una iteración del algoritmo de autoaprendizaje.

6.4c Registrar el número de veces que una tupla $(clase, atributo)$ ocurre para cada clase y cada atributo del conjunto de entrenamiento compuesto por los datos supervisados y las instancias anotadas automáticamente.

6.4d Repetir los pasos previos para cada iteración del algoritmo.

El experimento 6.5 registra la distribución de las clases a través de una ejecución del algoritmo de autoaprendizaje. El experimento registra el número de instancias que pertenecen a una clase en cada paso del algoritmo. Esta cifra es la suma de los datos originales supervisados más los datos anotados automáticamente.

Experimento 6.5.

6.5a Registrar el número de veces que cada clase ocurre en el conjunto de datos supervisados de entrenamiento.

6.5b Ejecutar una iteración del algoritmo de autoaprendizaje.

6.5c Registrar el número de veces que cada clase ocurre con el nuevo conjunto de datos compuesto por los datos supervisados y las instancias anotadas automáticamente.

6.5d Repetir los pasos previos para cada iteración del algoritmo.

6.3.6 Métricas

Las métricas vistas en los capítulos anteriores también se utilizan en este capítulo. Recuerden que hasta ahora he definido dos métricas principales para reportar resultados

diferentes. La métrica 1 (definido en la sección 4.3.5.1) es la macro y la media ponderada de la puntuación F1. Esta métrica es útil para tratar el problema de evaluar el rendimiento cuando las clases están desequilibradas.

El error debido a la varianza se mide con Metric 3 (definido en Sección 4.3.5.3). El objetivo de esta métrica es medir cuánto sobrescribe un modelo los datos a medida que aumenta el tamaño del conjunto de datos.

La información mutua punto a punto (PMI) es una medida estadística de la asociación entre eventos. Se define como:

$$\text{pmi}(x;y) = \log \frac{p(x,y)}{p(x)p(y)}$$

Métrica 4. La PMI entre características y clases se evalúa mediante iteraciones utilizando los siguientes pasos para cada iteración del algoritmo:

4a Calcular la PMI para cada tupla $(clase, atributo)$.

4b Cada atributo es asociado a la clase con mayor PMI.

4c Calcular la media y el error estándar de la media de todas las PMIs para cada atributo asociado a una clase, por cada clase.

4d The PMI data for that iteration is stored and the process is repeated in the next iteration.

4e La información de la PMI para esa iteración es guardada y el proceso se repite para la iteración siguiente.

Para obtener la certeza promedio del clasificador sobre los datos utilizo la siguiente métrica:

Métrica 5. Para cada iteración del algoritmo calcular:

5a Obtener la certeza del modelo sobre los datos no anotados a predecir.

5b Guardar la media y el error estándar de la media para la certeza de todos los ejemplos en esa iteración.

5c Continuar con la iteración siguiente.

6.4 Análisis de resultados

En esta sección se reportan los resultados obtenidos a través de los experimentos descritos anteriormente. Este capítulo tiene un mayor número de resultados y gráficos que cualquiera de los otros. Esto es consecuencia de tener el mayor número de hipótesis en todos los capítulos. Esto significa que los resultados mostrados en las cifras hicieron que fuera bastante difícil de seguir. La visualización y las métricas muestran sólo una

parte de todos estos resultados, los que consideré más relevantes. Sin embargo, esto puede llevar a que algunos resultados se oscurezcan. Haré todo lo que esté en mi mano para dejar claro cuando algún resultado esté siendo oscurecido.

6.4.1 Hipótesis 3.1

Exploro los resultados para probar Hypothesis 3.1, que establece que el rendimiento de un modelo sobre un conjunto de datos de prueba resistida mejora con el algoritmo de autoaprendizaje.

Recordemos que sólo hubo dos momentos en los que el modelo fue evaluado sobre los datos de la prueba: la iteración inicial y la final. La iteración inicial se evalúa antes de iniciar las iteraciones del algoritmo de autoaprendizaje, cuando el modelo se entrena sólo en datos supervisados. La iteración final se evalúa después de que el algoritmo de autoaprendizaje se detiene. Esta vez el modelo es el último obtenido por el algoritmo de autoaprendizaje, entrenado tanto con datos manualmente supervisados como automáticamente anotados. La idea es evaluar el impacto de los nuevos datos añadidos en el modelo, según lo evaluado en el conjunto de datos de prueba original.

La figura 6.1 reporta los resultados del desempeño en el corpus de pruebas antes y después del autoaprendizaje para cada uno de los lemas simbólicos que presenté en la Sección 6.3.1. La parcela es una parcela de bar estructurada de la siguiente manera:

- Cada fila muestra los resultados para un lema testigo: “acceder”, “buscar”, “explicar”, “facilitar”, “llegar”, y “pensar”.
- Cada columna es un tipo de representación: atributos manuales y vectores de palabras periodísticos.
- Cada grupo de barras en cada gráfico representa una métrica: promedio macro y ponderado del F1-score.
- Cada gráfico de barras de color distinto dentro de un grupo representa el momento de evaluación: iteración inicial y final.
- El gráfico de barras representa el valor de la métrica.

Como se explica en la sección 4.3.5.1, el promedio ponderado es útil para mostrar el rendimiento del algoritmo dada la distribución de clases (es decir, las clases más frecuentes son más importantes en el resultado final). Por otro lado, el macro promedio es útil para ver el desempeño en clases menos frecuentes. Luego, el margen entre la macro y el promedio ponderado muestra cuán sesgado está el algoritmo para la clase más frecuente.

Tenga en cuenta que, como se describe en el capítulo anterior, los embeddings de palabras tienen un rendimiento medio peor que las características hechas a mano. Sin embargo, en dos de los verbos simbólicos que se muestran aquí, los embeddings de palabras funcionan mejor que las características hechas a mano, en uno de los

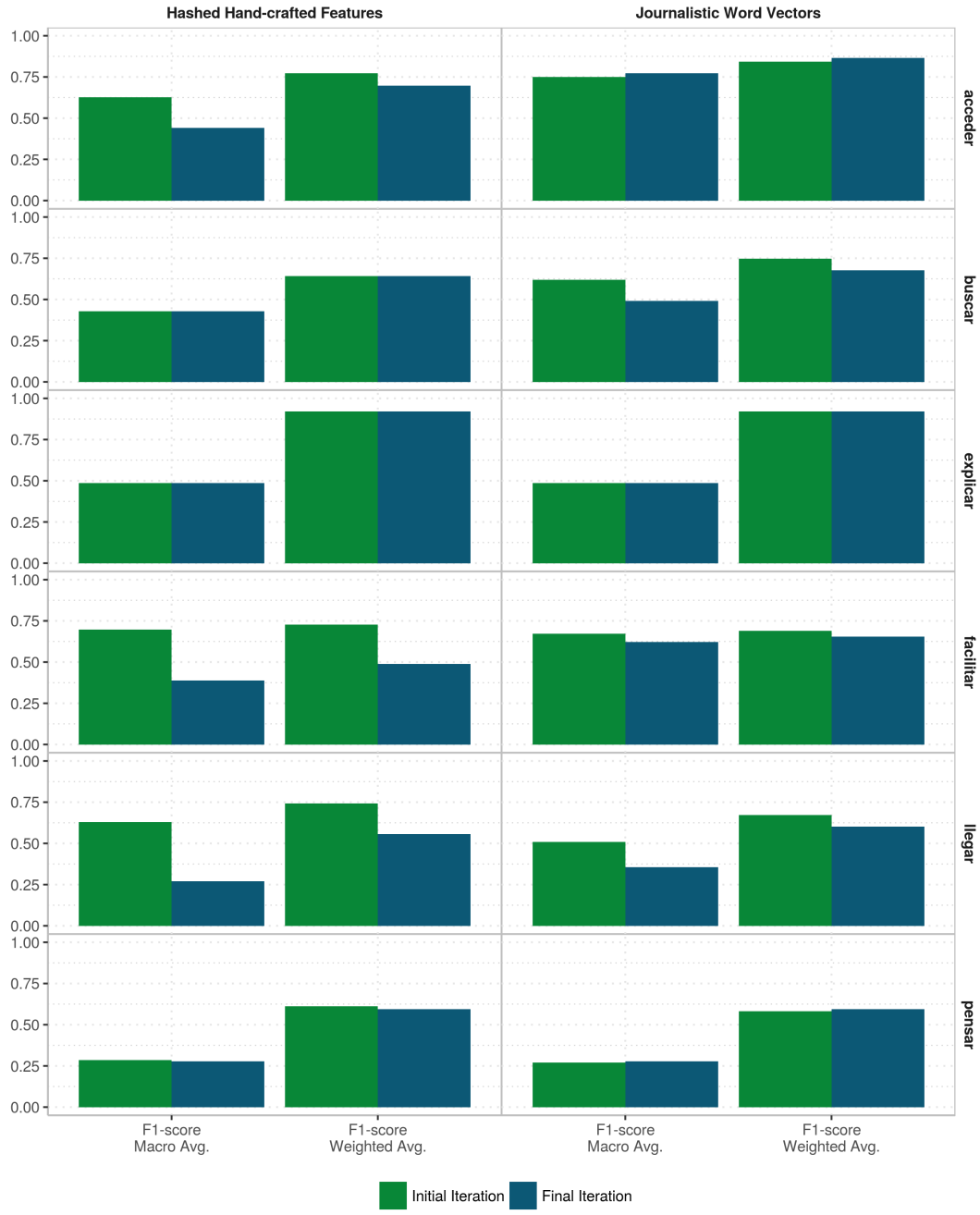


FIGURA 6.1: Comparación de los promedios macro y ponderado para el F1-score antes y después de las iteraciones del algoritmo de autoaprendizaje

verbos funcionan al mismo nivel y en tres casos funcionan un poco peor. En realidad, aunque el promedio era diferente, la mediana era muy similar para los embeddings de palabras y las características hechas a mano, así que podemos considerar que estos verbos simbólicos son representativos de la población.

La figura muestra que, a excepción de dos lemas, las características hechas a mano pierden rendimiento después de que termina el algoritmo de autoaprendizaje. En ninguno de los lemas el algoritmo mejora el rendimiento sobre el modelo original sin los

nuevos datos. Para embeddings de palabras el caso es un poco diferente, ya que en la mitad de los lemas no pierde rendimiento, e incluso mejora en dos de ellos.

Comparando la pérdida de rendimiento para las dos representaciones diferentes, la caída es más fuerte para las características artesanales, especialmente en el promedio macro. A partir de estos resultados puedo formular la hipótesis de que para las características artesanales el algoritmo de autoaprendizaje es más propenso a añadir instancias de la clase más frecuente. Esto repercute en el rendimiento de las clases menos frecuentes y es la razón por la que la caída media macro del rendimiento es mucho mayor.

Este descenso en el rendimiento medido por el macro promedio para los vectores de palabras no es tan pronunciado como para las características artesanales. Además, la caída general en el rendimiento, en los casos en que ocurre, no es tan grande como en el caso de las características artesanales, incluso si estas últimas tienen un mejor rendimiento para la iteración inicial (por ejemplo, “facilitar”, “llegar”).

Se puede concluir que, en términos generales, los vectores de palabras funcionan mejor que los elementos artesanales en este entorno. Esto da una fuerte indicación del sesgo del dominio en la tarea. Las características hechas a mano pueden tener un mejor rendimiento en la tarea con el conjunto de datos supervisados originales, pero no se generalizan bien a un dominio mayor. En este caso los embeddings de palabras proporcionan una mejor representación porque, como vimos en el capítulo anterior, recordando Hipótesis 2.2, los embeddings de palabras tienen menos tendencia a sobredimensionar un modelo. Esto es crucial en un entorno como el autoaprendizaje, donde los nuevos ejemplos se anotan automáticamente. En este escenario, una mejor representación hace que el modelo se adapte mejor a los nuevos datos. Es decir, las características hechas a mano están sobreadaptando el modelo al conjunto de datos inicial etiquetado y ese sobreadaptación se está transfiriendo a las decisiones del algoritmo de autoaprendizaje con respecto a las instancias para anotar automáticamente, produciendo una deriva con un sesgo a la clase mayoritaria. Por lo tanto, puedo concluir que los vectores de palabras producen una mejor generalización de los datos para ejemplos no vistos.

Los resultados no muestran suficiente evidencia para aceptar la Hipótesis 3.1. En particular, las características hechas a mano muestran evidencia para rechazar la Hipótesis a medida que el rendimiento general se degrada después de que se ejecuta el algoritmo de autoaprendizaje. Sin embargo, cuando los embeddings de palabras se integran en el entorno, los resultados son más fuertes para aceptar la Hipótesis, aunque no sean concluyentes.

6.4.2 Hipótesis 3.2

La presente sección discute los resultados que encontré con respecto a Hipótesis 3.2. La Hipótesis indicó que el modelo entrenado después de cada iteración de autoaprendizaje, añadiendo así nuevos datos de fuentes no etiquetadas, aumentaba la certeza promedio para predecir la clase de nuevos ejemplos no vistos. Detrás de esta hipótesis

está la suposición de que el modelo, al añadir nuevos ejemplos no etiquetados, puede representar mejor los nuevos ejemplos.

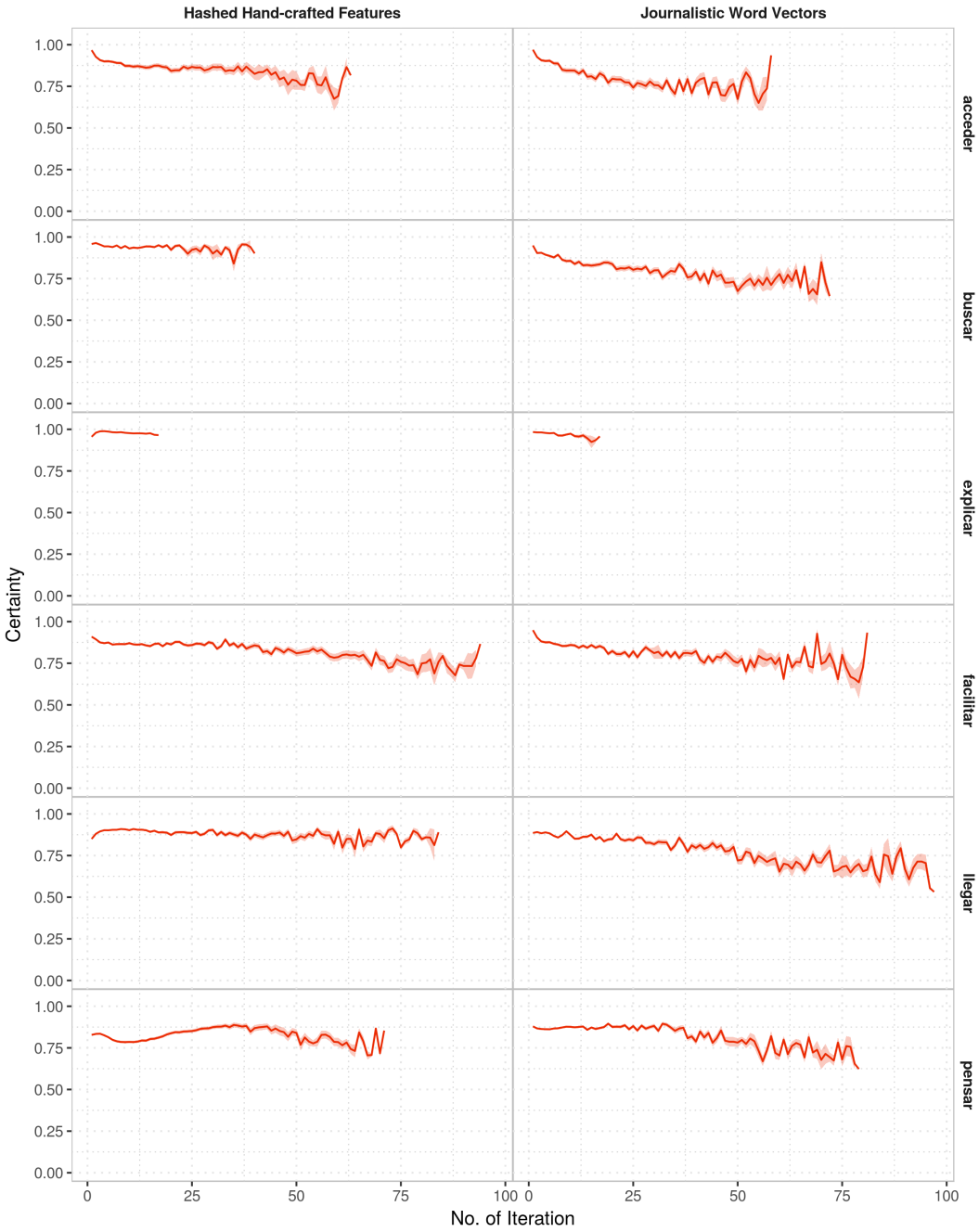


FIGURA 6.2: Certeza promedio para predecir nuevas clases para cada iteración de autoaprendizaje

La figura 6.2 muestra la certeza media del modelo en nuevos ejemplos a lo largo de las iteraciones de autoaprendizaje. Los datos se obtienen ejecutando Experiment 6.2, que registra la certeza de las clases predichas del modelo para todos los nuevos ejemplos del conjunto de datos no etiquetados. La métrica para medir esto es un promedio no ponderado descrito en Metric 5. La figura muestra un gráfico de líneas con la siguiente estructura:

- Cada fila muestra los resultados para un lema testigo: “acceder”, “buscar”, “explicar”, “facilitar”, “llegar”, y “pensar”.
- Cada columna es un tipo de representación: atributos manuales y vectores de palabras periodísticos.
- La coordenada x representa el número de iteración en el algoritmo de autoaprendizaje.
- La coordenada y representa el conteo de atributos.
- La línea sólida más oscura representa la media de certeza que el modelo tiene sobre los datos no anotados que está prediciendo.
- El área sombreada, con un color más claro, representa el error estándar de la media de la certeza.

Las tramas que muestro son bastante complejas de seguir y de razonar. Lo que es más notable es que la certeza del algoritmo sobre los ejemplos no vistos es menos uniforme a medida que se añaden más iteraciones. En contraste con lo que asumí originalmente en hipótesis 3.2, el algoritmo muestra una menor certeza sobre nuevos ejemplos.

Es notable que las características artesanales comienzan con una certeza que es más uniforme a través de las iteraciones iniciales. Sin embargo, a medida que las iteraciones van más allá, la certeza es cada vez más variable. Mientras tanto, los vectores de palabras siempre muestran una certeza variable sobre ejemplos no vistos. Cómo evoluciona la certeza a través de las iteraciones en el algoritmo de autoaprendizaje nos da la idea de que el modelo no está convergiendo, al menos no en términos medidos por la certeza.

Está claro que los nuevos ejemplos añaden más incertidumbre al modelo, algo que empeora al añadir más ejemplos. Dado que el algoritmo va a la deriva para clasificar los ejemplos en la clase más frecuente, cabría esperar que aumente la certeza para la clase mayoritaria. Sin embargo, lo que realmente sucede es que la clase mayoritaria acumula muchos ejemplos diferentes. La mayoría de estos ejemplos están mal caracterizados por el modelo, porque la mayoría de sus características son desconocidas para el modelo. Por lo tanto, el clasificador no tiene una gran certeza sobre la clase mayoritaria. En cambio, parece que muchos casos se clasifican en esa clase sólo porque el modelo no tiene suficiente información para clasificarlos en ninguna clase, y simplemente elige la más probable. En suma, la integración de más ejemplos no parece aumentar la certeza porque no están integrados de manera inteligente, sino que se basan únicamente en la probabilidad de la clase mayoritaria. Entonces, no se agregan nuevos ejemplos a las clases minoritarias (y su certeza no aumenta), y los ejemplos agregados a la clase mayoritaria sólo contribuyen a dispersar la certeza.

6.4.3 Hipótesis 3.3

Está claro que los nuevos ejemplos añaden más incertidumbre al modelo, algo que empeora Hipótesis **??** se basa en lo que he comentado en capítulos anteriores sobre la tendencia de un modelo a sobredimensionar los datos de formación. Afirma que la adición automática de nuevos ejemplos ocultos a través del algoritmo de autoaprendizaje ayuda a reducir el sobreequipamiento del modelo.

La Hipótesis se prueba con los resultados reportados por Experiment 6.3. El experimento es similar a lo que hice en el capítulo anterior para ver la evolución del error debido a la varianza utilizando la curva de aprendizaje del modelo. La diferencia principal radica en que en los capítulos anteriores necesité dividir el conjunto de datos etiquetado para simular la adición de nuevos ejemplos a un modelo. Así, en los capítulos anteriores, los nuevos ejemplos añadidos fueron etiquetados como ejemplos que dejé afuera al principio.

El autoaprendizaje me da la posibilidad de explorar cómo los ejemplos reales e invisibles afectan a la tendencia a modificar un modelo medido por la curva de aprendizaje definida como métrica 3 en el capítulo 4.

Sin embargo, hay dos vistas posibles de la curva de aprendizaje en este caso. La primera aproximación consiste en trazar la curva en función de las iteraciones del algoritmo. El segundo enfoque consiste en trazar la curva en función del número de ejemplos utilizados para el entrenamiento.

El primer enfoque se centra más en cómo funciona el algoritmo de autoaprendizaje que en cómo evoluciona el modelo. En cada iteración el número de ejemplos añadidos puede ser alto o bajo. Como no he limitado el número de nuevos ejemplos a añadir, no hay una forma segura de saber cuántos está añadiendo el algoritmo para cada iteración. Lo que los resultados muestran en este caso es más bien cómo los ejemplos añadidos afectan al modelo en esa iteración. El segundo enfoque es más análogo a lo que se muestra en los capítulos anteriores, ya que la trama se realiza basándose en el número de ejemplos del conjunto de datos de formación en lugar de en la iteración en la que se agregaron.

Ambos puntos de vista son útiles a su manera, sin embargo, para la comparación con los métodos anteriores opté por analizar el segundo enfoque, ya que se acerca más a lo que ya he discutido anteriormente para los enfoques supervisado y de embeddings de palabras.

La figura 6.3 muestra el gráfico de la curva de aprendizaje en función del número de ejemplos del conjunto de datos de formación. La estructura de la curva de aprendizaje es la siguiente:

- Cada fila muestra los resultados para un lema testigo: “acceder”, “buscar”, “explicar”, “facilitar”, “llegar”, y “pensar”.
- Cada columna es un tipo de representación: atributos manuales y vectores de palabras periodísticos.

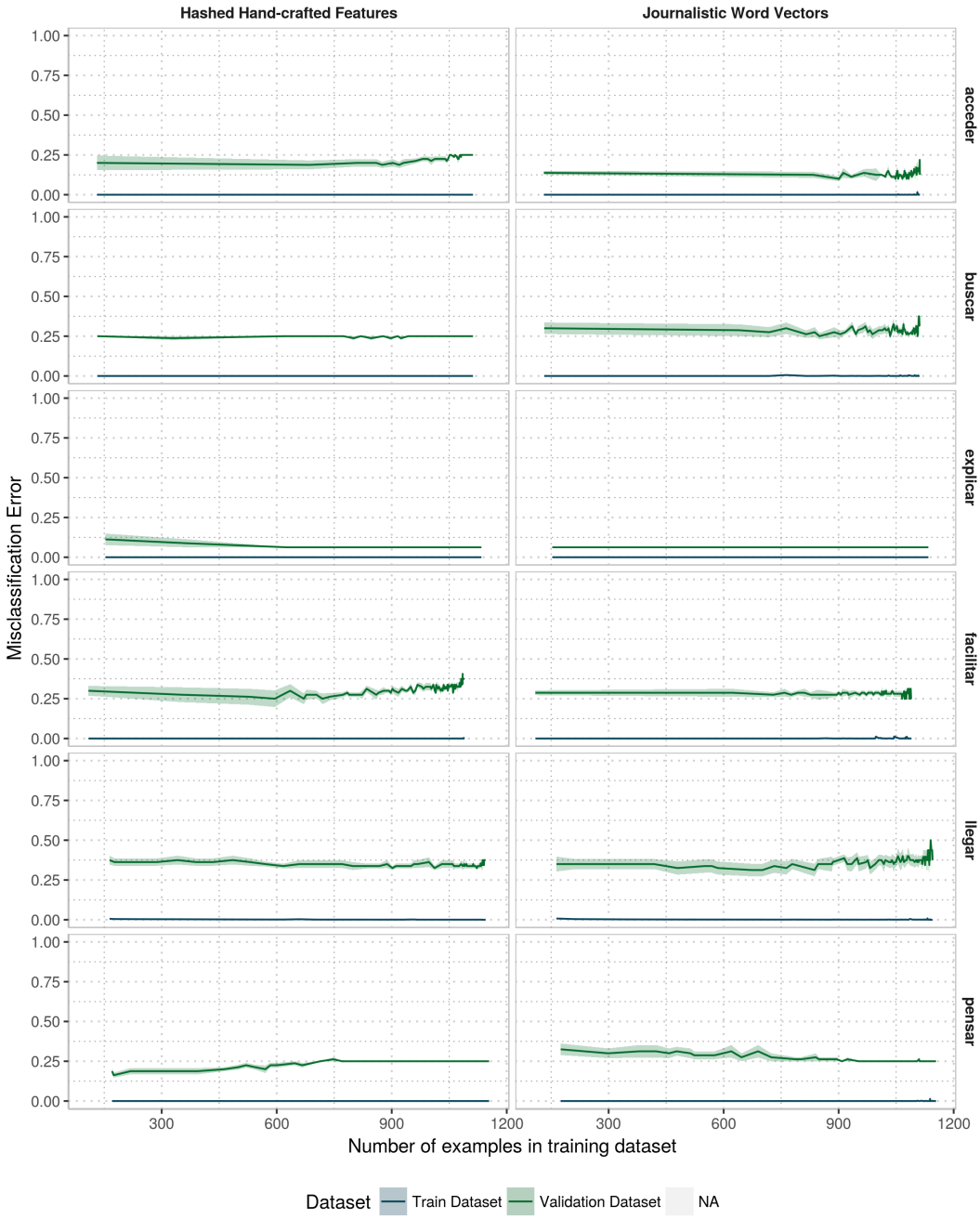


FIGURA 6.3: Curva de aprendizaje como una función del número de ejemplos agrgados por el algoritmo de autoaprendizaje

- La coordenada x representa el número de iteración en el algoritmo de autoaprendizaje.
- Hay dos colores representando los conjuntos de datos: entrenamiento y validación (en este caso es el conjunto de validación).
- La línea sólida más oscura representa la media del error de clasificación a través de las distintas iteraciones de los conjuntos de datos sobre todos los modelos.

- El área sombreada, con un color más claro, representa el error estándar de la media del error de clasificación.

Recordar que estoy limitando mi conjunto de datos no anotados a 1000 ejemplos y por eso ninguno de los gráfico tiene más de 1150 ejemplos totales (i.e. el conjunto de datos supervisados original más los ejemplos anotados automáticamente).

Un primer vistazo a los gráficos muestra que el error de clasificación desciende un poco después de que se añade el primer conjunto de ejemplos anotados automáticamente. Esto es más o menos uniforme para todos los lemas de los primeros incrementos. Si tomo en consideración que estos ejemplos se agregan en las primeras iteraciones, es interesante ver que las primeras iteraciones son las que se suman para mejorar la generalización del modelo. Esto sucede independientemente de la representación utilizada.

También es interesante ver que, en términos generales, el área sombreada, que es el error estándar de la media del error de clasificación errónea, se hace más estrecha a medida que se añaden más ejemplos. Esto significa que el modelo tiene menos variación sobre el conjunto de datos de validación. Esta es una indicación de que el modelo reduce el error debido a la varianza, lo cual es uno de nuestros indicadores de sobreajuste reducido.

Además de eso, es sorprendente cómo el error en el conjunto de datos de validación comienza a ser altamente irregular por cada nuevo ejemplo añadido. Esto ocurre en la mayoría de los casos para los ejemplos añadidos en iteraciones posteriores. Esto puede deberse al hecho de que en las iteraciones finales del algoritmo, el nivel de error tolerado en el conjunto de datos de validación es mayor. De hecho, el error tolerado en el conjunto de datos de validación se incrementa lentamente con el método a medida que avanza la iteración y hasta que alcanza el umbral de parada. Esto es especialmente relevante en el caso de embeddings de palabras. Tal vez sea necesario un análisis más a fondo para el trabajo futuro por ahora.

Esta tendencia podría explorarse más a fondo y sistematizarse hasta el punto de que pueda utilizarse como criterio de parada para el algoritmo. De hecho, la diferencia en la validación y el error de entrenamiento en las iteraciones podría ser un buen criterio de parada.

6.4.4 Hipótesis 3.4

La hipótesis 3.4 está basada en lo que hablé previamente respecto a cobertura del modelo. Para medir la cobertura lo hago contando las características que reconoce el modelo. En el caso de las funciones artesanales, se trata de todas las funciones obtenidas a partir de los datos de la formación (que incluyen palabras, fichas, parte de la palabra, etc., como se explica en la sección 4.3.2). Para el caso de embeddings de palabras, las características son las palabras que rodean al verbo desambiguar y la posición relativa a ese verbo. Hay más de una manera de medir la cobertura de un modelo por sus características. Hipótesis 3.4 está interesado en mostrar que el modelo

aumenta su número de características mediante la adición de nuevos ejemplos, de modo que obtengo el recuento bruto del número total de características que el modelo tiene en el conjunto de datos de formación sobre las iteraciones de autoaprendizaje.

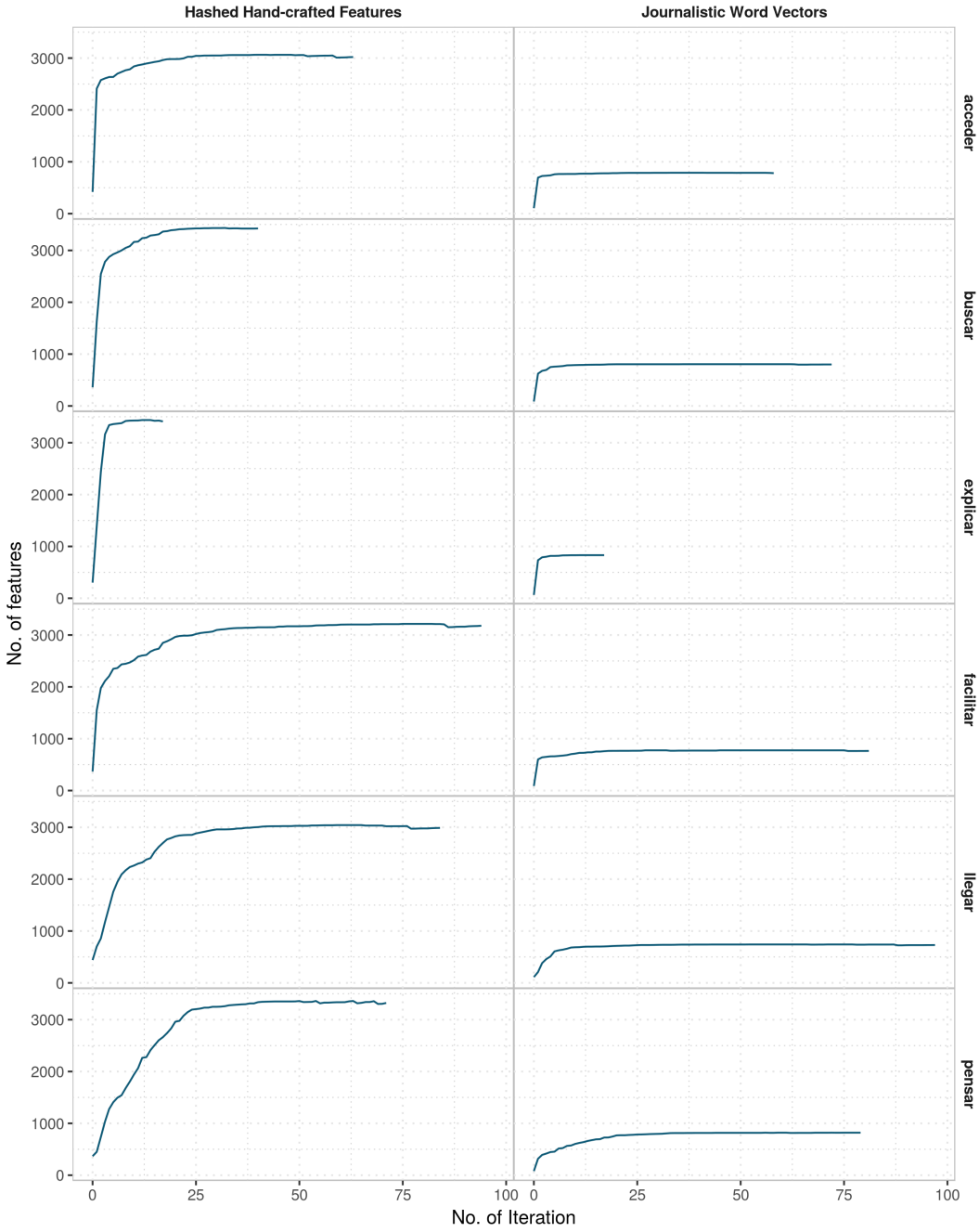


FIGURA 6.4: Crecimiento de atributos en el modelo a lo largo de las iteraciones de autoaprendizaje

La figura 6.4 reporta el crecimiento de las características del modelo a través de las iteraciones de autoaprendizaje. Ese es el número total de características únicas hasta esa iteración obtenida tanto del conjunto de datos supervisados originales como de los datos anotados automáticamente. La figura muestra un gráfico de líneas con la siguiente estructura:

- Cada fila muestra los resultados para un lema testigo: “acceder”, “buscar”, “explicar”, “facilitar”, “llegar”, y “pensar”.
- Cada columna es un tipo de representación: atributos manuales y vectores de palabras periodísticos.
- La coordenada x representa el número de iteración en el algoritmo de autoaprendizaje.
- La coordenada y representa el conteo de atributos únicos.

De la figura resulta visualmente llamativo que la mayoría de las nuevas características se adquieran en las primeras iteraciones del algoritmo de autoaprendizaje. Después de eso, el número de características se reduce. Esto es una indicación de que el algoritmo añade muchos ejemplos en las primeras iteraciones mientras que casi ninguno en las siguientes.

Es interesante notar la diferencia entre los rasgos hechos a mano y los vectores de palabras. Es natural por lo que expliqué anteriormente sobre lo que estoy considerando como características para cada tipo de representación. El conjunto de características para el modelo de embeddings de palabras es mucho más pequeño que para las características hechas a mano. Independientemente de la escala, hay un patrón compartido para ambas representaciones con respecto a la forma en que las características en los modelos crecen.

Estos resultados muestran que el modelo aumenta efectivamente las características asociadas a él, lo que se afirma en Hipótesis 3.4. La hipótesis es aceptada de esa manera. Sin embargo, esto no es necesariamente un aumento en la cobertura del modelo como mostraré en secciones posteriores.

6.4.5 Hipótesis 3.5

En esta sección discuto los resultados con respecto a Hipótesis 3.5. Se espera que esta hipótesis, junto con las dos siguientes en este capítulo, sean falsas (en contraste con las anteriores). Hipótesis 3.5 afirma que el autoaprendizaje ayuda a mejorar el rendimiento del modelo en cada clase de forma uniforme.

Para probar esta hipótesis trabajo de nuevo en los resultados del Experimento **??**. Recuerde que en ese experimento la idea es evaluar el modelo sobre el conjunto de datos de prueba antes y después de ejecutar las iteraciones de autoaprendizaje. Sin embargo, esta vez, en lugar de reportar los promedios definidos en Metric 1, muestro el puntaje F1 para la clase más frecuente, segunda más frecuente y, en algunos casos, tercera más frecuente del lema. Hago esto porque estas son las clases de los lemas simbólicos que no fueron filtrados en el preprocesamiento del corpus. De los 6 lemas simbólicos, 4 de ellos tienen sólo dos sentidos con instancias en el conjunto de datos etiquetado (“acceder”, “buscar”, “explicar”, y “facilitar”), y sólo 2 de ellos tienen 3 sentidos con instancias en el conjunto de datos (“llegar” y “pensar”).

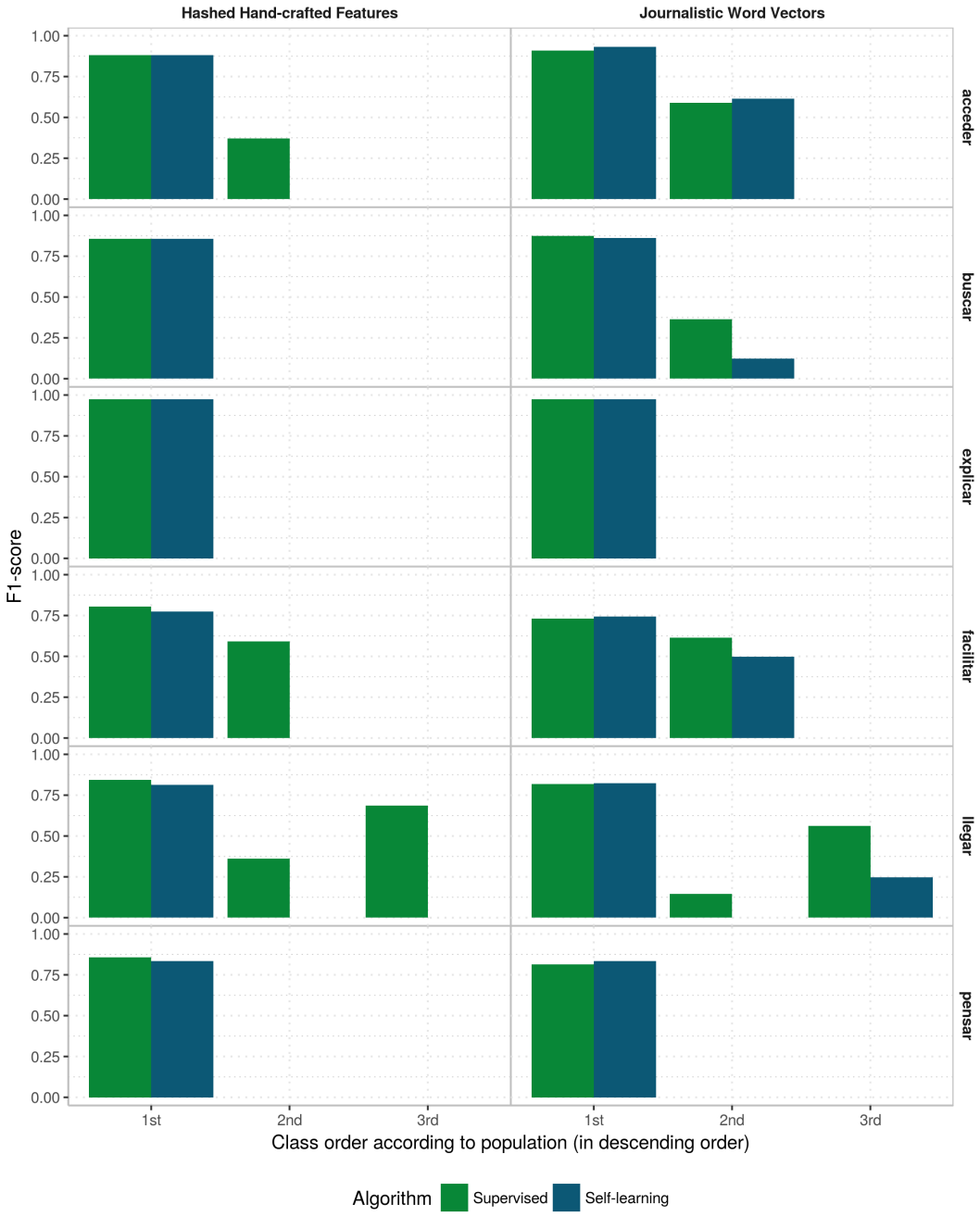


FIGURA 6.5: Comparación del F1-score para cada clase antes y después de las iteraciones del algoritmo de autoaprendizaje para los lemas testigos

La figura 6.5 informa de los resultados del rendimiento en el corpus de pruebas para los algoritmos supervisados (es decir, antes de que se inicie la iteración de autoaprendizaje) y de autoaprendizaje para cada clase (es decir, el sentido) de cada uno de los lemas simbólicos que presenté en la Sección 6.3.1. La parcela es una parcela de bar estructurada de la siguiente manera:

- Cada fila muestra los resultados para un lema testigo: “acceder”, “buscar”, “explicar”, “facilitar”, “llegar”, y “pensar”.

- Cada columna es un tipo de representación: atributos manuales y vectores de palabras periodísticos.
- Cada grupo de barras en cada gráfico representa una clase (i.e. un sentido) para ese lema. Estos están ordenado de acuerdo al número de ocurrencias de la clase en el conjunto de datos.
- Cada gráfico de barra de color distinto dentro de un grupo representa el algoritmo: supervisado (i.e. momento de evaluación de la iteración inicial), autoaprendizaje (i.e. momento de evaluación de la iteración final luego de que autoaprendizaje termine).
- La altura de la barra representa el valor de F1-score para cada clase.

Recuerda de nuevo que sólo las dos últimas filas de los gráficos tienen lemas con 3 sentidos (es decir, "llegar" y "pensar"). Los primeros cuatro lemas pueden mostrar como máximo resultados para dos sentidos.

De la figura se desprende claramente que el algoritmo de autoaprendizaje está afectando negativamente al rendimiento de todas las clases que no son las más frecuentes. El único lema en el que hay una ligera mejora en el rendimiento para una clase que no es la más frecuente es el lemma "acceder" y sólo lo hace con la palabra representación de embeddings.

Este gráfico proporciona más información basada en cada clase en lugar de ocultar todo en una métrica promediada. De él puedo ver que aunque los embeddings de palabras pueden tener menos rendimiento en clases que las más frecuentes, el algoritmo de autoaprendizaje no lo afecta tanto como lo hace para las características hechas a mano. Con características hechas a mano, el puntaje de F1 cae directamente a cero para todas las clases que no sean la más frecuente. Esto es claramente una señal de divergencia con la clase más frecuente cuando se añaden nuevos ejemplos mediante el algoritmo de autoaprendizaje.

Los resultados son claros en este caso y es seguro rechazar Hipótesis 3.5 como pensé originalmente.

6.4.6 Hipótesis 3.6

La siguiente Hipótesis a probar es 3.6. Esta es una extensión de la explicación de la hipótesis anterior, ya que afirma que la representatividad de cada clase en el conjunto de datos se mantiene a través de todas las iteraciones del algoritmo de autoaprendizaje. Por representatividad me refiero al número de clases como proporción del conjunto de datos de formación. Esta Hipótesis se espera que sea falsa y es el resultado de la tendencia del algoritmo de autoaprendizaje a clasificar cada nueva instancia como la clase más frecuente.

El experimento 6.5 se utiliza en esta sección para informar de los resultados del algoritmo de autoaprendizaje con respecto a la representatividad de cada clase a través de iteraciones. El experimento registra la distribución de las clases en el conjunto

de datos de formación en cada iteración. La métrica para medir esto es el conteo proporcional de ocurrencias de cada clase, con respecto a todos los datos disponibles.

Para visualizar estos resultados me centro en dos aspectos que considero relevantes a la hora de considerar la hipótesis. La primera es la cuenta proporcional del número de instancias por clase a lo largo de las iteraciones del algoritmo. En segundo lugar quiero mostrar cuántos ejemplos de cada clase se añaden en cada una de las iteraciones del algoritmo como proporción de todos los ejemplos añadidos en esa iteración.

6.4.6.1 Distribución poblacional de las clases a lo largo de las iteraciones

La figura 6.6 muestra la distribución de la población de las clases a través de las iteraciones de autoaprendizaje del algoritmo. La población de cada clase se representa como la proporción del número total de ejemplos en el conjunto de datos de formación para esa iteración. La parcela es una parcela de superficie apilada que sigue esta estructura:

- Cada fila muestra los resultados para un lema testigo: “acceder”, “buscar”, “explicar”, “facilitar”, “llegar”, y “pensar”.
- Cada columna es un tipo de representación: atributos manuales y vectores de palabras periodísticos.
- La coordenada x representa las iteraciones del algoritmo de autoaprendizaje.
- La coordenada y representa el porcentaje de la población.
- Cada área de un color diferente representa la proporción de ejemplos para cada clase del conjunto de datos. Las clases nuevamente están ordenadas de acuerdo al número de ejemplos en el conjunto de datos supervisado original.

La figura 6.6 muestra el claro predominio de la clase más frecuente en el algoritmo de autoaprendizaje. Para algunos lemas el progreso es un poco menos pronunciado, para otros el algoritmo comienza a clasificar la mayoría de las instancias en la clase más frecuente desde el principio.

Recordemos que el número de clases está muy desequilibrado en el corpus original anotado manualmente. Para hacer esto menos problemático, el corpus que he estado usando en todos estos experimentos ha sobremuestreado las clases minoritarias de modo que el número de ejemplos por clase se distribuye de manera más uniforme. Aún así, los modelos todavía divergen rápidamente hacia la clase mayoritaria.

De esta figura tengo evidencia para rechazar la hipótesis de que la distribución de las clases sea uniforme en las iteraciones de autoaprendizaje. Esta cifra me da una idea de eso. Sin embargo, para ver más claramente lo que sucede con las clases por iteración, la siguiente visualización proporciona una mejor visión de los datos.

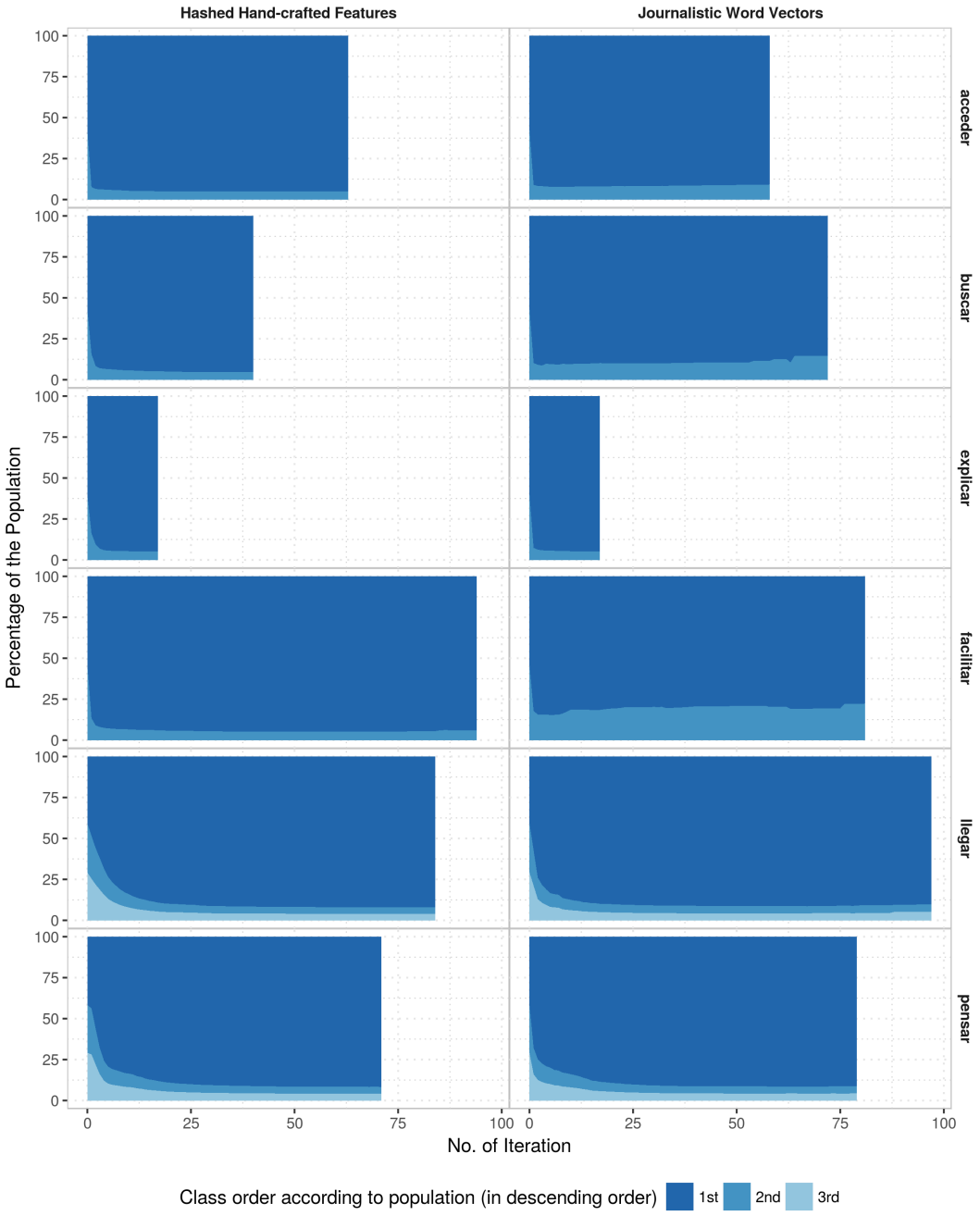


FIGURA 6.6: Distribución poblacional de las clases a lo largo de las iteraciones del algoritmo de autoaprendizaje como una proporción de todo el conjunto de entrenamiento

6.4.6.2 Población agregada por sentido por iteración

La figura 6.7 muestra la proporción de ejemplos añadidos por clase en cada iteración. Es un gráfico de barras apiladas donde cada barra representa el total de ejemplos añadidos en la iteración, y cada color de la barra representa la proporción de clases anotadas automáticamente como tales. La estructura de la parcela es la siguiente:

FIGURA 6.7: Población agregada por sentido en cada iteración del algoritmo de autoaprendizaje como el conteo proporcional de todos los ejemplos agregados en esa iteración

- Cada fila muestra los resultados para un lema testigo: "acceder", "buscar", "explicar", "facilitar", "llegar", y "pensar".
- Cada columna es un tipo de representación: atributos manuales y vectores de palabras periodísticos.
- La coordenada x representa las iteraciones del algoritmo de autoaprendizaje.

- La coordenada y representa el porcentaje de ejemplos anotados automáticamente y agregados al modelo.
- Cada barra representa la distribución de ejemplos agregados in esa iteración. Cada color de la barra representa la clase con la cual fueron anotados los ejemplos.

Esta figura da una mejor idea de lo que está ocurriendo en cada iteración del algoritmo y ayuda a explicar la distribución que se muestra en la Figura 6.6, ya que muestra la proporción de clases añadidas en cada iteración de cada clase.

Puedo ver que el modelo entrenado con características artesanales prácticamente no etiqueta los datos sin etiquetar como algo que no es la clase más frecuente. Esto se suma a lo que se muestra en los párrafos anteriores con respecto a las características artesanales que tienen problemas para generalizarse bien a otros dominios.

Por otro lado, la representación con embeddings de palabras también se ve afectada, como he visto en la figura anterior, con el modelo divergente a la clase más frecuente. Sin embargo, existe una clara tendencia de que el modelo también anote clases distintas a la más frecuente en los ejemplos. Y como ya he comentado en la Sección 6.4.1 los resultados del algoritmo de autoaprendizaje mediante embeddings de palabras fueron muy superiores a los obtenidos para las características artesanales. Esto significa que los embeddings de palabras tienen un buen impacto en un modelo de autoaprendizaje, ya que ayudan al clasificador a generalizar mejor y, por lo tanto, a añadir nuevos ejemplos, incluso de diferentes dominios, al modelo.

De esto y del análisis anterior puedo concluir que la Hipótesis 3.6 es falsa. Claramente no sucede que el modelo de autoaprendizaje esté añadiendo nuevos datos distribuidos uniformemente entre las clases. Sin embargo, los embeddings de palabras ayudan al modelo a añadir datos relevantes y no a etiquetar todo como parte de la clase más frecuente. Esta es una interesante línea de trabajo futuro como un análisis más profundo, quizás con evaluación manual, para ver qué tan bien los embeddings de palabras están ayudando a clasificar ejemplos como parte de las clases menos frecuentes.

6.4.7 Hipótesis 3.7

La última hipótesis que quería rechazar en este capítulo es Hipótesis 3.7. La hipótesis establece que la cobertura dada por las características es uniforme en todas las clases. Recordemos que mostré en la Sección 6.4.4 que el número de características crecía a través de iteraciones del algoritmo de autoaprendizaje, esto fue declarado por Hipótesis 3.4. La sección anterior mostró que el número de clases no se mantiene uniformemente a través de las iteraciones de autoaprendizaje, por lo que es natural pensar que esto también sucederá con las características. Sin embargo, los resultados que mostraré en esta parte son útiles para arrojar más luz sobre los resultados de las secciones anteriores.

Para probar esta hipótesis quiero medir dos cosas: la cuenta bruta de las características para cada una de las clases, y cuán fuertemente una característica está asociada a una clase basada en la información mutua puntual que tiene con esa clase. Estos resultados se obtienen a partir de Experiment 6.4, que informa sobre el número de veces que se produce una característica con una clase en el conjunto de datos de formación para cada iteración del algoritmo de autoaprendizaje. Para reducir el ruido en los datos filtré todos los pares de $(atributos, clases)$ que co-ocurren menos de 2 veces.

6.4.7.1 Conteo de atributos

La primera vista de los datos que quiero mostrar es lo que viene a medir los resultados del Experimento **??** usando el recuento crudo. Esto muestra el número de características únicas que ocurren con cada clase en cada iteración del algoritmo de autoaprendizaje. En esta visión de los resultados, las características pueden estar presentes en dos clases al mismo tiempo, si ocurrieron más de dos veces con esa clase, sin importar cuán fuertemente asociadas estén con la clase.

La figura 6.8 muestra los resultados de contar las diferentes características que ocurren con cada una de las clases en los conjuntos de datos. Ese es el número total de características únicas hasta esa iteración obtenida tanto del conjunto de datos supervisados originales como de los datos anotados automáticamente. La figura muestra un gráfico de líneas con la siguiente estructura:

- Cada fila muestra los resultados para un lema testigo: “acceder”, “buscar”, “explicar”, “facilitar”, “llegar”, y “pensar”.
- Cada columna es un tipo de representación: atributos manuales y vectores de palabras periodísticos.
- La coordenada x representa las iteraciones del algoritmo de autoaprendizaje.
- La coordenada y representa el conteo de atributos únicos en una escala logarítmica. Se seleccionó esta escala porque, en comparación, el número de atributos de la clase más frecuente es de un orden de magnitud mayor que de las clases menos frecuentes y cualquier variación en clases menos frecuentes (e.g. agregar más atributos) se pierde si se lo compara con la clase más frecuente usando una escala lineal.
- Cada línea representa el número de atributos y los colores distintos representan la clase a la que ese atributo pertenece.

La figura muestra que siempre hay una clase para la cual el número de características aumenta mucho más rápidamente y en un número más alto que las otras. Esto es consecuencia de la divergencia del modelo hacia la clase más frecuente. La mayoría de las nuevas características también provienen de las primeras iteraciones del algoritmo y en las siguientes prácticamente no hay más nuevas características añadidas. Esto es

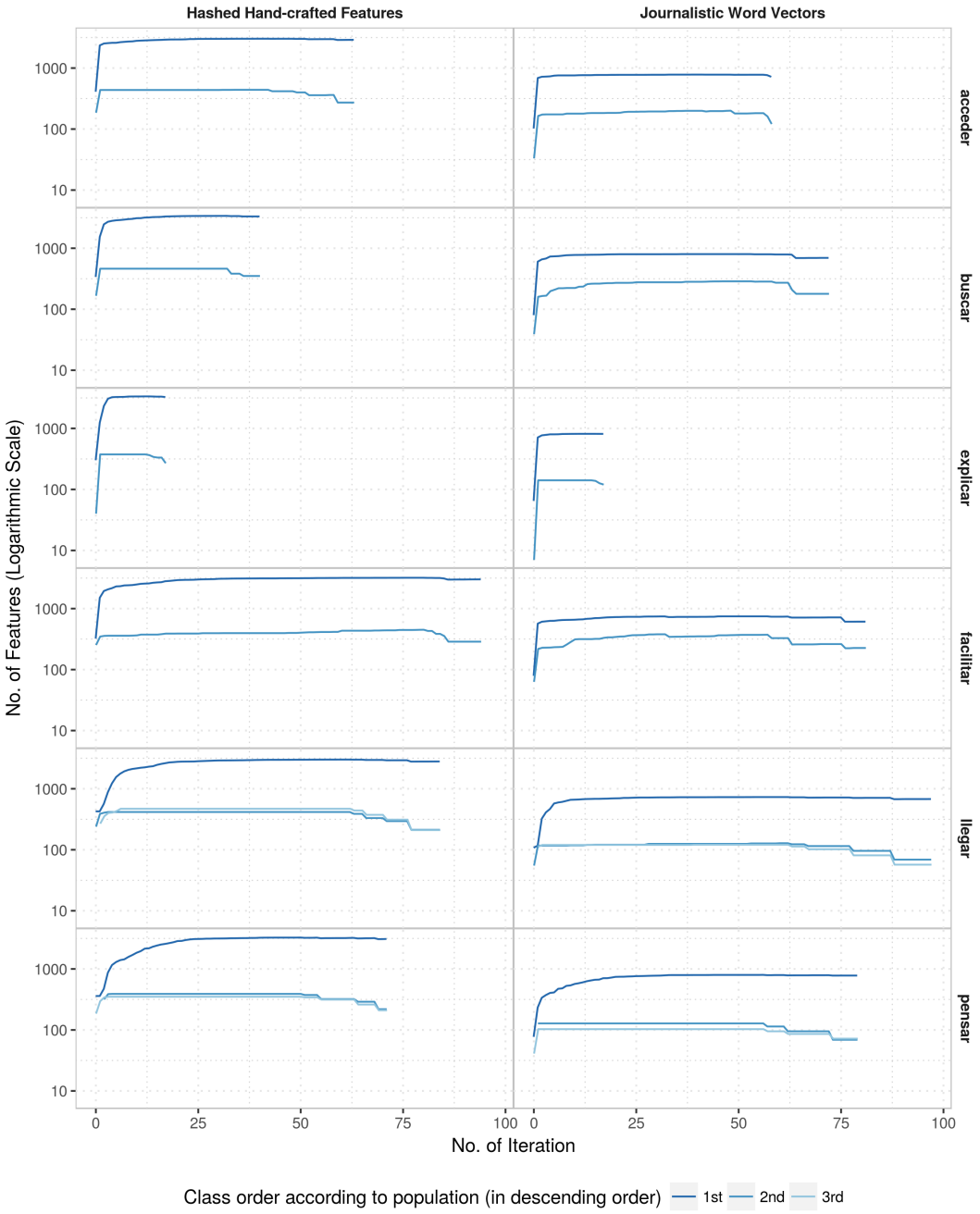


FIGURA 6.8: Crecimiento de atributos por clase a través de las iteraciones de autoaprendizaje (en escala logarítmica)

particularmente notable para las clases menos frecuentes. Como consecuencia de esto, después de algunas iteraciones iniciales el algoritmo se desvía y todas las nuevas instancias se añaden directamente a la clase más frecuente. Así, en las últimas iteraciones del algoritmo no hay nuevas características para las clases menos frecuentes.

Una vez más, la diferencia entre las características hechas a mano y los embeddings de palabras reside en el conjunto de características. Como expliqué anteriormente, el conjunto de características que componen el modelo de embeddings de palabras es mucho más pequeño que el de las características hechas a mano. Sin embargo es

interesante que en algunos de los lemas los rasgos crecen para cada clase en las últimas iteraciones para embeddings de palabras, mientras que no lo hace para rasgos hechos a mano. Esto viene de lo que vi en la sección anterior con respecto a los embeddings de palabras que ayudan al modelo a generalizar mejor y a anotar nuevas clases además de las más frecuentes.

Hay un aumento efectivo en la cobertura, ya que se añaden muchas características al modelo. Sin embargo, estas características pertenecen en su mayoría a una sola clase, la clase mayoritaria. En la siguiente sección mostraré que este aumento en el número de características no implica directamente una mejora del modelo.

6.4.7.2 PMI de los atributos

La segunda vista de los datos que quiero mostrar se basa en los resultados del experimento que mide la correlación entre características y clases. Esto viene dado por la información mutua puntual (PMI) entre las características y las clases asociadas dada por la métrica 4.

La figura **??** muestra la media y el error estándar de la media de la información mutua puntual entre características y clases a través de las iteraciones de autoaprendizaje. Por lo tanto, es mayor cuando las características asociadas a una clase se asocian más fuertemente a la clase. Recordemos que la métrica asocia cada característica a la clase con la que tiene un PMI más alto. La figura muestra un gráfico de líneas con la siguiente estructura:

- Cada fila muestra los resultados para un lema testigo: “acceder”, “buscar”, “explicar”, “facilitar”, “llegar”, y “pensar”.
- Cada columna es un tipo de representación: atributos manuales y vectores de palabras periodísticos.
- La coordenada x representa las iteraciones del algoritmo de autoaprendizaje.
- La coordenada y representa la PMI de los atributos con las clases.
- Cada color representa una clase.
- La línea más oscura representa la media de la PMI para esa clase y el área sombreada representa el error estándar de la media para esa PMI.

A diferencia de la figura anterior, ahora las clases con un menor número de características son las que tienen los valores más grandes en la parcela. Parece que en clases con un número menor de características, éstas se asocian más fuertemente a la clase. Por el contrario, en clases con un mayor número de características, éstas se asocian más débilmente. Parece que la clase más frecuente funciona como un “catch all” para nuevos ejemplos, sobre todo aquellos sin características fuertemente asociadas a cualquier otra clase, y debido a la tendencia del algoritmo a anotar ejemplos en ella. Luego, las características de estos nuevos ejemplos se asocian a esa clase. Sin

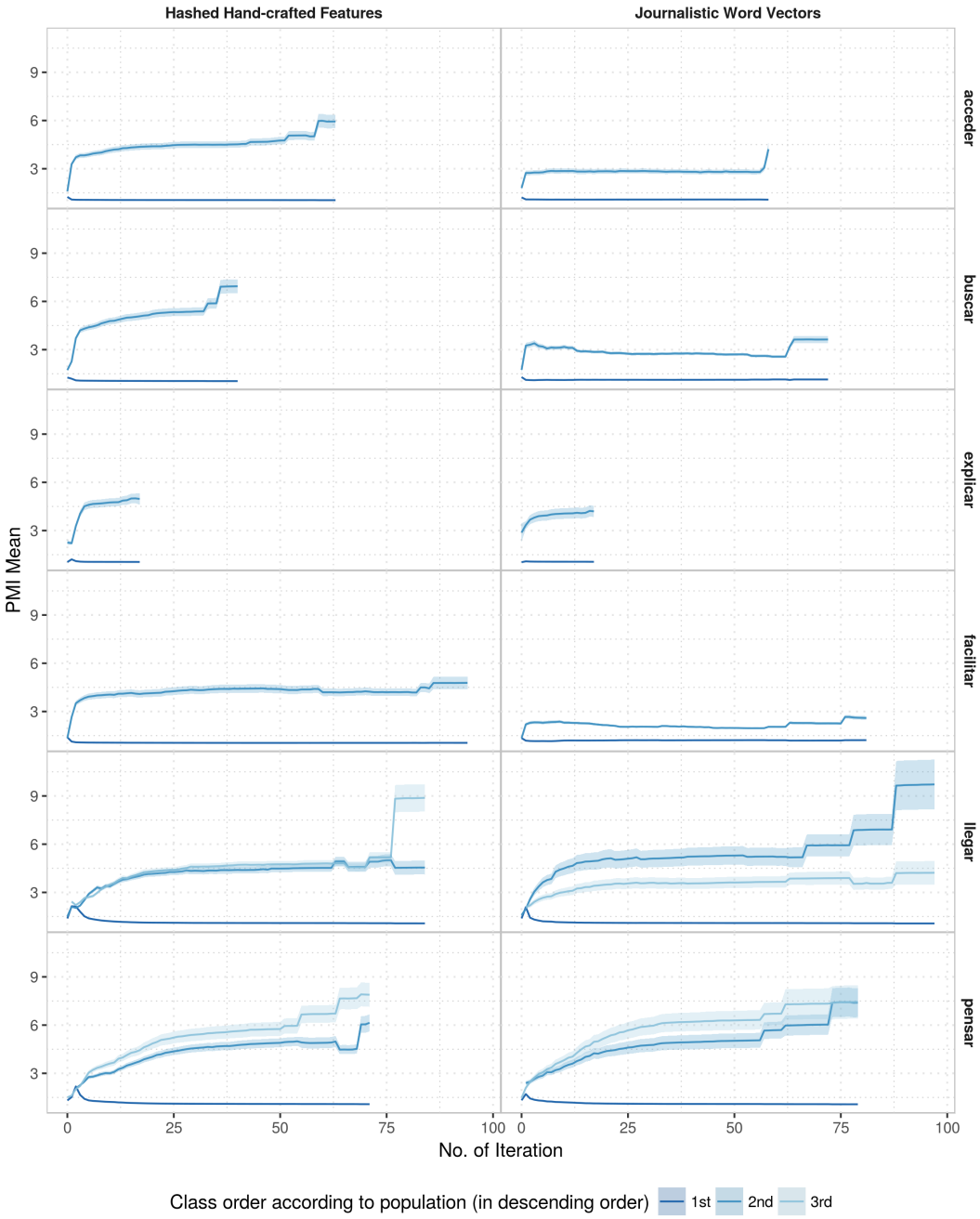


FIGURA 6.9: PMI de los atributos con cada clase a través de las iteraciones de autoaprendizaje

embargo, estas características no están muy relacionadas con esa clase, porque pertenecen a ejemplos que han sido clasificados en la clase por razones débiles, como la probabilidad de la clase o la falta de características caracterizadoras. Esto hace que estas características no sean un buen parámetro para ayudar a discriminar esta clase. Por otro lado, las características de las clases menos frecuentes se vuelven mucho más discriminatorias de dichas clases porque los nuevos ejemplos sólo se clasifican en ellas cuando hay pruebas sólidas de que pertenecen a la clase, es decir, cuando la evidencia es más fuerte que la falta de clasificación en la clase mayoritaria.

Aunque parece que la cobertura del clasificador está aumentando, debido al número de características que lo hacen, a partir de estos datos hay una clara indicación de que no es así porque las nuevas características pertenecen en su mayoría a la clase más frecuente y no contribuyen a aumentar la certeza del modelo, lo que se refleja en una media baja del PMI con la clase más frecuente.

Por otro lado, aquellos rasgos que se asocian a las clases menos frecuentes lo hacen con una fuerte asociación hasta el punto de que a medida que las iteraciones de autoaprendizaje añaden nuevos ejemplos los rasgos se vuelven más discriminatorios de estas clases menos frecuentes. Sin embargo, el número de estas características no aumenta tanto, como se mostró en la sección anterior.

6.4.8 Resumen

Los resultados de la evolución de la certidumbre y de la tendencia al ajuste en el modelo dan la impresión de que las características artesanales tienen menos variación que los embeddings de palabras. Sin embargo, un análisis más detallado muestra que aunque la variabilidad es menor en las características artesanales, la razón detrás de esto es que sólo agrega nuevos ejemplos de una clase particular, la más frecuente. Este no es el caso de los word embeddings.

Los word embeddings muestran una mayor variabilidad en la tendencia a sobredimensionar y la media de la certeza, ya que está añadiendo efectivamente nuevos ejemplos anotados en clases que no sólo son las más frecuentes. Además, la Figura 6.7 muestra que en algunos casos sólo se añaden elementos de las clases menos frecuentes.

6.5 conclusiones

En este capítulo he presentado una aproximación al autoaprendizaje para superar los problemas de sobreequipamiento y cobertura presentados en los capítulos anteriores. Mi hipótesis inicial era que el sobreajuste es causado por el bajo número de ejemplos y que el aumento de dicho número ayuda a reducirlo.

Vi que añadir nuevos ejemplos de forma efectiva ayuda a reducir el sobreajuste en las iteraciones iniciales. Sin embargo, el autoaprendizaje está sesgado a la clase más frecuente. Esto hace que el modelo agregue muchos ejemplos a esa clase, ignorando las clases menos frecuentes. Esto es particularmente notable para las características hechas a mano.

Para sacar esta conclusión definí algunas subhipótesis al principio del capítulo. Los experimentos realizados para probar estas hipótesis me dieron la posibilidad de sacar las siguientes conclusiones.

La hipótesis 3.1 that the performance of the model over a held-out test dataset improves after the self-learning algorithm was shown false for the case of hand-crafted features but I could not obtain conclusive results to reject it with respect to word embeddings. In the latter case the results depended on the lemma, as for some the performance decreased but for others it increases or remains the same. An important

results I could gather from this is that hand-crafted features had a steeper drop in performance than word embeddings, a sign that the representation fits the labeled dataset well but fails to generalize.

La hipótesis 3.2, que afirmaba que la certeza media del modelo sobre ejemplos no vistos en cada iteración aumenta, es rechazada a la luz de los resultados obtenidos. Esto se debe a que hay una clara tendencia de la certeza de la media de que el algoritmo tiene más de datos no vistos a disminuir a través de iteraciones.

La hipótesis 3.3, que afirma que el overfitting se reduce a través de iteraciones de autoaprendizaje, es ligeramente rechazada. El hecho es que el número de ejemplos redujo efectivamente el overfitting en las primeras iteraciones del algoritmo. Sin embargo, como no lo uso como criterio de parada en el algoritmo, cuando comienza a desviarse a la clase más frecuente, la tendencia a sobreajustarse aumenta.

Hipótesis 3.4 afirmaba que la cobertura del modelo aumenta como resultado de aumentar el número de características asociadas a este modelo. De hecho, el número de características asociadas al modelo aumenta a medida que los resultados lo demuestran. Estos resultados son suficientes para aceptar la hipótesis planteada. Sin embargo, como vi en los resultados asociados a Hipótesis 3.7, cuyas conclusiones se encuentran en los siguientes párrafos, estos resultados no son suficientes para mostrar que hay una indicación de que el modelo efectivamente está aumentando la cobertura como resultado de aumentar el número de ejemplos.

Las hipótesis anteriores se formularon con la idea de aceptarlas. Este no fue exactamente el caso. Sin embargo, las otras tres hipótesis de este capítulo se formularon con la idea de que serían rechazadas.

Hipótesis 3.5 establece que el rendimiento del algoritmo de autoaprendizaje sobre todas las clases es mejorado. La hipótesis es rechazada porque sólo el rendimiento de las clases más frecuentes aumenta a través de iteraciones del algoritmo de autoaprendizaje, mientras que el resto de las clases muestran un rendimiento degradado.

La hipótesis 3.6 establece que la representatividad de cada clase en el conjunto de datos se mantiene a través de todas las iteraciones del algoritmo. La hipótesis también es rechazada. Los resultados muestran que los nuevos ejemplos añadidos al modelo pertenecen en su mayoría a la clase más frecuente y que la distribución de las clases está muy sesgada a través de las iteraciones. En particular, las características hechas a mano tienen el peor rendimiento, ya que el modelo sólo añade ejemplos de la clase más frecuente. Los embeddings de palabras tienen un mejor rendimiento ya que el algoritmo también añade ejemplos de las clases menos frecuentes. Esto último tiene relación con lo que sucede en los resultados de la hipótesis anterior, donde hay algunos casos en los que el rendimiento de las clases menos frecuentes no baja a cero como en el caso de las características artesanales.

Hipótesis 3.7 establece que la cobertura de las características se distribuye uniformemente entre las clases. Los resultados obtenidos sirven para probar que la hipótesis es falsa. Esta hipótesis se relaciona con la Hipótesis 3.4 que establece que el número de

características asociadas al modelo aumenta después de las iteraciones de autoaprendizaje, lo cual es cierto, pero no de la manera que el modelo necesita para incrementar efectivamente la cobertura. Esto se debe a que la certeza del modelo no aumenta con el número de características. En un análisis detallado se observa que la mayoría de las características están débilmente asociadas a las clases (en particular a la clase más frecuente). Por lo tanto, su contribución al modelo es pequeña. Por lo tanto, un mayor número de características no implica un mejor modelo, ni siquiera una mejor cobertura, porque no son lo suficientemente discriminatorias. Una posible causa de ello es que, cuando el modelo comienza a sobredimensionarse, la clase más frecuente comienza a cooptar elementos que no forman parte de la clase. Esto hace que el modelo se desvíe y empiece a tener menos certeza sobre los nuevos ejemplos.

Como he visto en los resultados de este capítulo, el mayor problema para el autoaprendizaje es la tendencia a divergir y añadir todo a la clase más frecuente. Una forma de atacar este problema es usar técnicas que ayuden a detener ese sesgo. Hay diferentes maneras de hacerlo. Una es remuestreando los datos. Sólo exploré el sobremuestreo aleatorio de las clases menos frecuentes, que no resultó útil. Otros métodos de remuestreo incluyen el submuestreo aleatorio de la clase más frecuente o una mezcla de ambas.

En el próximo capítulo exploraré una técnica basada en la anotación humana que selecciona ejemplos basados en la alta incertidumbre y se los da a un experto en el dominio para que los anote correctamente. La idea es que estos ejemplos ayudarán al modelo a definir mejor los límites de decisión en la clase más frecuente, algo que el algoritmo de autoaprendizaje no está haciendo bien.

El trabajo futuro de este capítulo incluye el cambio del criterio de parada para que algo se detenga una vez que el modelo se vuelva a ajustar (la diferencia entre error de entrenamiento y error de validación aumenta). Otra línea de trabajo futuro es un análisis más general de todos los lemas, y un análisis más específico de algunos lemas más al lado de las fichas seleccionadas.

También tengo que hacer un análisis más profundo de la certeza que el modelo tiene sobre nuevos ejemplos (es decir, no vistos previamente) y por qué se vuelve tan irregular como las iteraciones del algoritmo van más allá.

Finalmente, como los embeddings de palabras resultaron tener resultados interesantes y, a diferencia de las características hechas a mano, no se desplazaron tan rápido o tan firmemente a la clase más frecuente, sería interesante hacer una evaluación manual y cometer errores.

Capítulo 7

Aprendizaje activo

7.1 Estructura

En el capítulo 4 establecí que el número de ejemplos tiene un impacto directo en el desempeño de un algoritmo supervisado. También expliqué que los corpus pequeños supervisados tienen un impacto directo tanto en la cobertura como en la tendencia a la adaptación de un modelo supervisado. Para atacar esto último, vi en el capítulo 5 que el uso de embeddings de palabras, aunque disminuyendo el rendimiento en la porción de prueba del corpus supervisado, ayudaba en la tendencia a sobreajustar un modelo. En el enfoque de autoaprendizaje, los embeddings de palabras también muestran un rendimiento interesante: incluso si las cifras de precisión gruesa son más o menos comparables para las características hechas a mano y para los embeddings de palabras, estas últimas tienen mejor memoria en las clases minoritarias e incorporan características más informativas.

En el capítulo 6 exploré el autoaprendizaje como un método conjunto semi-supervisado. La idea original de utilizar este método era superar el problema de la cobertura del modelo añadiendo nuevos datos, que se anotan automáticamente, al conjunto de formación. El capítulo también me dio más información sobre los beneficios de usar word embeddings como representación cuando me enfrento a una tarea en la que el dominio cambia. Sin embargo, hay un reto en el método de autoaprendizaje: la tendencia del método a añadir sólo ejemplos de la clase más frecuente.

This chapter introduces another semi-supervised joint learning framework: *active learning*. Like self-learning, active learning consists in expanding the training examples with new data provided by an unannotated corpus. Also, as a wrapper method, it does so by training a model from labeled data and improving that model with new labeled data obtained from the unlabeled corpus.

Este capítulo presenta otro marco de aprendizaje conjunto semi-supervisado: *aprendizaje activo*. Al igual que el autoaprendizaje, el aprendizaje activo consiste en ampliar los ejemplos de formación con nuevos datos proporcionados por un corpus no comentado. Además, como método de envoltura, lo hace capacitando a un modelo a partir de datos etiquetados y mejorando ese modelo con nuevos del corpus no etiquetado.

La diferencia entre el autoaprendizaje y el aprendizaje activo radica en cómo el último método etiqueta los datos no etiquetados. Esto no se hace automáticamente, sino por medio de un oráculo, es decir, un anotador que tiene una alta certeza sobre los

datos que está etiquetando (en términos generales, esto lo hace un anotador humano, más específicamente un experto en la materia).

Pero esta no es la única diferencia, ya que la forma en que se seleccionan las instancias no etiquetadas para la anotación también difiere del método utilizado por el autoaprendizaje. Mientras que el autoaprendizaje se basa en la certeza que el algoritmo tiene sobre los datos para asumir con seguridad que la etiqueta dada es correcta, el aprendizaje activo busca datos que, una vez etiquetados, producirán la mayor mejora sobre el modelo. Estos datos están cerca de los límites de decisión del modelo y generalmente es más difícil distinguirlos de los datos con otras etiquetas.

Una de las técnicas más populares para seleccionar los datos que mejorarán el modelo es *muestreo por incertidumbre*. Lo presenté brevemente en el capítulo 2. En esta técnica, la hipótesis subyacente es que las instancias en las que el clasificador tiene menos confianza son las más cercanas al límite de la decisión y, por lo tanto, las que marcan la mayor diferencia después de haber sido etiquetado.

En nuestro entorno, las instancias que tienen mayor impacto en un modelo son típicamente las que están subrepresentadas en el conjunto de datos etiquetado. Y es precisamente aquí donde el aprendizaje activo supera uno de los problemas del autoaprendizaje: el sesgo de la clase más frecuente. Como el autoaprendizaje es un modelo basado en la certeza, esto implica que agregará mayormente instancias de la clase más frecuente (típicamente aquella sobre la que el modelo tiene mayor certeza, aunque sólo sea por probabilidad). El aprendizaje activo, por otro lado, ayuda al modelo a detectar instancias útiles de las clases subrepresentadas en el conjunto de datos etiquetado y proporciona los medios para agregarlas para una cobertura más amplia.

Además, el aprendizaje activo no sólo añade ejemplos de las clases preexistentes del modelo, sino también de nuevas clases. Recuerde que en capítulos anteriores expliqué que algunas clases del corpus SenSem fueron filtradas porque no tenían suficientes ocurrencias en el conjunto de datos supervisados. Además de esos casos también hay sentidos en el léxico SenSem que no tenían ningún ejemplo en el corpus anotado. Como el corpus etiquetado es específico de un dominio, esto hace que el corpus no tenga ejemplos de algunos de los posibles sentidos. Sin embargo, cuando se aplica el aprendizaje activo sobre un corpus no etiquetado de un dominio general, aparecen ejemplos de estos sentidos en el flujo del algoritmo. Este es un fenómeno común, ya que la certeza que el modelo tiene sobre aquellos ejemplos que nunca antes había visto será baja, por lo tanto será recogida por la técnica de muestreo por incertidumbre. Hay incluso sentidos que directamente no son tomados en cuenta por el recurso, pero se omiten al estar fuera del alcance de esta tesis porque requiere la extensión del léxico SenSem.

En cuanto a lo dicho en el párrafo anterior, hay una observación que hacer sobre el autoaprendizaje. Recordemos que el último capítulo mostró que con el modelo de autoaprendizaje la certeza sobre nuevos ejemplos disminuía en cada iteración. Por lo tanto, es posible que el algoritmo de autoaprendizaje estuviera marcando aquellos

ejemplos sobre los que el modelo tenía menos certeza como parte de la clase más frecuente. Sin embargo, estos ejemplos ni siquiera formaban parte del conjunto de datos inicial etiquetado, ya que las clases de esos ejemplos no están presentes.

Este capítulo evaluará las siguientes hipótesis:

Hipótesis IV *El aprendizaje activo mejora el rendimiento de un modelo supervisado al reducir el sesgo del conjunto de datos inicial que se correlaciona con el ajuste excesivo del modelo supervisado.*

Para aceptar o rechazar tal hipótesis, la divido en otras más específicas:

Subhipótesis 4.1 *La representatividad de la población para cada clase en el conjunto de datos se mantiene a través de todas las iteraciones del algoritmo.*

Subhipótesis 4.2 *El algoritmo de aprendizaje activo aumenta el número de características asociadas al modelo más rápidamente que el autoaprendizaje, lo que es un indicador de un incremento más rápido en la cobertura del modelo. Fin de la subhipótesis.*

Estas hipótesis se evaluarán usando la siguiente estructura:

- El experimento 7.2 registra el número de ocurrencias por cada clase en el conjunto de datos de entrenamiento en cada iteración del algoritmo de aprendizaje activo. La población de las clases se mide por el recuento proporcional con respecto a todo el conjunto de datos de formación. Los resultados de este experimento, mostrados en la Sección 7.4.2, sirven para aceptar la Hipótesis 4.1 que establece que la población de todas las clases se mantiene a lo largo de las iteraciones del algoritmo.

- El experimento 7.3 registra el número de veces que una característica y una clase ocurren juntas en una instancia del conjunto de datos de formación. Mido estos resultados usando dos métricas diferentes. En primer lugar, en comparación con los resultados obtenidos del experimento **??** (que registra los mismos datos pero para el algoritmo de autoaprendizaje), utilizo la cuenta normalizada de características por número de ejemplos por iteración (métrica 6). Es necesario normalizar porque el número de ejemplos añadidos en cada aprendizaje activo es mucho menor que en el autoaprendizaje, por lo que la normalización es necesaria para una comparación significativa. Por otro lado analizo el PMI a lo largo de las iteraciones para las clases y las características asociadas a él (métrica 4). Los resultados se muestran en la sección 7.4.3 y sirven para aceptar la Hipótesis 4.2 que establece que el algoritmo de aprendizaje activo mejora la cobertura del modelo más rápidamente que el autoaprendizaje.

En la sección 7.2 Reviso algunos trabajos previos en aprendizaje activo en general y también aplicados específicamente a desambiguación de sentidos verbales.

En la sección 7.3. Repaso los puntos relevantes que conciernen a la experimentación en el capítulo. La mayor parte de la metodología es muy similar a la de los capítulos

anteriores, en la mayoría de los casos proporciono indicaciones sobre las secciones en las que se describe por primera vez esta metodología. En particular, la sección 7.3.4 explica el detalle fino del algoritmo de aprendizaje activo, por ejemplo, el criterio de parada que utiliza el algoritmo, cómo se seleccionan las instancias para anotar y cómo se lleva a cabo la anotación, así como algunas de las dificultades de la anotación.

La sección 7.4 informa de los resultados de los experimentos y los analiza para aceptar las hipótesis expuestas en el capítulo.

Por último, en la sección 7.5 se extraen las conclusiones de este capítulo, recapitulando las Hipótesis y las implicaciones de aceptarlas o rechazarlas según las evidencias recogidas en los resultados. Establece las deficiencias de los métodos explorados en este capítulo y lo que quiero lograr en el siguiente. Termina esbozando el trabajo futuro.

7.2 Trabajo relevante

El aprendizaje activo es una tarea para el aprendizaje semi-supervisado, basado en la idea clave de que si se permite que el algoritmo de aprendizaje elija los datos de los que aprende, funcionará mejor con menos entrenamiento. Una encuesta muy completa en el área es la realizada por Settles [Settles, 2009]. Cubre todos los temas principales en el área de aprendizaje activo.

Un tema particularmente importante en el aprendizaje activo es la estrategia para seleccionar las instancias a presentar al oráculo. Se han estudiado diferentes estrategias, siendo el *muestreo por incertidumbre* el más intuitivo y popular [Lewis and Catlett, 1994] y el utilizado en este capítulo. Para otras estrategias refiérase a la sección 2.2.3 del capítulo 2.

Uno de los principales problemas a la hora de aplicar el aprendizaje activo a las tareas de procesamiento del lenguaje natural es el desequilibrio de clases, algo común en la zona. En estos casos, el muestreo de incertidumbre no siempre resulta ser la mejor manera de seleccionar los datos dadas algunas dificultades que surgen como explicaré en la sección 7.3.4.3. Hay algunos enfoques para lidiar con ello, pero nada definitivo.

En particular, Dligach y Palmer [Dligach and Palmer, 2011] aplicaron el aprendizaje activo a desambiguación de sentidos verbales. Exploraron los beneficios de usar un modelo de lenguaje no supervisado para seleccionar ejemplos semilla como un corpus inicial para un enfoque de aprendizaje activo iterativo.

7.3 Metodología

Este capítulo explora el uso del aprendizaje activo con un anotador humano para expandir un modelo entrenado con datos etiquetados. Al igual que el algoritmo de autoaprendizaje del capítulo anterior, el aprendizaje activo es también *algoritmo envolvente*. El algoritmo utiliza un clasificador supervisado para seleccionar instancias de los datos no etiquetados y se las da a un humano para que las verifique.

El objetivo de este capítulo es mostrar las tendencias que he encontrado utilizando el aprendizaje activo. No presento un análisis exhaustivo del aprendizaje activo, ya que requiere la anotación de recursos a través de expertos humanos que están fuera de las posibilidades de investigación de esta tesis. La idea es analizar los resultados preliminares que pueden ayudar como un comienzo inicial para el trabajo futuro utilizando más recursos.

7.3.1 Recursos

En este capítulo sigo usando los conjuntos de datos en los que he estado trabajando hasta ahora: el corpus SenSem como corpus etiquetado y SBWCE como corpus no etiquetado.

Analizar el aprendizaje activo fue la razón principal para reducir el análisis a algunos de los lemas del léxico. Los lemas seleccionados sobre los que hago mi análisis son: "explicar", "pensar", "acceder", "llegar", "buscar" y "facilitar". Una explicación detallada de esto se hace en la sección 6.3.1, por favor refiérase a ella para más información.

El corpus SenSem se utiliza como la semilla anotada en la que se inicia el algoritmo de aprendizaje activo. Para una descripción detallada del recurso, consulte la sección 4.3.1.1.

La fuente de datos no etiquetada para nuevas instancias a dar al experto humano para anotar manualmente se toma del corpus SBWCE. Por favor, consulte la sección 5.3.1.2 para más detalles sobre el corpus. El corpus es el preprocesado con Freeling obtenido por la metodología descrita en la sección 6.3.1.2. Por favor refiérase a esa sección para más detalles sobre cómo se hace esto.

7.3.2 Atributos

Los atributos utilizados en esta parte son las mismas que en los capítulos anteriores. Ambas tipos de atributos, manuales con hashing trick y word embeddings siguen lo que se ha discutido en todos los capítulos anteriores.

Para obtener más información sobre los detalles de las características artesanales utilizadas en este capítulo, consulte la sección 4.3.2. Para información detallada sobre cómo lidiar con la expansión de las características dadas por los nuevos ejemplos añadidos al modelo, por favor refiérase a la sección 6.3.2.1.

Los word embeddings son los formados a partir del corpus periodístico descrito en detalle en la sección 5.3.1.3. El método para combinarlos en una instancia es la concatenación de vectores de palabras descrita en la sección 5.3.2.2.

7.3.3 Clasificadores

Al igual que en el autoaprendizaje, el aprendizaje activo es un método de envoltura que se construye alrededor de un clasificador supervisado. El clasificador seleccionado es el mismo que en los capítulos anteriores. Utilizo un perceptrón multicapa con tres capas de tamaño 500, 250 y 100 cada una.

7.3.4 Algoritmo de aprendizaje activo

El algoritmo de aprendizaje activo sigue la misma estructura que el algoritmo de autoaprendizaje descrito en la sección 6.3.4, tomando como entrada casi los mismos parámetros iniciales:

- Un conjunto de entrenamiento etiquetado para entrenar el modelo supervisado inicial.
- Un conjunto de evaluación etiquetado para registrar el desempeño del modelo antes y después de terminadas las iteraciones del algoritmo de autoaprendizaje.
- Un conjunto de datos no etiquetados para obtener los nuevos datos a anotar manualmente.
- Un clasificador probabilístico para usar en el procedimiento de anotación.
- Un número máximo de elementos para anotar en cada iteración.
- Una tolerancia de error como criterio de parada.

Las diferencias fundamentales en este caso residen en los siguientes puntos: (i) cómo el modelo selecciona las instancias para etiquetar a partir del conjunto de datos sin etiquetar, (ii) cómo se anotan esas instancias, (iii) cómo el modelo decide si se deben incluir nuevas instancias en el corpus de capacitación, (iv) cómo termina el algoritmo.

7.3.4.1 Selección de instancias y anotación

En cada iteración del algoritmo, una vez que el clasificador es entrenado con los ejemplos recolectados hasta el momento (ambos parte del conjunto inicial de datos de semillas o de las anotaciones del oráculo) el algoritmo aplica el clasificador a todo el conjunto de datos no anotados. A partir de la anotación automática de ese conjunto de ejemplos sin etiquetar, el algoritmo toma aquellos de los que el clasificador está menos seguro y los presenta al oráculo para su anotación manual. Esto se denomina *muestreo por incertidumbre*, es decir, seleccionar aquellos casos en los que la certeza del clasificador es baja. Se supone que esas instancias están cerca de los límites de decisión del modelo. La intuición detrás del uso de este método en lugar de dar al azar los ejemplos de oráculo para anotar es que estos ejemplos tendrán el mayor impacto en el modelo final, es decir, el modelo mejorará su precisión al máximo. Esto se debe a que el clasificador gana confianza en esas clases e instancias sobre las que tiene menos información en el corpus de semillas etiquetadas. Esta es una forma de optimizar el aprendizaje con una cantidad determinada de datos de formación, mejorando la calidad (no la cantidad) de los ejemplos de formación.

A diferencia del autoaprendizaje, que selecciona todas las instancias por encima de un umbral, en el aprendizaje activo hay un número máximo de instancias posibles. Se

da como parámetro inicial del algoritmo. Este número es seleccionado empíricamente y debe ser lo suficientemente grande para hacer una diferencia en el nuevo modelo, pero no demasiado grande para que el oráculo dedique demasiado tiempo a anotar, ya que el aprendizaje activo ofrece una forma inteligente de minimizar el esfuerzo de anotación. Después de algunos experimentos, se seleccionó el número de 10 ejemplos por iteración como base para los experimentos en este capítulo.

El oráculo anota los ejemplos seleccionados por el algoritmo de aprendizaje activo, que luego se añaden al modelo. Sin embargo, hay un problema, como expliqué antes: el corpus inicial anotado puede no tener una etiqueta correcta para algunos ejemplos porque no fue considerado originalmente cuando se diseñó el léxico. En esos casos, para reducir el número de posibilidades a considerar y como está fuera del alcance de esta tesis extender el recurso, decidí ignorar ejemplos pertenecientes a un sentido no presente en SenSem.

7.3.4.2 Validación del modelo y parada temprana del algoritmo

Una vez que las instancias son anotadas por el oráculo, el siguiente paso es, como en el autoaprendizaje, añadirlas como nuevos ejemplos de entrenamiento. Sin embargo, el algoritmo comprueba primero el impacto de añadir estos nuevos ejemplos al modelo, evaluando el rendimiento del modelo con el corpus de entrenamiento más grande.

Para el autoaprendizaje, esto se hizo utilizando un conjunto de datos de validación extraído del conjunto de datos original. Al principio pensé en utilizar la validación cruzada sobre el conjunto de datos de formación en autoaprendizaje, pero finalmente abandoné la idea. La tendencia del algoritmo de autoaprendizaje a añadir todo como parte de la clase más frecuente puede haber introducido sesgos en ese método de validación.

Sin embargo, para el aprendizaje activo esto no es un problema. Como la anotación no se deja al clasificador sino a un humano, la tendencia a anotar todo como la clase más frecuente no prevalece tanto. Decidí entonces usar la validación cruzada sobre los ejemplos de capacitación y ver la media del error de clasificación errónea obtenido de ella como la métrica para validar el impacto de los nuevos ejemplos de capacitación en el modelo.

La razón principal para utilizar la validación cruzada sobre el conjunto de datos de formación y no un conjunto de datos de validación como en el autoaprendizaje es porque, a diferencia del autoaprendizaje, este algoritmo puede añadir ejemplos de clases no vistas por el modelo hasta ahora. Recuerde que algunos sentidos fueron filtrados del corpus SenSem porque no tenían suficientes instancias disponibles. Además, algunos de los sentidos del léxico no tienen ningún ejemplo. Bien, estos sentidos ocurren en un corpus más grande como SBWCE, y como el clasificador no tendrá suficiente información inicialmente sobre esos sentidos, es probable que se escojan ejemplos de esos para darle al oráculo usando la técnica de muestreo de incertidumbre.

Como pueden añadirse nuevos ejemplos de clases no vistas a los datos de formación, esto puede dar lugar a que el corpus de validación no refleje el rendimiento real del

modelo. La razón es que el corpus de validación no tiene ninguna información sobre las nuevas clases no vistas que el modelo está añadiendo.

Una vez que el algoritmo calcula el error de clasificación a partir de la validación cruzada sobre el conjunto de datos de formación, aplica el mismo principio que el algoritmo de autoaprendizaje, lo compara con el error mínimo obtenido hasta el momento y da una tolerancia de un máximo del 10 % menos que el azar. Si el error en el nuevo modelo es mayor, el algoritmo descarta el modelo y devuelve el último modelo obtenido hasta el momento. Como requiere anotación humana, el algoritmo también proporciona la opción para que el humano detenga las iteraciones cuando quiera.

7.3.4.3 Dificultades para la anotación manual

La anotación de los ejemplos seleccionados por el algoritmo sobre la base de una gran incertidumbre presentaba dos dificultades principales: la ambigüedad y la cobertura de las clases disponibles en el recurso.

Dado que el inventario de los sentidos es dado por el ser humano, es arbitrario, moldeado por el ser humano creando el recurso y el objetivo final que dicho recurso tiene. Esto significa que dependiendo de estas variables, el inventario de los sentidos puede ser de grano grueso o fino. Esto, por supuesto, afecta a las posibilidades de anotar, ya que un inventario más grueso puede caer en el problema de tener sentidos que son adecuados para más de un sentido. Como el algoritmo seleccionó aquellas instancias que están cerca de los límites de decisión, un problema común es dar ejemplos de oráculo que son ambiguos incluso para que el oráculo pueda discernir.

Además del problema de tener sentidos con límites de decisión borrosos, que se superponen entre sí, el otro problema que puede tener el recurso es la falta de cobertura. Por lo tanto, hay algunos ejemplos que no son adecuados para ninguno de los sentidos que se dan en el recurso. Esto es particularmente común ya que una parte importante del corpus de la SBWCE consiste en libros escritos en español antiguo (es decir, los compilados en la iniciativa Wikisource). Así, hay algunos sentidos que cayeron en desuso y no fueron considerados en la construcción del recurso.

7.3.5 Experimentos

Hay tres experimentos importantes que hice en este capítulo: uno para comparar el rendimiento del algoritmo de aprendizaje activo con el algoritmo de autoaprendizaje, y otros dos para probar las hipótesis.

First, Experiment 7.1 is very similar to Experiment 6.1 in Chapter 6. The experiment is not aimed at testing any of the hypotheses of this chapter but rather as a comparison with the previous chapter. It basically tests the active learning algorithm and how it affects the performance of the supervised classifier over the held-out test data.

Primero, el esperimento 7.1 es muy similar al experimento 6.1 del capítulo 6. El experimento no pretende probar ninguna de las hipótesis de este capítulo, sino más

bien una comparación con el capítulo anterior. Básicamente prueba el algoritmo de aprendizaje activo y cómo afecta el rendimiento del clasificador supervisado sobre los datos de las pruebas retenidas.

Experimento 7.1.

7.1a Entrenar un modelo con el conjunto de datos de entrenamiento supervisado.

7.1b Evaluar el modelo sobre el conjunto de datos de evaluación.

7.1c Run the active algorithm until it stops.

7.1d Correr el algoritmo de active learning hasta que pare.

7.1e Evaluar el modelo obtenido luego del algoritmo de active learning sobre los datos de evaluación.

El experimento 7.2 es el primer experimento para probar una hipótesis en este capítulo. El experimento consiste en evaluar la representatividad de la población para cada clase a través de las iteraciones de aprendizaje activo. Se hace para probar la hipótesis 4.1, que establece que esta representatividad se mantiene a través de todas las iteraciones del algoritmo. Este experimento se basa en el experimento 6.5 del capítulo anterior.

Experimento 7.2.

7.2a Registrar el número de veces que cada clase ocurre en el conjunto de datos supervisado.

7.2b Correr una iteración del algoritmo de active learning.

7.2c Registrar el número de veces que cada clase ocurre en el nuevo conjunto de entrenamiento compuesto por los datos supervisados originales y los nuevos datos anotados.

7.2d Repetir el paso previo para cada iteración del algoritmo.

Finalmente, para probar la hipótesis 4.2 con respecto al número y asociación de características a las clases, existe el experimento 7.3, que se basa en el experimento 6.4 en el capítulo anterior. El experimento registra el número de veces que cada tupla de clases y características aparecen en el conjunto de datos.

Experimento 7.3.

7.3a Registrar el número de veces que se produce una tupla $(clase, atributo)$ para cada clase y cada atributo del conjunto de datos supervisados.

7.3b Correr una iteración del algoritmo de aprendizaje activo.

7.3c Registrar el número de veces que se produce una tupla $(clase, atributo)$ para cada clase y cada atributo del conjunto de datos de entrenamiento compuesto por los datos supervisados y las instancias anotadas automáticamente.

7.3d Repetir el paso previo para cada iteración del algoritmo.

7.3.6 Métricas

Para reportar los resultados del experimento 7.1 utilizo el F1-score para cada clase en el conjunto de datos de prueba de los lemas testigo, para cada uno de los tres algoritmos: aprendizaje supervisado, autoaprendizaje y aprendizaje activo.

Para la hipótesis 4.2, que compara la asociación de atributos que proviene de las iteraciones de aprendizaje activo, me centro principalmente en dos métricas. Una es la métrica 4, que calcula el PMI entre atributos y clases y asocia cada atributo a la clase con la que tiene un PMI más alto. La otra es la métrica 6 que es el recuento normalizado de características mediante ejemplos:

Métrica 6.

6a Contar el número total de atributos de una iteración para un algoritmo de aprendizaje conjunto.

6b Normalizar el conteo por el número total de ejemplos agregados en esa misma iteración.

7.4 Análisis de resultados

En esta sección se reportan los resultados obtenidos por el experimento descrito anteriormente, con las métricas mencionadas. Hay una observación importante que hacer sobre estos resultados. Este es un trabajo exploratorio y preliminar. El aprendizaje activo, ya que requiere recursos de anotación de un experto en el dominio, es muy costoso. Para esta tesis, la cantidad de esfuerzo que se podía dedicar a la anotación era muy limitada, por lo que el aprendizaje activo se presenta aquí como una prueba de concepto, mostrando tendencias en lugar de resultados claramente definidos. Sin embargo, los resultados son interesantes de mostrar y analizar. Una vez más, la visualización y las métricas utilizadas aquí pueden oscurecer algunos aspectos de los resultados a favor o en contra. Intentaré dar el análisis más imparcial sobre estos datos.

7.4.1 Comparación de desempeño para supervisado, autoaprendizaje y aprendizaje activo

Antes de ahondar en el análisis de las hipótesis de este capítulo, quiero hacer una revisión rápida de cómo se comparan los dos algoritmos diferentes vistos hasta ahora

(autoaprendizaje y aprendizaje activo) en términos de rendimiento con el algoritmo supervisado. Esto se hace para establecer una base común para todos los algoritmos de aprendizaje conjunto.

Estos resultados se obtienen a partir de Experiment 7.1, que mide el rendimiento del modelo utilizando el F1-score por clase, mostrando el rendimiento para la clase más frecuente, la segunda más frecuente y la tercera más frecuente del lema. Hago esto porque estas son las clases de los lemas simbólicos que no fueron filtrados en el preprocesamiento del corpus. De los 6 lemas simbólicos, 4 de ellos tienen sólo dos sentidos con instancias en el conjunto de datos etiquetado ("acceder", "buscar", "explicar", y "facilitar"), y sólo 2 de ellos tienen 3 sentidos con instancias en el conjunto de datos ("llegar" y "pensar").

La figura 7.1 muestra la macro de la puntuación F1 y la media ponderada para el aprendizaje supervisado, autoaprendizaje y activo sobre el conjunto de datos de la prueba. En este caso, "supervisado" es la evaluación del modelo en la iteración inicial de cualquiera de los algoritmos de aprendizaje conjunto (ya que es el mismo para ambos, es decir, sólo usando los datos etiquetados manualmente). Las barras de autoaprendizaje/aprendizaje activo representan el rendimiento del modelo sobre el conjunto de datos de prueba después de terminar las iteraciones del algoritmo correspondiente. La estructura del gráfico es la siguiente:

- Cada fila muestra los resultados para un lema testigo: "acceder", "buscar", "explicar", "facilitar", "llegar", y "pensar".
- Cada columna es un tipo de representación: atributos manuales y vectores de palabras periodísticos.
- Cada grupo de barras en cada gráfico representa una clase (i.e. un sentido) para ese lema. Estos están ordenado de acuerdo al número de ocurrencias de la clase en el conjunto de datos.
- Cada gráfico de barra de color distinto dentro de un grupo representa el algoritmo: supervisado (i.e. momento de evaluación de la iteración inicial), autoaprendizaje (i.e. momento de evaluación de la iteración final luego de que autoaprendizaje termine) y aprendizaje activo (i.e. momento de evaluación de la iteración final luego de finalizado aprendizaje activo).
- La altura de la barra representa el valor de F1-score para cada clase.

Recordar de nuevo que sólo las dos últimas filas del gráfico representan lemas con 3 sentidos (es decir, "llegar" y "pensar"). Los primeros cuatro lemas pueden mostrar como máximo resultados para dos sentidos.

En general, el aprendizaje activo funciona mejor que el autoaprendizaje e incluso mejor que el supervisado en algunos casos, tanto en las clases más frecuentes como en las menos frecuentes. Puede ser que el aprendizaje activo funcione mejor que el

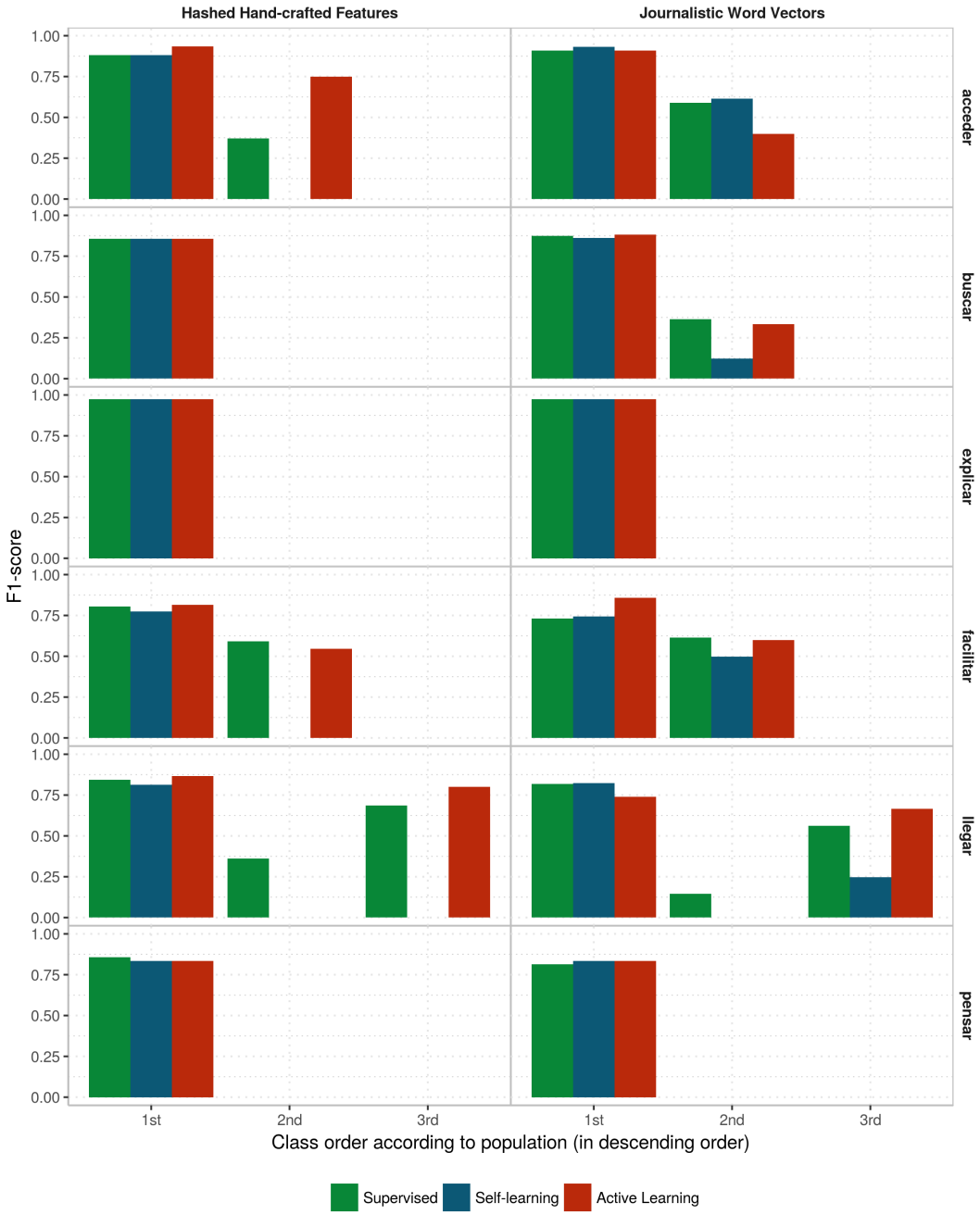


FIGURA 7.1: Comparación de los promedios macro y ponderado del F1-score para aprendizaje supervisado, autoaprendizaje y aprendizaje activo

aprendizaje supervisado simplemente porque la cantidad de ejemplos anotados aumenta. Sin embargo, la cantidad de ejemplos anotados también aumenta en el caso del autoaprendizaje, pero no influye en el rendimiento. Esta es una señal clara de que la selección de ejemplos para dar al oráculo a anotar va por buen camino, porque tiene un impacto visible en el rendimiento del modelo resultante, más allá del mero incremento en el número de ejemplos anotados.

Una vez más, al igual que en el capítulo anterior, los embeddings de palabras

siguen mostrando un mejor rendimiento general que las características artesanales: los sentidos minoritarios están mejor representados, sin una pérdida importante de rendimiento en los sentidos mayoritarios. En combinación con el aprendizaje activo, los embeddings de palabras se vuelven particularmente útiles, ya que caracterizan adecuadamente los sentidos de las minorías.

Ahora que tengo una comparación de base de los dos algoritmos, tengo fuerte evidencia de que el aprendizaje activo es mejor para el desempeño del modelo. Para comprobar por qué es así, profundizaré en las hipótesis.

7.4.2 Hipótesis 4.1

Esta sección prueba la hipótesis 4.1. La hipótesis establece que la representatividad de la población de cada clase en el conjunto de datos de entrenamiento se mantiene a través de todas las iteraciones del algoritmo. Recordemos que para el algoritmo de autoaprendizaje encontré evidencia para aceptar Hipótesis 3.6, que establece lo opuesto de Hipótesis 4.1.

Más precisamente, establecí que el hecho de que el autoaprendizaje no estuviera representando adecuadamente a todas las clases era la causa de que el autoaprendizaje degradara el rendimiento de las clases menos frecuentes, como se afirma en la hipótesis 3.5 en el capítulo anterior. Como he visto en el apartado anterior, este no es el caso del aprendizaje activo, ya que mejora el rendimiento de las clases que no son las más frecuentes.

Muestro los resultados de la hipótesis 4.1 para evaluar si la razón detrás de la mejora en el rendimiento es efectivamente una mejor representación de las clases minoritarias con el algoritmo de aprendizaje activo.

Para probar la hipótesis mido los resultados obtenidos después de hacer el experimento 7.2 que registra el número de instancias que cada una de las clases tiene en el conjunto de datos de entrenamiento después de cada iteración de aprendizaje activo.

Una vez más, como en el capítulo anterior, para visualizar estos resultados utilizo dos técnicas para mostrar lo que, para mí, es relevante en la comprobación de la Hipótesis: el recuento proporcional del número de instancias por clase a lo largo de las iteraciones, y el número proporcional de instancias añadidas para cada clase por iteración. Estos dos resultados permitirán aceptar la hipótesis 4.1.

7.4.2.1 Distribución poblacional de las clases a través de las iteraciones

La figura 7.2 muestra la distribución de la población de las clases a través de las iteraciones de aprendizaje activas. La población de cada clase se representa como la proporción del número total de ejemplos en el conjunto de datos de formación para esa iteración. La parcela es una parcela de superficie apilada que sigue esta estructura:

- Cada fila muestra los resultados para un lema testigo: “acceder”, “buscar”, “explicar”, “facilitar”, “llegar”, y “pensar”.

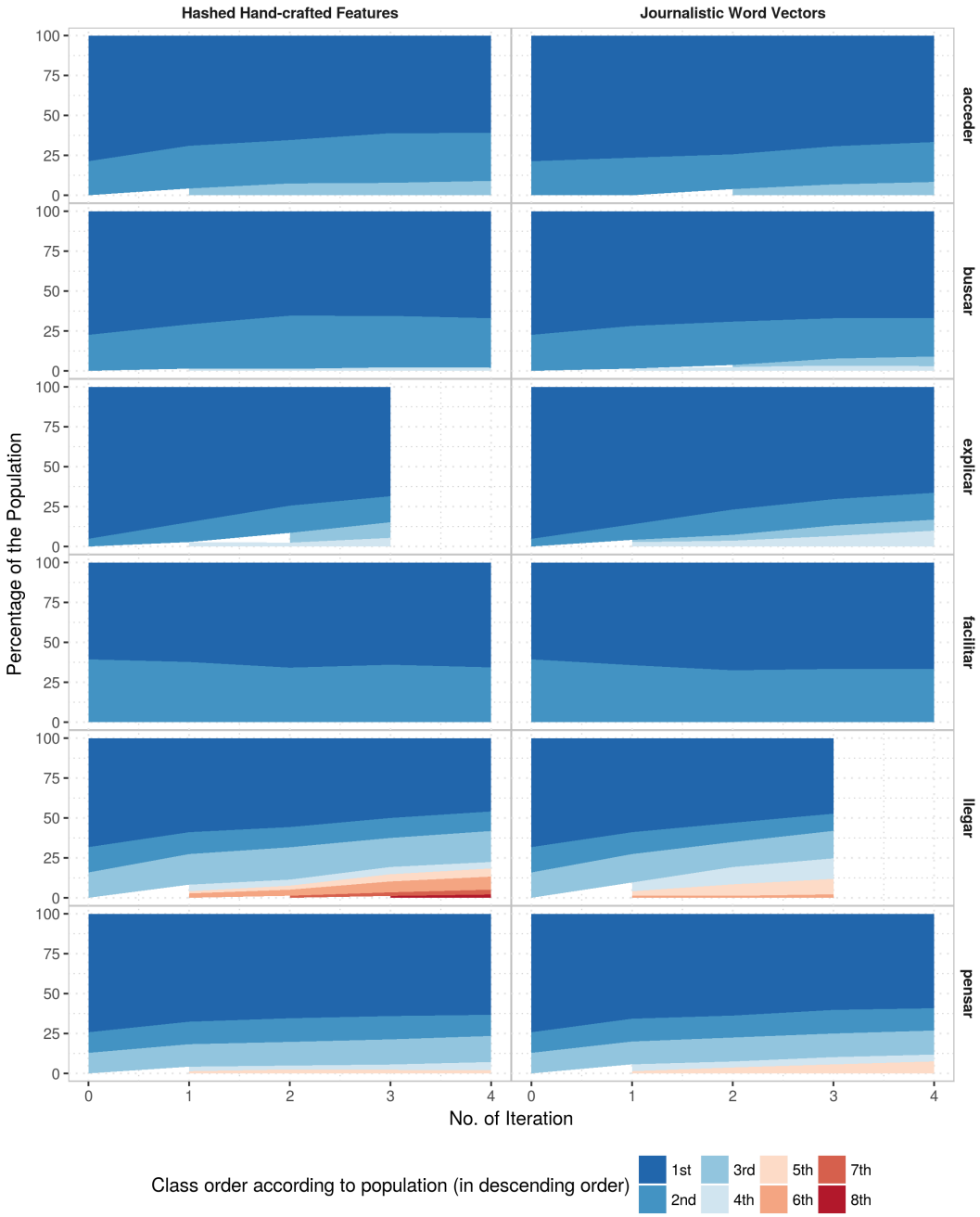


FIGURA 7.2: Distribución poblacional de las clases a través de las iteraciones del algoritmo de aprendizaje activo como proporción del total de conjunto de datos de entrenamiento

- Cada columna es un tipo de representación: atributos manuales y vectores de palabras periodísticos.
- La coordenada x representa las iteraciones del algoritmo de aprendizaje activo.
- La coordenada y representa el porcentaje de la población.
- Cada área de un color diferente representa la proporción de ejemplos para cada clase del conjunto de datos. Las clases nuevamente están ordenadas de acuerdo

al número de ejemplos en el conjunto de datos supervisado original.

Lo primero que hay que notar en la figura, especialmente comparada con la figura similar del capítulo anterior, es el número de iteraciones. Mientras que para las iteraciones de autoaprendizaje podría llegar a 100, para el aprendizaje activo sólo hice las anotaciones para 4 iteraciones en total (más la iteración inicial, que se representa en el número 0). Por eso digo que el trabajo realizado para el aprendizaje activo es en su mayor parte exploratorio.

Nótese que hay dos casos en los que las iteraciones son menores: "explicar" (para la representación de los rasgos artesanales) y "llegar" (para la palabra representación de embeddings). En estos casos la iteración terminó antes porque se cumplió el criterio de parada del error de validación.

La segunda cosa notable en la figura es el aumento en el número de clases que ocurren a medida que avanzan las iteraciones. Esto es el resultado de lo que expliqué antes: el algoritmo selecciona aquellas instancias sobre las que no tiene suficiente información, que son generalmente aquellas que no forman parte del corpus supervisado original. Esta es una marcada diferencia con el autoaprendizaje: en el aprendizaje activo las clases menos frecuentes también crecen en número de ejemplos a través de las iteraciones, no sólo la más frecuente.

7.4.2.2 Población agregada por sentido por iteración

La figura 7.3 muestra la proporción de ejemplos añadidos por clase en cada iteración. Es un gráfico de barras apiladas donde cada barra representa el total de ejemplos añadidos en la iteración divididos por la proporción de clases anotadas automáticamente como tales. La estructura de la parcela es la siguiente:

- Cada fila muestra los resultados para un lema testigo: "acceder", "buscar", "explicar", "facilitar", "llegar", y "pensar".
- Cada columna es un tipo de representación: atributos manuales y vectores de palabras periodísticos.
- La coordenada x representa las iteraciones del algoritmo de aprendizaje activo.
- La coordenada y representa el porcentaje de ejemplos anotados automáticamente y agregados al modelo.
- Cada barra representa la distribución de ejemplos agregados in esa iteración. Cada color de la barra representa la clase con la cual fueron anotados los ejemplos.

En la figura hay una mejor vista de lo que está sucediendo a lo largo de cada iteración. En general, las clases menos frecuentes añaden más ejemplos que las clases más frecuentes a través de iteraciones. Esto es consecuencia del uso del muestreo de incertidumbre, que selecciona las clases cercanas al límite de decisión del clasificador.

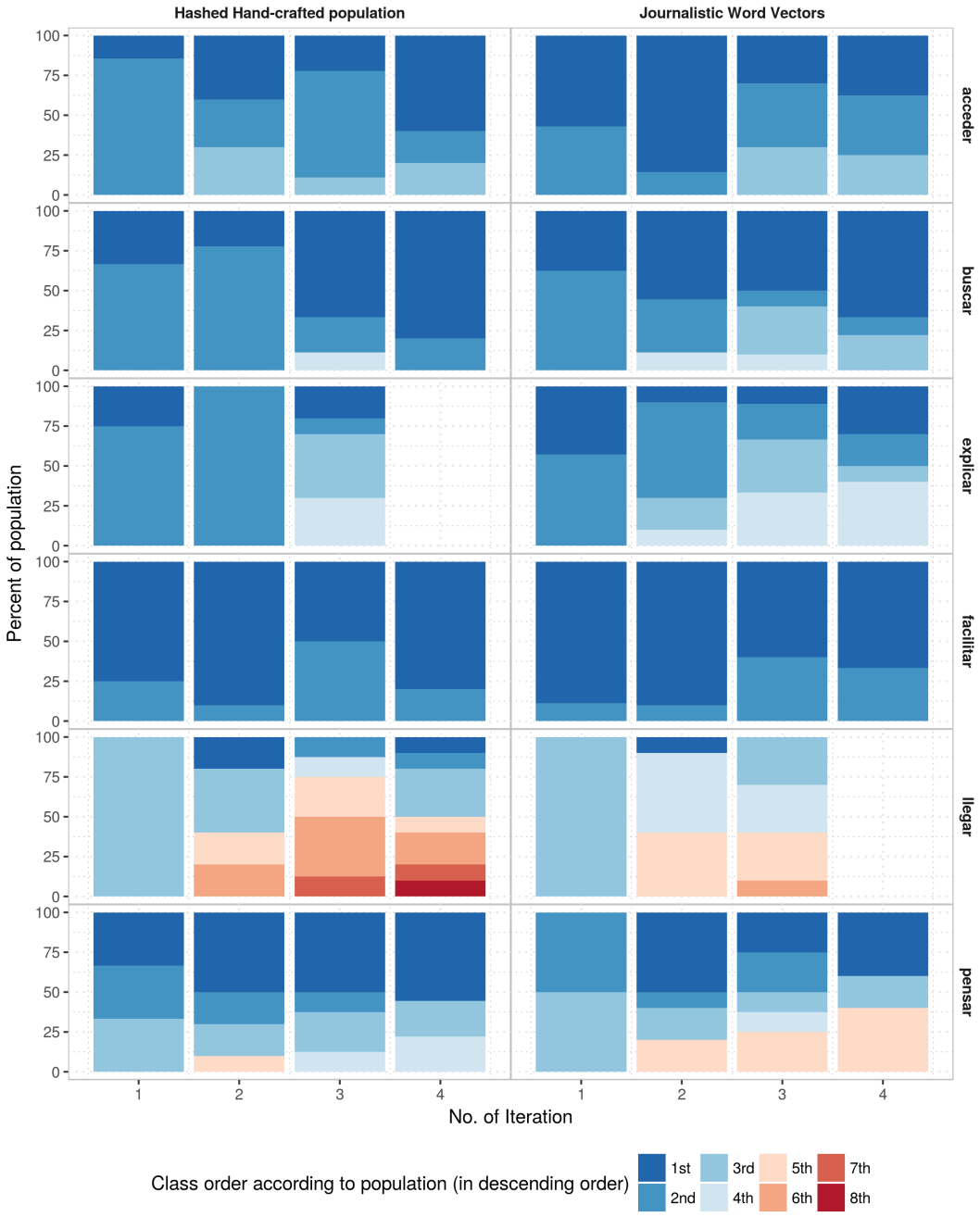


FIGURA 7.3: Población agregada por sentido en cada iteración de aprendizaje activo como proporción de todos los ejemplos agregados en la iteración

En general, los embeddings de palabras son más uniformes cuando se añaden ejemplos de muchas clases diferentes. La consecuencia de esto, puedo hipotetizar, es que para los rasgos artesanales, y como consecuencia de la baja generalización que tienen, el clasificador está más seguro de la clase más frecuente y por lo tanto a la hora de aplicar la técnica de muestreo de incertidumbre elige sobre todo ejemplos de las clases menos frecuentes. Los embeddings de palabras, por otro lado, tienen una mejor generalización desde el principio, por lo que cuando se muestrean elementos con

incertidumbre puede tener poca certeza también para algunos de los ejemplos de la clase más frecuente.

En cualquier caso, a partir de estos resultados tengo suficiente evidencia para apoyar la hipótesis 4.1 que afirma que se mantiene la distribución de las diferentes clases a través de las iteraciones del algoritmo de aprendizaje activo.

7.4.3 Hipótesis 4.2

Continúo con la prueba de la hipótesis 4.2. Recuerde que la hipótesis establece que el algoritmo de aprendizaje activo agrega más información con los ejemplos que agrega al modelo en comparación con el autoaprendizaje. Esto es consecuencia de que los ejemplos añadidos son de clases sobre las que el modelo ya tiene menos información debido a la técnica de muestreo de incertidumbre.

Para probar la hipótesis mido los resultados del experimento 7.3, que registra el número de veces que una característica aparece con una clase. Sin embargo, lo que más me interesa en esta ocasión es el número total de características añadidas por iteración tanto de algoritmos de autoaprendizaje como de aprendizaje activo. Estos resultados son medidos por Metric 6 que es el conteo normalizado de características añadidas en una iteración por los ejemplos añadidos en la misma iteración. La razón para utilizar esta medida en lugar del recuento bruto es que está claro que el número de características para el autoaprendizaje será mayor que para el aprendizaje activo, ya que el número de ejemplos que el autoaprendizaje anota en cada iteración es de una magnitud mayor que el aprendizaje activo. Sin embargo, lo que la Hipótesis establece, y lo que me interesa, es si estos menos ejemplos del algoritmo de aprendizaje activo realmente añaden más información que el algoritmo de autoaprendizaje.

La figura 7.4 muestra la cantidad de características añadidas en cada iteración tanto de autoaprendizaje como de aprendizaje activo normalizadas por la cantidad de ejemplos añadidos. La estructura del gráfico es la siguiente:

- Cada fila muestra los resultados para un lema testigo: “acceder”, “buscar”, “explicar”, “facilitar”, “llegar”, y “pensar”.
- Cada columna es un tipo de representación: atributos manuales y vectores de palabras periodísticos.
- La coordenada x representa las iteraciones del algoritmo de aprendizaje activo.
- The y-coordinate axis represents the normalized count of unique features by the number of examples added.
- La coordenada y representa el conteo de atributos únicos normalizado por el número de ejemplos agregados.
- Cada línea representa el número de atributos y los colores distintos representan el algoritmo: autoaprendizaje y aprendizaje activo.

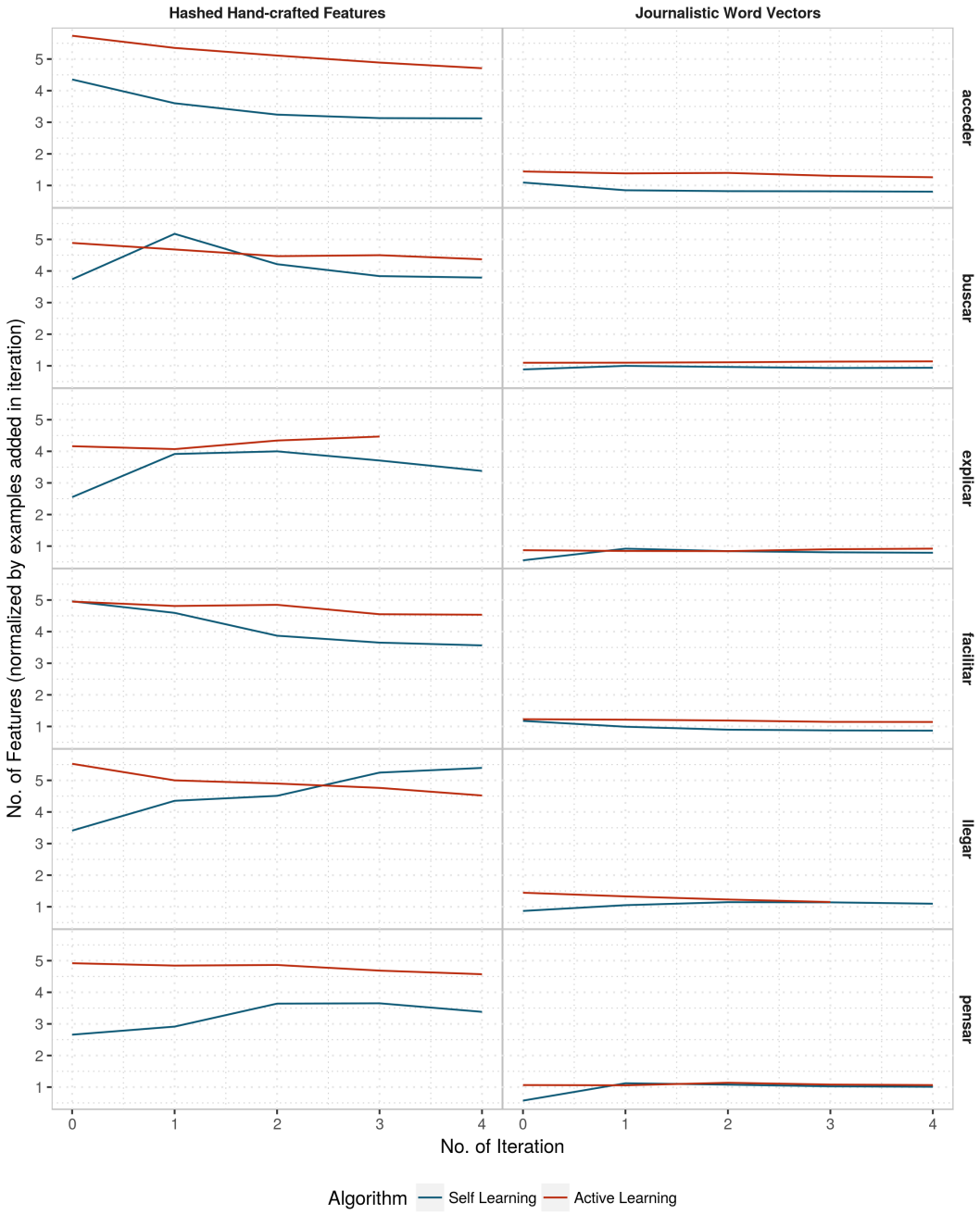


FIGURA 7.4: Número de atributos agregados en cada iteración de autoaprendizaje y aprendizaje activo. Los atributos están normalizados por el número de ejemplos agregados en la iteración.

La tendencia mostrada en la figura 7.4 es bastante clara. El aprendizaje activo es añadir más información al modelo añadiendo más funciones por ejemplo en comparación con el autoaprendizaje. Esto es claramente una consecuencia del aprendizaje activo añadiendo ejemplos de clases menos frecuentes en contraste con el autoaprendizaje como mostré en los resultados de la Hipótesis anterior. Además, el aprendizaje activo está añadiendo ejemplos de clases que no existen para el autoaprendizaje, ya que las clases no forman parte del modelo inicial.

Para el aprendizaje activo, la clase más frecuente, que contiene la mayoría de las características del modelo inicial, no es la única que crece en cada iteración (de hecho, a veces es la que tiene el menor número de ejemplos añadidos en una iteración). Por lo tanto, esa clase no está superando todas las características añadidas por iteración. Luego, cuando se agregan instancias de clases con menos ocurrencias, los datos enriquecen más el modelo inicial. Esto también se puede ver inspeccionando el PMI de las características y las clases, como en la siguiente sección.

7.4.3.1 PMI de atributos

Hay otra vista para los resultados del experimento 7.3, que es la medida por el PMI definido en la métrica 5, que asocia las características a las clases con un PMI superior y luego obtiene el PMI medio por clase. En esta sección muestro los resultados medidos por esa métrica y saco conclusiones con respecto a la Hipótesis.

La figura 7.5 muestra la media y el error estándar de la media de la información mutua puntual entre características y clases a través de las iteraciones de aprendizaje activas. Recordemos que la métrica asocia características a la clase con la que tiene un PMI mayor. La figura muestra un gráfico de líneas con la siguiente estructura:

- Cada fila muestra los resultados para un lema testigo: “acceder”, “buscar”, “explicar”, “facilitar”, “llegar”, y “pensar”.
- Cada columna es un tipo de representación: atributos manuales y vectores de palabras periodísticos.
- La coordenada x representa las iteraciones del algoritmo de aprendizaje activo.
- La coordenada y representa la PMI de los atributos con las clases en escala logarítmica.
- Cada color representa una clase.
- La línea más oscura representa la media de la PMI para esa clase y el área sombreada representa el error estándar de la media para esa PMI.

La figura muestra que la media del PMI de la clase más frecuente con respecto a las características a las que se asocia sigue siendo la más baja en comparación con el resto de las clases. Esto es análogo a lo que sucedió en el autoaprendizaje y ya lo mostré en la Sección 6.4.7.2.

Sin embargo, a diferencia de lo ocurrido en el autoaprendizaje, el PMI no está disminuyendo en particular para la clase más frecuente o para aquellas clases que se consolidan en el modelo. Disminuye para aquellas nuevas clases añadidas a través de las iteraciones que tienen pocos ejemplos. Esto sin embargo puede ser una consecuencia directa de Métrico 5 modelar clases inadecuadamente con pocos ejemplos. En cualquier caso, los ejemplos de clases desconocidas para el modelo supervisado de semillas tienen poca información para empezar, especialmente en las primeras iteraciones después de

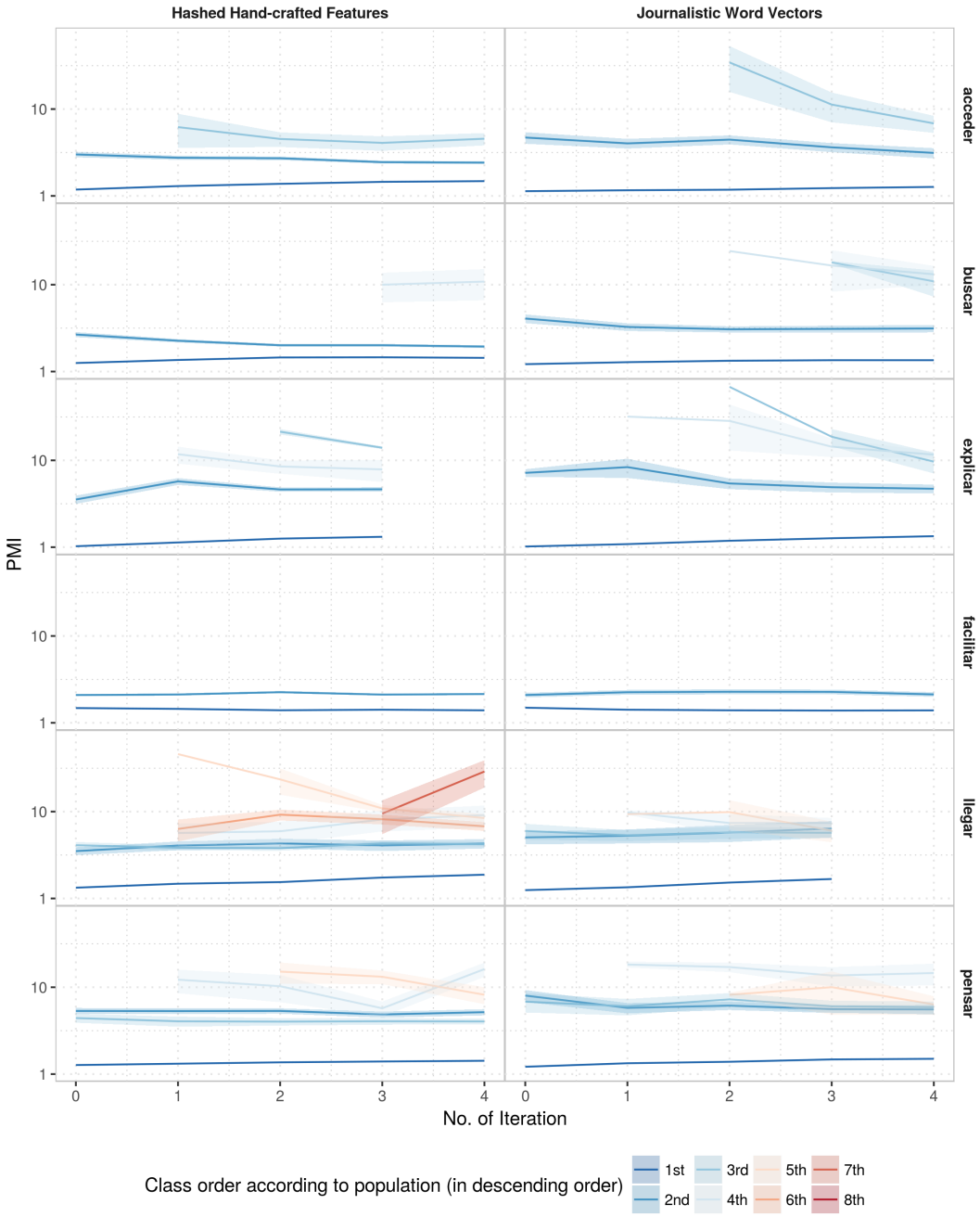


FIGURA 7.5: PMI de los atributos con cada clases asociada a lo largo de las iteraciones de aprendizaje activo

que ocurren. En algunos casos, cuando no hay área sombreada es porque los ejemplos agregados para dicha clase son menos de 3 en total, por lo tanto no tiene ninguna variación.

El hecho de que el PMI de las clases consolidadas (las que aparecen en el conjunto de datos supervisados originales) no disminuya contrasta con lo que mostré para el autoaprendizaje. Dado que el aprendizaje activo no deriva a añadir sólo ejemplos de la clase más frecuente, entonces tenemos que la clase no arrastra el ruido mediante la adición de características que no son necesariamente de la clase. Es decir, cualquier

información presente en la clase se mantiene a medida que aumenta el número de ejemplos.

Puedo concluir que la forma en que se incorporan nuevos ejemplos al modelo es radicalmente diferente para cada algoritmo. El autoaprendizaje añade como ejemplos de la clase más frecuente aquellas instancias débilmente definidas por el mero peso que la clase tiene en las decisiones del modelo. Estas instancias débilmente definidas desdibujan los límites de decisión que el modelo tiene sobre esa clase. Por el contrario, la estrategia basada en el muestreo de incertidumbre está orientada precisamente a aumentar la definición del límite de decisión.

De los resultados vistos en esta sección es seguro asumir que la hipótesis 4.2 tiene suficiente evidencia para ser aceptada.

7.5 Conclusiones

Los experimentos que hice en este capítulo tenían muy pocos datos debido al costo adicional de anotar ejemplos, necesarios para el aprendizaje activo. Decidí ir más allá con esta opción en lugar de hacer algún tipo de simulación basada en los ejemplos ya comentados ya que el conjunto de datos etiquetado era tan pequeño que seleccionar ejemplos en ese universo invalida una de las suposiciones del aprendizaje activo, que es obtener los ejemplos del universo que maximizarán el aprendizaje en el modelo. Esto se debe a que el número de ejemplos posibles que utilizan una simulación de este tipo no cubre lo suficiente el universo de ejemplos posibles.

En primer lugar, el aprendizaje activo muestra un mejor rendimiento general que el autoaprendizaje, al mostrar mejores resultados en el corpus de pruebas. Esto no es sólo para las clases más frecuentes, sino también para algunas de las clases menos frecuentes. Este mejor desempeño se explica por los resultados obtenidos al aceptar ambas hipótesis del capítulo.

Los resultados experimentales me llevan a aceptar la hipótesis 4.1. El modelo de aprendizaje activo mantiene la representatividad de los datos a través del algoritmo, a diferencia de lo que sucedió con el autoaprendizaje, donde los nuevos ejemplos se agregaron como parte de la clase más frecuente. En particular, las clases menos frecuentes son las que más se benefician con este modelo como consecuencia de la naturaleza de la propia técnica de muestreo de incertidumbre, que selecciona para la anotación las instancias sobre las que el modelo tiene menos información.

La hipótesis 4.2 también puede aceptarse según los resultados experimentales. La tendencia muestra que el aprendizaje activo añade en cada iteración más información que el autoaprendizaje, teniendo en cuenta el número de ejemplos que añade en cada iteración. De hecho, el modelo añade más información añadiendo más características nuevas por ejemplo, que también resultan ser más informativas de las clases, especialmente las menos frecuentes (esto es lo que se muestra cuando se muestra gráficamente PMI). De esta manera hay límites más claros para las clases menos frecuentes y, recíprocamente, también en la clase más frecuente. En contraste, yo había encontrado que

el autoaprendizaje asocia nuevas características mayormente a la clase más frecuente y así debilitando los límites de decisión que el modelo tenía sobre ella.

Una vez más, se trata sólo de resultados preliminares y las conclusiones que he podido extraer de ellos son sólo provisionales. Para hacer un estudio más exhaustivo debería anotar más datos con la ayuda de un experto en dominios y ver cómo estos resultados se amplían con más iteraciones. Una posible hipótesis a desarrollar en este sentido es que eventualmente las clases menos frecuentes encontrarán un límite superior para que el modelo tenga suficiente certeza sobre cada una de las clases menos frecuentes. En ese momento, el algoritmo comenzará a converger y seleccionará las instancias del conjunto no etiquetado de casi todas las clases de forma uniforme.

En cualquier caso, la principal conclusión de este capítulo es que el aprendizaje activo es una fuente interesante para ampliar la cobertura de un modelo. No sufre el problema fundamental del autoaprendizaje, es decir, la deriva hacia la clase más frecuente. Esto es una consecuencia directa de la forma en que las instancias a anotar son seleccionadas por el algoritmo. De hecho, esta selección de instancias más el valor añadido de un experto en el dominio que realiza la anotación en lugar del propio algoritmo basado en la certeza, da como resultado un modelo más robusto con más información sobre las clases sobre las que originalmente tenía poca o ninguna información.

Sin embargo, aún quedan retos por delante. En primer lugar, el proceso de anotación es costoso. En comparación con el autoaprendizaje, que hace la anotación automáticamente, el aprendizaje activo requiere que un humano haga trabajo manual. Aunque el algoritmo trata de seleccionar aquellas instancias que tienen el mayor impacto en el modelo, el trabajo manual sigue siendo costoso. Además, la anotación no es fácil, como expliqué en la Sección 7.3.4.3, porque el recurso no es perfecto ya que es arbitrario en algunos aspectos como la granularidad de los sentidos o lo que se consideran los sentidos.

En el próximo capítulo de esta tesis exploraré otra técnica de aprendizaje conjunto. En ese caso, el algoritmo no es una envoltura sobre algún clasificador supervisado que agrega datos no anotados al modelo. El algoritmo llamado *redes en escalera* es un algoritmo que minimiza dos objetivos diferentes, uno para los datos supervisados y otro para los datos no supervisados. El algoritmo hace todo el proceso automáticamente, en contraste con el aprendizaje activo, pero como no añade el ruido de los ejemplos como un conjunto de datos etiquetados, no deriva a la clase más frecuente como lo hace el autoaprendizaje.

El trabajo futuro de este capítulo se centrará en realizar experimentos más exhaustivos, utilizando más recursos de anotación. En particular, quiero centrarme en dos aspectos principales: en qué medida el número de anotaciones por iteración afecta al resultado final, y qué cantidad de anotaciones es necesaria para tener una mejor representación para cada clase. En el futuro se seguirá trabajando en otras técnicas para seleccionar los ejemplos de aprendizaje activo. En particular, una comparación entre la selección de las instancias con menos certeza y la selección de instancias aleatorias.

También hice algunos trabajos preliminares sobre la combinación del aprendizaje activo con el autoaprendizaje para atacar el problema de la deriva de clase más frecuente, añadiendo constantemente ejemplos manualmente a través de un muestreo de incertidumbre y un oráculo. Sin embargo, este trabajo no fue explorado más a fondo en esta tesis y también se deja fuera como trabajo futuro.

Capítulo 8

Redes neuronales en escalera

8.1 Estructura

En esta tesis hasta ahora he estado trabajando en diferentes técnicas para ayudar a expandir un modelo puramente supervisado para el desambiguación de sentidos verbales del español. Los dos últimos capítulos presentaron dos enfoques para el *aprendizaje semi-supervisado conjunto*, en el que los datos etiquetados y no etiquetados (anotados automática o manualmente) se utilizan juntos como parte de los datos de formación para un clasificador supervisado (por ejemplo, una red neuronal). En este capítulo trataré la última técnica semi-supervisada que estudié en esta tesis.

Las **redes neuronales en escalera** es un modelo de aprendizaje profundo que utiliza una arquitectura de red neuronal para aprender tanto de datos supervisados como no supervisados presentados por Rasmus et al. [**?**]. La técnica es otro ejemplo de una tarea de aprendizaje conjunto semi-supervisada. En contraste con el auto-aprendizaje y el aprendizaje activo, que son algoritmos de envoltura que utilizan un clasificador supervisado bajo el capó para expandir los datos de una fuente no etiquetada, las redes de escaleras combinan ambos conjuntos de datos en una *función objetivo* común que minimiza el uso de la propagación inversa y el descenso en pendiente. En el trabajo original, el modelo fue probado en una tarea de visión artificial, pero la arquitectura era lo suficientemente general como para poder aplicarlo en el área del desambiguación de sentidos verbales del español.

El concepto clave detrás de la construcción de una red de escaleras es tomar una red neuronal de retroalimentación (por ejemplo, un perceptrón multicapa) y tratarla como la parte del codificador de un autoencoder. A continuación, añada una parte del decodificador y utilice un error de reconstrucción calculado capa por capa. Los datos etiquetados se utilizan para minimizar el error dado por el codificador y una función de coste etiquetada (por ejemplo, cross-entropy). Los datos no etiquetados atraviesan todo el autoencoder y se minimiza el error de reconstrucción. La función de coste de la red de escaleras es una suma de funciones de coste etiquetadas y no etiquetadas.

Los *autoencoders apilados* [Vincent et al., 2010] fueron una idea clave para ayudar al entrenamiento de las redes neurales profundas. Usándolas para un *pre-entrenamiento no supervisado* (también conocido como *refinamiento*), ayudó a las arquitecturas de redes neurales profundas a converger más rápidamente hacia una solución y evitar el problema de *desaparición del gradiente* [Bengio et al., 1994]. Las redes escalonadas

se inspiran en esa idea, pero en lugar de hacer el ajuste fino de las capas de la red en un paso previo como en el pre-entrenamiento no supervisado, afinan durante el entrenamiento de la red añadiendo el valor de la función de costo no supervisado (del autoencoder) a la función de costo de la red neuronal de retroalimentación.

Este esquema contrasta con el de los algoritmos de envoltura que utiliza la función de coste puramente etiquetada del clasificador envuelto. Para los algoritmos de envoltura, los datos sin etiquetar añaden información convirtiendo una instancia sin etiquetar en una etiquetada. La nueva información en una red de escaleras, que proviene de los datos no etiquetados, se agrega al modelo de una manera diferente. En cada época, el algoritmo de entrenamiento se ajusta a los parámetros de la red utilizando todo el conjunto de datos etiquetado (aleatoriamente barajado), pero sólo una parte del conjunto de datos no etiquetado. Los datos no etiquetados añaden un coste adicional al tren de las redes que evita que se sobreponga al conjunto de datos etiquetado. De la misma manera, esa información ayuda a la red a encontrar una mejor representación codificada de los datos no etiquetados que sea útil para la tarea que la red de la escalera está tratando de aprender.

Quiero evaluar esta arquitectura de red neuronal en particular en la tarea de desambiguación de sentidos verbales del español y ver cómo afecta al rendimiento final. La idea es abordar el problema del autoaprendizaje y su desviación a la clase más frecuente. Para hacer eso, las redes de escaleras no comenzarán por un modelo supervisado y luego agregarán datos no etiquetados a él, sino que aprenderán paralelamente de ambos datos etiquetados como no etiquetados. Por otro lado, como el método no requiere intervención humana, supera el costo de anotación del aprendizaje activo.

Este capítulo prueba la siguiente hipótesis:

Hipótesis V *Las redes en escalera obtienen un mejor modelo de los datos.*

Espero que la causa de un mejor modelo sea la integración de datos no supervisados para elegir el modelo que sea consistente con los datos etiquetados y al mismo tiempo más adecuado para explicar la distribución de los datos no etiquetados.

Esto se prueba a través de las siguientes subhipótesis:

Subhipótesis 5.1 *El modelo de red de escaleras mejora en comparación con los métodos puramente supervisados y otros semisupervisados en un corpus de pruebas de resistencia.*

Subhipótesis 5.2 *En los nuevos ejemplos anotados con este clasificador, se mantiene la representatividad de las clases.*

Subhipótesis 5.3 *El ajuste excesivo del corpus etiquetado se evita mediante el uso de datos no etiquetados para minimizar una función de coste no supervisada.*

- El experimento 8.1 informa sobre el rendimiento del modelo de red de escaleras en el equipo de pruebas. El rendimiento se mide por la puntuación F1 por clase. Los resultados mostrados en la Sección 8.4.1 sirven para aceptar la Hipótesis

5.1, que el rendimiento sobre un modelo de datos de prueba retenidos para la red de escaleras mejora sobre los métodos anteriores.

- Experiment 8.2 muestra la distribución de las clases que tiene el modelo anotando automáticamente la instancia extraída de un corpus no etiquetado. Esto se mide por el recuento proporcional de las clases. Los resultados mostrados en la sección 8.4.2 sirven para aceptar parcialmente la hipótesis 5.2 ya que es válido para embeddings de palabras pero no para características hechas a mano.

- El experimento 8.3 sembrar la sobreadaptación de tendencias midiendo el error debido a la varianza de un modelo entrenado en un conjunto de datos sobre otros conjuntos de datos. Para medir esto utilizo la curva de aprendizaje (métrica 3) que ya expliqué en capítulos anteriores. Es importante notar que, esta vez, la curva de aprendizaje no se mide usando la información dada por los datos etiquetados, sino que se mide en base a la información que los datos no etiquetados dan al modelo. Los resultados de los experimentos mostrados en la sección 8.4.3 sirven para aceptar la hipótesis 5.3 que establece que el uso de datos no etiquetados para minimizar una función de coste no supervisada ayuda a disminuir la tendencia al sobreajuste del modelo.

En la sección 8.2 Presento un breve resumen del modelo de red de escaleras, citando el trabajo original y las obras en las que los autores se inspiran para venir con este enfoque novedoso.

En la sección 8.3 Repaso los puntos relevantes que conciernen a la experimentación en el capítulo. La mayor parte de la metodología es muy similar a la de los capítulos anteriores, en la mayoría de los casos proporciono indicaciones sobre las secciones en las que se describe por primera vez esta metodología. La sección 8.3.3 es la parte más importante, donde repaso los detalles detallados del modelo, incluyendo la arquitectura de la red de escaleras, el entrenamiento del algoritmo y la definición de las funciones de coste y las funciones auxiliares.

La sección 8.4 informa de los resultados de los experimentos y los analiza para aceptar las hipótesis expuestas en el capítulo.

Por último, en la sección 8.5 se extraen las conclusiones de este capítulo, recapitulando las Hipótesis y las implicaciones de aceptarlas o rechazarlas de acuerdo con las evidencias recogidas en los resultados. Termina esbozando el trabajo futuro.

8.2 Trabajo relevante

La idea de utilizar el aprendizaje sin supervisión para ayudar a entrenar una red neuronal fue propuesta por Suddarth y Kergosien [Suddarth and Kergosien, 1990]. La mayoría de los métodos que utilizan una tarea auxiliar para ayudar al aprendizaje supervisado sólo se aplican en la formación previa, seguida de un aprendizaje supervisado normal [Hinton and Salakhutdinov, 2006]. En contraste, con las redes de escaleras, las representaciones se aprenden conjuntamente [Rasmus et al., 2015a]. El

aprendizaje no supervisado se implementa a través de una tarea auxiliar, por ejemplo, la reconstrucción de la entrada. En el aprendizaje, las representaciones ocultas entre tareas supervisadas y no supervisadas son compartidas, y así la red se generaliza mejor. Proporciono detalles del método en lo que sigue.

El método de la red de escaleras presentado por Rasmus et al. sigue el trabajo de Valpola [Valpola, 2014], que propuso una red de escaleras en la que la tarea auxiliar es denotar representaciones en todos los niveles del modelo. La estructura del modelo es un autoencoder con conexiones de salto del codificador al decodificador y la tarea de aprendizaje es similar a la de los autocodificadores denotadores pero se aplica a todas las capas, no sólo a las entradas. Las conexiones de salto alivian la presión para representar detalles en las capas superiores del modelo porque, a través de las conexiones de salto, el decodificador puede recuperar cualquier detalle descartado por el codificador. Originalmente, las redes de escaleras sólo se aplicaban al aprendizaje no supervisado [Valpola, 2014, Rasmus et al., 2014], pero luego se combinaban con el aprendizaje supervisado.

Los aspectos clave de las redes de escaleras, como se describe en [Rasmus et al., 2015a], son los siguientes:

Compatibilidad con métodos supervisados La parte no supervisada complementa lo que se encuentra en el aprendizaje supervisado, añadiendo información y manteniendo la compatibilidad con el modelo puramente supervisado. Además, puede añadirse a las redes neuronales de retroalimentación existentes, por ejemplo, las percepciones multicapa o las redes neuronales convolucionales.

Escalabilidad como resultado del aprendizaje local Además de un objetivo de aprendizaje supervisado en la capa superior, el modelo tiene objetivos de aprendizaje locales no supervisados en cada capa, lo que lo hace adecuado para redes neuronales muy profundas.

Eficiencia computacional La parte del codificador del modelo corresponde al aprendizaje supervisado normal. Añadir un decodificador, como se propone en el documento, triplica aproximadamente el cómputo durante la capacitación, pero no necesariamente el tiempo de capacitación, ya que el mismo resultado puede lograrse más rápidamente mediante una mejor utilización de la información disponible. En general, el cálculo por escalas de actualización es similar a cualquier enfoque de aprendizaje supervisado que se utilice, con un pequeño factor multiplicador.

Los pasos involucrados en la implementación de la red Ladder son típicamente los siguientes: (i) tomar un modelo de retroalimentación que sirve para el aprendizaje supervisado como el codificador; (ii) agregar un decodificador que puede invertir los mapeos en cada capa del codificador y soporta el aprendizaje no supervisado; y (iii) capacitar a toda la red de la Escalera minimizando la suma de todos los términos de la función de costo.

8.3 Metodología

Este capítulo explora el uso de redes de escaleras para desambiguación de sentidos verbales del español. El modelo aprende minimizando una función de costo compuesta por objetivos supervisados y no supervisados. Tal como se define en la publicación original, las redes de escala no agregan ni tratan los datos no etiquetados como están etiquetados (a diferencia del autoaprendizaje y el aprendizaje activo). La mejor comparación posible con los otros métodos se realiza en el equipo de prueba. Sin embargo, para tener algunos puntos de referencia más, hice algunos cambios sobre el esquema original para hacerlo más comparable al enfoque de autoaprendizaje.

Una vez más, los resultados que muestro en este capítulo tratan de ser tan objetivos como sea posible, pero sólo hay un número de posibles métricas de evaluación y herramientas de visualización que puedo usar, y algunos de ellos pueden oscurecer otros resultados.

8.3.1 Recursos

Al igual que con el autoaprendizaje, hay dos recursos principales que se requieren para las redes de escaleras: un conjunto de datos con y sin etiqueta. Seguiré utilizando los mismos conjuntos de datos que he utilizado en todas las tesis: SenSem como datos supervisados y SBWCE como datos no supervisados.

Por favor refiérase a la sección 4.3.1.1 para ver los detalles del corpus SenSem y cómo fueron los conjuntos de datos de entrenamiento/prueba de introducción dividida siguiendo una aplicación de división estratificada, pero reteniendo al menos un ejemplo de cada clase por división. Al igual que lo que se hizo en el capítulo 6, en los experimentos de este capítulo el conjunto de datos SenSem fue sobremuestreado aleatoriamente en las clases menos frecuentes para tener una distribución más uniforme de todas las clases del conjunto de datos supervisados.

Las instancias no etiquetadas se toman del conjunto de datos de SBWCE que se describió en la sección 5.3.1.2 y se procesan como se describe en la sección 6.3.1.2. Consulte estas secciones para obtener más información sobre este conjunto de datos.

8.3.2 Atributos

Las características utilizadas en esta parte son las mismas que en los capítulos anteriores. Ambas características hechas a mano usando el truco de hash y embeddings de palabras siguen lo que se ha discutido en todos los capítulos anteriores.

Para obtener más información sobre los detalles de las características artesanales utilizadas en este capítulo, consulte la sección 4.3.2. Para información detallada sobre cómo lidiar con la expansión de las características dadas por los nuevos ejemplos añadidos al modelo, por favor refiérase a la sección 6.3.2.1.

Los embeddings de palabras son las formadas a partir del corpus periodístico descrito en detalle en la sección 5.3.1.3. El método para combinarlos en una instancia es la concatenación de vectores de palabras descrita en la sección 5.3.2.2.

8.3.3 Modelo de redes en escalera

En esta sección explicaré con más detalle el modelo de red Ladder. Esta es sólo una breve introducción tras el trabajo presentado por Rasmus et al. [Rasmus et al., 2015a]. Recomiendo al lector que se refiera a su trabajo que presenta una versión más detallada de lo que aquí describo.

8.3.3.1 Arquitectura

La arquitectura de una red de escaleras se basa en un autocodificador cuya parte codificadora también funciona como clasificador supervisado y cuya parte descodificadora funciona como aprendiz no supervisado reconstruyendo la entrada. La estructura de una red de escaleras suele seguir estos pasos:

1. Configurar un codificador que funcione como clasificador supervisado utilizando un modelo de feed-forward. La red tiene dos rutas de codificador: limpia y corrupta. La única diferencia es que el codificador corrupto añade ruido gaussiano en todas las capas. La adición de ruido sirve para evitar el sobreequipamiento del modelo resultante.

2. Establecer un decodificador que funcione como un estudiante no supervisado invirtiendo los mapeos en cada capa del codificador. El decodificador utiliza una función de eliminación de ruido para reconstruir las activaciones de cada capa dada la versión dañada. El objetivo en cada capa es la versión limpia de la activación y la diferencia entre la reconstrucción y la versión limpia sirve como coste de eliminación de la capa.

3. El coste supervisado, es decir, el error entre la etiqueta predicha y la etiqueta de verdad de tierra, se calcula a partir de la salida del codificador dañado y la etiqueta de destino. Por otro lado, el coste no supervisado es la suma del coste de eliminación de todas las capas escalado por un hiperparámetro que denota la importancia de cada capa. Por ejemplo, las primeras capas son más importantes que la última para reconstruir la entrada. El coste final es la suma del coste supervisado y el coste no supervisado.

Toda la red se entrena en una configuración totalmente etiquetada o semi-supervisada utilizando técnicas de optimización estándar (como el descenso por gradiente estocástico) para minimizar estos costes. Tenga en cuenta que la red de escaleras puede funcionar incluso sin datos auxiliares no etiquetados, pero la motivación original era hacer posible tomar clasificadores de retroalimentación de buen rendimiento y aumentarlos con un decodificador auxiliar.

La figura 8.1 muestra la estructura de una red de escaleras. La ruta corrupta (a la izquierda en la figura) añade ruido gaussiano $\mathcal{N}(0, \sigma^2)$ a cada capa del codificador. Cada capa contribuye a la función de coste, un término $C^{(l)} = ||\mathbf{z}^{(l)} - \hat{\mathbf{z}}^{(l)}||^2$ que entrena las capas superiores (tanto el codificador como el decodificador) para aprender

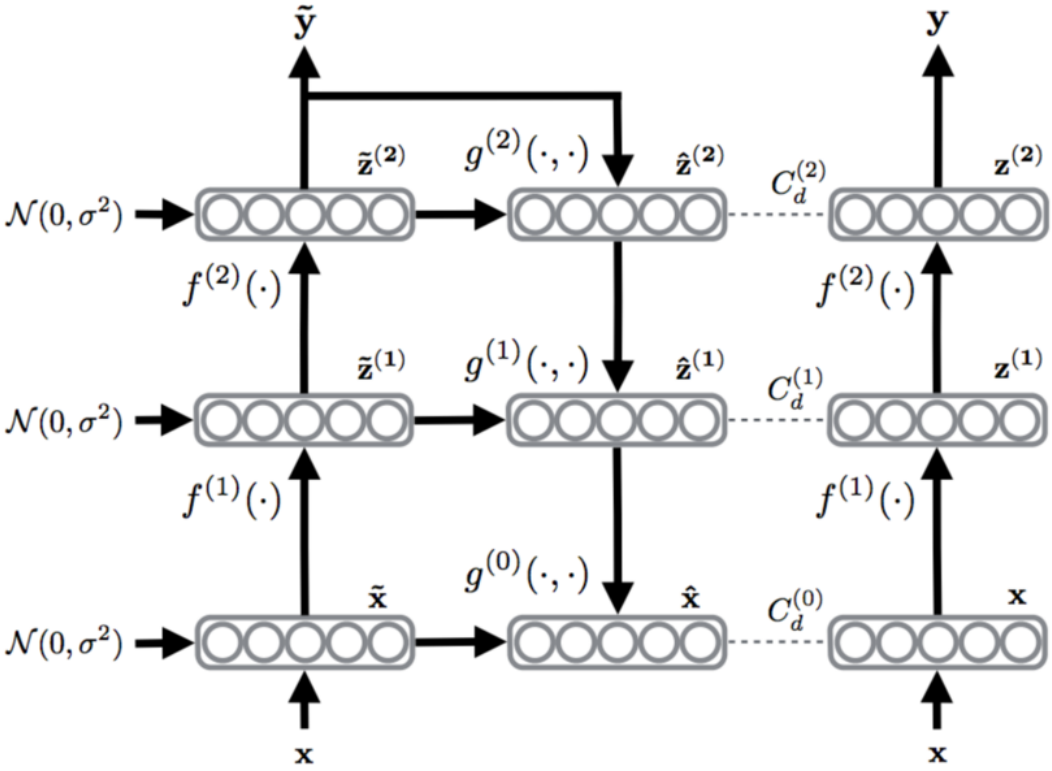


FIGURA 8.1: Una ilustración concetual de la red en escalera con dos capas ocultas ($L = 2$). El camino hacia adelante ($\mathbf{x} \rightarrow \mathbf{z}^{(1)} \rightarrow \mathbf{z}^{(2)} \rightarrow \mathbf{y}$) comparte la función $f^{(l)}$ con el camino corrupto, el codificador ($\mathbf{x} \rightarrow \tilde{\mathbf{z}}^{(1)} \rightarrow \tilde{\mathbf{z}}^{(2)} \rightarrow \tilde{\mathbf{y}}$). El decodificador ($\hat{\mathbf{z}}^{(2)} \rightarrow \hat{\mathbf{z}}^{(1)} \rightarrow \hat{\mathbf{x}}$) consiste en las funciones que remueven ruido $g^{(l)}$ y tiene las funciones de coste $C_d^{(l)}$ en cada capa cuyo objetivo es minimizar la diferencia entre $\hat{\mathbf{z}}^{(l)}$ y $\mathbf{z}^{(l)}$. La salida $\tilde{\mathbf{y}}$ del codificador también puede ser entrenada para coincidir con etiquetas disponibles $t(n)$. La figure original se encuentra disponible en el trabajo de Rasmus et al. [Rasmus et al., 2015a].

la función de ruido $\hat{\mathbf{z}}^{(l)} = g^{(l)}(\tilde{\mathbf{z}}^{(l)}, \hat{\mathbf{z}}^{(\mathbf{l+1})})$, el cual mapea el corrupto $\tilde{\mathbf{z}}^{(l)}$ en el ruido estimado $\hat{\mathbf{z}}^{(l)}$. Como la estimación $\hat{\mathbf{z}}^{(l)}$ incorpora todo el conocimiento previo acerca de $\mathbf{z}$, el mismo término de función de coste también entrena las capas de codificador que se muestran a continuación para encontrar características más limpias que coincidan mejor con las expectativas previas.

Dado que la función de coste necesita tanto el $\mathbf{z}^{(l)}$ limpio como el $\mathbf{tildez}^{(l)}$ corrompido, durante el entrenamiento el codificador se ejecuta dos veces: un pase limpio para $\mathbf{z}^{(l)}$ limpio y un pase corrompido para $\tilde{\mathbf{z}}^{(l)}$.

En los autocodificadores denoising [Vincent et al., 2010], un autocodificador está entrenado para reconstruir la observación original $\mathbf{x}$ a partir de una versión corrupta $\tilde{\mathbf{x}}$. El aprendizaje se basa simplemente en minimizar la norma de la diferencia del original $\mathbf{x}$ y su reconstrucción $\hat{\mathbf{x}}$ del dañado $\tilde{\mathbf{x}}$; ese es el costo$||\hat{\mathbf{x}} - \mathbf{x}||^2$. La diferencia principal es que en las redes de escaleras el coste de la reconstrucción se calcula capa por capa añadiendo las funciones de eliminación de ruido $\hat{\mathbf{z}} = g(\mathbf{z})$.

Una manera de imaginar la red Ladder es considerarla como una colección de autocodificadores densificadores anidados que comparten partes de la maquinaria densificadora entre sí a través de la función de coste C_d. Si no se utilizara esta función, desde el punto de vista del autoencoder en la capa l, las representaciones en las capas superiores serían opacas, tratadas como neuronas ocultas. En otras palabras, no habría ninguna razón en particular por la que la representación intermedia $\hat{\mathbf{z}}^{(l+i)}$ producida por el decodificador debería parecerse a las representaciones correspondientes $\mathbf{z}^{(l+i)}$ producidas por el codificador. Sólo la función de coste $C_d^{(l+i)}$ los une y obliga a la inferencia a proceder en orden inverso en el decodificador. Esta compartición ayuda a un autoencoder de denotación profunda a aprender el proceso de denotación, ya que divide la tarea en subtareas significativas de representaciones intermedias de denotación.

La normalización por lotes [Ioffe and Szegedy, 2015] se aplica a cada preactivación incluyendo la capa superior para mejorar la convergencia (debido a la reducción del desplazamiento de covariables) y para evitar que el coste de eliminación fomente la solución trivial (el codificador emite valores constantes ya que son los más fáciles de eliminar). Se utiliza la conexión directa entre una capa y su reconstrucción decodificada. La red se denomina red de escalera porque la arquitectura resultante del codificador/decodificador se asemeja a una escalera, ya que la función de coste entre las capas de espejo podría verse como las cadenas en los peldaños de la escalera.

8.3.3.2 Entrenamiento de la red

Algoritmo 1 enumera el feed-forward pass de la red completa de Ladder con una instancia de formación (con o sin etiqueta). Los resultados del algoritmo son los resultados de predicción $\mathbf{y}$ de la red así como la función de coste C. Como se puede ver en la figura, el algoritmo pasa por las dos rutas del codificador –limpia y corrupta– así como la ruta del decodificador. Como se ve en el algoritmo 1 la función de coste resulta de la suma del coste supervisado dado por la probabilidad de registro negativa (si es aplicable, es decir, la instancia tiene una etiqueta), y el coste no supervisado de la reconstrucción capa por capa.

El algoritmo 1, sin embargo, muestra cómo se obtiene la función de coste para una sola instancia. Para un lote de instancias, la función de coste supervisado es la probabilidad media de registro negativo de que la salida ruidosa $\tilde{\mathbf{y}}$ coincida con el objetivo $t(n)$ dadas las entradas $\mathbf{x}(n)$.

$$C_c = -\frac{1}{N}\sum_{n=1}^{N} \log P(\tilde{\mathbf{y}} = t(n)|\mathbf{x}(n)).$$

La función de coste de eliminación de ruido no supervisada, que se calcula capa por capa, para más de un caso es

$$C_d = \sum_{l=0}^{L} \lambda_l C_d^{(l)} = \sum_{l=0}^{L} \frac{\lambda_l}{N m_l} \sum_{n=1}^{N} \left|\left|\mathbf{z}^{(l)}(n) - \hat{\mathbf{z}}_{\mathrm{BN}}^{(l)}(n)\right|\right|^2,$$

Algorithm 1 Cálculo de las funciones de coste y salida de la red en escaleras. Tomado de Rasmus et al. [Rasmus et al., 2015a]

Require: $\mathbf{x}(n)$
  # Corrupted encoder and training output
  $\tilde{\mathbf{h}}^{(0)} \leftarrow \tilde{\mathbf{z}}^{(0)} \leftarrow \mathbf{x}(n) + \texttt{noise}$
  for $l = 1$ **to** L **do**
    $\tilde{\mathbf{z}}^{(l)}_{\text{pre}} \leftarrow \mathbf{W}^{(l)}\tilde{\mathbf{h}}^{(l-1)}$
    $\tilde{\boldsymbol{\mu}}^{(l)} \leftarrow \texttt{batchmean}(\tilde{\mathbf{z}}^{(l)}_{\text{pre}})$
    $\tilde{\boldsymbol{\sigma}}^{(l)} \leftarrow \texttt{batchstd}(\tilde{\mathbf{z}}^{(l)}_{\text{pre}})$
    $\tilde{\mathbf{z}}^{(l)} \leftarrow \texttt{batchnorm}(\tilde{\mathbf{z}}^{(l)}_{\text{pre}}, \tilde{\boldsymbol{\mu}}^{(l)}, \tilde{\boldsymbol{\sigma}}^{(l)}) + \texttt{noise}$
    $\tilde{\mathbf{h}}^{(l)} \leftarrow \texttt{activation}(\boldsymbol{\gamma}^{(l)} \odot (\tilde{\mathbf{z}}^{(l)} + \boldsymbol{\beta}^{(l)}))$
  end for
  $P(\tilde{\mathbf{y}}|\mathbf{x}) \leftarrow \tilde{\mathbf{h}}^{(L)}$
  # Clean encoder
  $\mathbf{h}^{(0)} \leftarrow \mathbf{z}^{(0)} \leftarrow \mathbf{x}(n)$
  for $l = 1$ **to** L **do**
    $\mathbf{z}^{(l)}_{\text{pre}} \leftarrow \mathbf{W}^{(l)}\mathbf{h}^{(l-1)}$
    $\boldsymbol{\mu}^{(l)} \leftarrow \texttt{batchmean}(\mathbf{z}^{(l)}_{\text{pre}})$
    $\boldsymbol{\sigma}^{(l)} \leftarrow \texttt{batchstd}(\mathbf{z}^{(l)}_{\text{pre}})$
    $\mathbf{z}^{(l)} \leftarrow \texttt{batchnorm}(\mathbf{z}^{(l)}_{\text{pre}}, \boldsymbol{\mu}^{(l)}, \boldsymbol{\sigma}^{(l)})$
    $\mathbf{h}^{(l)} \leftarrow \texttt{activation}(\boldsymbol{\gamma}^{(l)} \odot (\mathbf{z}^{(l)} + \boldsymbol{\beta}^{(l)}))$
  end for
  # Prediction output
  $P(\mathbf{y}|\mathbf{x}) \leftarrow \mathbf{h}^{(L)}$
  # Decoder and denoising
  for $l = L$ **to** 0 **do**
    if $l = L$ **then**
      $\mathbf{u}^{(L)}_{\text{pre}} \leftarrow \tilde{\mathbf{h}}^{(L)}$
    else
      $\mathbf{u}^{(l)}_{\text{pre}} \leftarrow \mathbf{V}^{(l+1)}\hat{\mathbf{z}}^{(l+1)}$
    end if
    $\boldsymbol{\mu}^{(l)} \leftarrow \texttt{batchmean}(\mathbf{u}^{(l)}_{\text{pre}})$
    $\boldsymbol{\sigma}^{(l)} \leftarrow \texttt{batchstd}(\mathbf{u}^{(l)}_{\text{pre}})$
    $\mathbf{u}^{(l)} \leftarrow \texttt{batchnorm}(\mathbf{u}^{(l)}_{\text{pre}}, \boldsymbol{\mu}^{(l)}, \boldsymbol{\sigma}^{(l)})$
    $\forall i : \hat{z}^{(l)}_i \leftarrow g(\tilde{z}^{(l)}_i, u^{(l)}_i)$
    $\forall i : \hat{z}^{(l)}_{i,\text{BN}} \leftarrow \frac{\hat{z}^{(l)}_i - \mu^{(l)}_i}{\sigma^{(l)}_i}$
  end for
  # Cost function C for training
  $C \leftarrow 0$
  if t(n) **then**
    # NLL cost for labeled data
    $C \leftarrow -\log P(\tilde{\mathbf{y}} = t(n)|\mathbf{x}(n))$
  end if
  $C \leftarrow C + \sum_{l=0}^{L} \lambda_l \left|\left|\mathbf{z}^{(l)} - \hat{\mathbf{z}}^{(l)}_{\text{BN}}\right|\right|^2$

donde m_l es el tamaño de la capa, N es el número de instancias de entrenamiento, y el hiperparámetro λ_l es un multiplicador por capas que determina la importancia del coste de eliminación. Finalmente, la función de costo se obtiene como la suma de las funciones de costo supervisadas y no supervisadas $C = C_c + C_d$.

Tanto en los codificadores –limpios o corruptos– como en el decodificador, cada capa utiliza la normalización por lotes, como se ha explicado anteriormente, para mejorar la convergencia y evitar que el coste de eliminación de ruido fomente la solución trivial. La ruta limpia difiere de la ruta corrupta en que esta última añade ruido gaussiano antes de la activación. Además, la salida corrupta se utiliza para la formación, pero la salida limpia se utiliza para la predicción.

La función g es la *función combinadora* de tal manera que $\mathbf{z}^{(l)} = g(\tilde{\mathbf{z}}^{(l)}, \hat{\mathbf{z}}^{(l+1)})$ que puede aproximar la función denotativa óptima para la familia de distribuciones observadas. Por lo tanto, se espera que la función g forme una reconstrucción $\hat{\mathbf{z}}^{(l)}$ que se asemeje al clean $\mathbf{z}^{(l)}$ dado el corrupto $\tilde{\mathbf{z}}^{(l)}$ y la reconstrucción de nivel superior $\hat{\mathbf{z}}^{(l+1)}$.

El trabajo de Rasmus et al. [Rasmus et al., 2015a] presenta la instauración de g que utilicé en la implementación para los experimentos de este capítulo. Sin embargo,

puede haber otras versiones de la función de combinador. Pezeshki et al. [**?**] presentaron otra versión de la función combinadora que, según ellos, supera al original. La experimentación con otras funciones de combinación, o incluso el diseño de una función de combinador para la tarea de desambiguación de sentidos verbales del español en particular, se deja para el trabajo futuro. La función del combinador utilizada para estos experimentos, definida por Rasmus et al. [**?**], es la siguiente:

$$\hat{z}_i^{(l)} = g_i(\tilde{z}_i^{(l)}, u_i^{(l)}) = \left(\tilde{z}_i^{(l)} - \phi_i(u_i^{(l)})\right) \psi_i(u_i^{(l)}) + \phi_i(u_i^{(l)}),$$

Donde las funciones $\phi_i(u_i^{(l)})$ y $\psi_i(u_i^{(l)})$ se modelan como no linealidades expresivas:

$$\begin{aligned}\phi_i(u_i^{(l)}) &= a_{1,i}^{(l)} \sigma(a_{2,i}^{(l)} u_i^{(l)} + a_{3,i}^{(l)}) + a_{4,i}^{(l)} u_i^{(l)} + a_{5,i}^{(l)} \\ \psi_i(u_i^{(l)}) &= a_{6,i}^{(l)} \sigma(a_{7,i}^{(l)} u_i^{(l)} + a_{8,i}^{(l)}) + a_{9,i}^{(l)} u_i^{(l)} + a_{10,i}^{(l)},\end{aligned}$$

donde σ es la *función sigmoide*.

El modelo tiene los siguientes parámetros entrenables, los cuales pueden ser entrenados simplemente usando el algoritmo de retropropagación para optimizar la función de costo C:

$\mathbf{W}^{(l)}$, $\mathbf{V}^{(l)}$ Las matrices de peso del codificador y del decodificador respectivamente por cada capa l de la red. Notar que $\mathbf{V}^{(l)}$ tiene la misma dimensión que la transpuesta de $\mathbf{W}^{(l)}$.

$\boldsymbol{\gamma}^{(l)}$, $\boldsymbol{\beta}^{(l)}$ Los parámetros de sesgo y escala respectivamente para cada capa l de la red.

$a_{1,i}^{(l)}, \ldots, a_{10,i}^{(l)}$ Los parámetros de las funciones $\phi_i(u_i^{(l)})$ y $\psi_i(u_i^{(l)})$, propuestas por los autores.

La función de coste C se minimiza utilizando el descenso de gradiente estocástico. En cada iteración (también llamada época), el algoritmo toma un lote de elementos del mismo tamaño tanto del conjunto de datos etiquetado como del no etiquetado. Utiliza esos elementos para hacer una propagación hacia adelante. Esto va lote por lote hasta que todas las instancias en uno de los dos conjuntos de datos, ya sea el conjunto de datos etiquetado o no etiquetado, se han utilizado para la formación. Normalmente, el conjunto de datos etiquetado es más pequeño que el conjunto de datos no etiquetado, por lo que suele ocurrir que el conjunto de datos etiquetado se consume mucho antes que el conjunto de datos no etiquetado. A continuación, se baraja el conjunto de datos que se ha utilizado completamente para la formación (normalmente el conjunto de datos supervisado) y el algoritmo comienza de nuevo a tomar lotes de ese conjunto de datos como si fuera nuevo. El set de datos que no se ha consumido completamente se sigue consumiendo de la misma manera. Utilizando la implementación original del algoritmo, el tamaño del lote era igual al tamaño del conjunto de datos etiquetado.

Esto significa que en en cada iteración la red de escalera minimizó la función de costo sobre la totalidad de los datos etiquetados (con un orden aleatorio) y un segmento del conjunto de datos sin etiquetar (que es más grande).

8.3.3.3 Hiperparámetros

El algoritmo de la red de escalera tiene un número importante de hiperparámetros para sintonizar y experimentar. Algunos de estos hiperparámetros vienen dados por la red neuronal (por ejemplo, el número y el tamaño de las capas), otros por el modelo de red de escalera (por ejemplo, el factor de escala para el ruido) y otros por el procedimiento de entrenamiento (por ejemplo, el criterio de parada). A medida que el número de experimentos crece exponencialmente mientras más hiperparámetros sintonizo, decidí mantenerlo lo más simple posible para evitar perder el enfoque de la experimentación del capítulo.

En primer lugar está la estructura de la red. Para mantener un punto de comparación con los experimentos de capítulos anteriores, el codificador tiene tres capas ocultas ($L = 3$) con 500, 250 y 100 neuronas cada una. Como tal, el decodificador, que es simétrico al codificador, tiene tres capas con 100, 250 y 500 neuronas cada una.

Los hiperparámetros definidos por el algoritmo de red de escalera son el factor de escala para el ruido en el codificador dañado, y el factor de capa que denota la importancia de cada coste de denotar (es decir, λ_l). Para evitar un crecimiento exponencial en el número de experimentos decidí usar los hiperparámetros descritos en los experimentos del trabajo de Rasmus et al. El factor de escala para el ruido fue de 0,3 y la importancia de las capas de coste de denotar se estableció en $\lambda^{(0)} = 1000$, $\lambda^{(1)} = 10$, y $\lambda^{(l \geq 2)} = 0{,}01$.

8.3.3.4 Criterio de parada

Las redes de escalera, a diferencia de los algoritmos de envoltura que he discutido antes, no parten de un algoritmo supervisado que haya sido previamente entrenado en los datos supervisados. En cambio, aprende minimizando la función de objetivo con datos supervisados y no supervisados desde cero. Así, la forma de detener el algoritmo es precisamente buscando la convergencia o estableciendo un número máximo de iteraciones.

Añadí otro criterio de parada para evitar que el algoritmo de redes de escaleras se sobrepase al conjunto de datos supervisado. En la sección 6.3.4 se explica que para el autoaprendizaje el 20 % de los datos de la formación está reservado para la validación, asegurando que cada clase está representada al menos por un ejemplo en ambos corpus. Utilizo esta misma técnica para seleccionar un conjunto de datos de validación a partir de los datos de formación. El conjunto de datos de validación sirve para comprobar en cada iteración que el algoritmo no está reduciendo el error de coste a expensas de la generalización.

El criterio de parada se convirtió entonces en uno de tres:

1. El algoritmo alcanzó el máximo número de iteraciones dadas.

2. El error en la función de costo convergió: el error de una iteración nueva es más grande o igual qu el error de la iteración previa más alguna tolerancia ϵ.

3. El error de los datos de validación es más grande o igual que el error de validación previo registrado por el algoritmo más una tolerancia η.

Se realizaron una serie de experimentos para verificar qué valores de η y ϵ mejoran la convergencia del modelo sin perder demasiada información. Sin embargo, en los experimentos, el valor que realmente marcó la diferencia fue el número de iteraciones, como comentaré más adelante en el capítulo.

8.3.4 Experimentos

Los experimentos presentados en esta sección ayudan a la aceptación o rechazo de las hipótesis presentadas al principio de este capítulo.

En primer lugar, la hipótesis 5.1 requiere un experimento para comparar entre redes de escaleras y los algoritmos vistos hasta ahora: aprendizaje supervisado, autoaprendizaje y aprendizaje activo. El experimento 8.1 prueba el modelo de red de escaleras en el set de pruebas y lo usa para compararlo con los algoritmos anteriores.

Experimento 8.1.

8.1a Entrenar el modelo con redes en escalera hasta que se llegue a un criterio de parada.

8.1b Evaluamos el modelo en el conjunto de datos de evaluación.

El experimento 8.2 informa de la distribución de las clases teniendo en cuenta tanto los datos etiquetados como algunos datos anotados automáticamente tomados de un conjunto de ejemplos no etiquetados diferentes del dado a la red de escaleras para minimizar la función de costes no supervisados.

Experimento 8.2.

1. Ejecute el algoritmo de red de la escalera sobre el conjunto de datos etiquetado L_1 y el conjunto de datos no etiquetado U_1.

2. En cada iteración utilice el modelo obtenido hasta ahora para predecir las etiquetas de un conjunto de datos no etiquetados U_2, diferente de los datos del conjunto de datos no etiquetados U_1.

3. A partir de ese conjunto, anote aquellos con una clase pronosticada que el modelo tenga una certeza por encima de un umbral. Este umbral se calcula siguiendo los pasos descritos en la sección 6.3.4: comienza al 100 % y se reduce lentamente hasta que tiene como máximo una certeza que es un 10 % mayor que la probabilidad aleatoria.

4. Quita esas instancias anotadas automáticamente del grupo de instancias no etiquetadas. Si no hay ninguna instancia en la que el modelo tenga la certeza requerida, no anote ninguna instancia.

5. Registrar la distribución de las clases tanto del conjunto de datos etiquetado como de las instancias etiquetadas automáticamente obtenidas hasta el momento.

El experimento 8.3 mide la tendencia al sobreequipamiento del clasificador de red de escaleras. Para ello, supervisa tanto el conjunto de datos de formación como el conjunto de datos de validación. Esto es muy similar a lo que hace el autoaprendizaje. Sin embargo, en este caso el número de ejemplos en el modelo no aumenta, ya que el propio modelo no añade ejemplos no comentados. Sin embargo, recuerde que cada iteración de capacitación toma todo el conjunto de datos supervisados y sólo una parte del conjunto de datos no supervisados. Así, en cada iteración, el algoritmo añade más información de los datos no supervisados con el objetivo de minimizar la parte no supervisada de la función de coste del algoritmo. Esto es hasta que el algoritmo haya cubierto todas las instancias no etiquetadas. Después de eso, el algoritmo no añade nuevos datos al modelo, sino que lleva a cabo iteraciones sucesivas dirigidas a ajustar mejor los datos disponibles. Sin embargo, para comparar con los algoritmos anteriores, para este experimento el algoritmo se detiene después de que atraviesa todo el conjunto de datos sin etiquetar una vez.

Experimento 8.3.

8.3a Tome sólo una parte de todo el conjunto de datos no supervisados para utilizarlos como datos no supervisados para el algoritmo.

8.3b Mezcle los conjuntos de datos de capacitación y validación y divídalos al azar con el muestreo estratificado.

8.3c Inicie el algoritmo de redes de escaleras.

8.3d En cada paso, ejecute una iteración de la capacitación y registre las predicciones sobre los datos de capacitación y validación.

8.3e Detenga el algoritmo una vez que se haya atravesado una vez todo el conjunto de datos sin etiquetar.

8.3f Repita todo el procedimiento n veces con una parte diferente del conjunto de datos no supervisados, hasta que se atraviese todo el conjunto de datos no supervisados.

8.3.5 Métricas

Los resultados del experimento 8.1 se informan con la puntuación F1 de cada clase en el conjunto de datos de prueba de los lemas simbólicos. Esto se hace para cada uno de

los cuatro algoritmos vistos hasta ahora: supervisado, autoaprendizaje, aprendizaje activo y redes de contactos.

Los resultados del experimento 8.2 reportan el conteo proporcional de elementos de cada clase como se hizo para los experimentos en los capítulos anteriores de esta tesis.

Finalmente, para el experimento 8.3 Utilizo el error debido a la varianza definido por la métrica 3 para reportar los resultados del experimento. Esto mide la tendencia del modelo a sobredimensionarse a medida que se añaden nuevos datos no etiquetados al modelo.

8.4 Análisis de resultados

En esta sección se reportan los resultados obtenidos por los experimentos y las medidas por las métricas que expliqué anteriormente.

En un primer conjunto de experimentos, ejecuto el algoritmo para un total de 100 iteraciones. Sin embargo, el algoritmo no se detuvo porque convergió, sino porque alcanzó el número máximo de iteraciones. Sin embargo, para la iteración 100 la mejora en el error era lo suficientemente baja como para considerar que el algoritmo había convergido.

Sin embargo, como mostraré a continuación, una vez visualizados los resultados en cuanto a la distribución de las clases (experimento 8.2), descubrí que cuando el algoritmo tenía unas 25 iteraciones la distribución de las clases empezaba a derivar hacia la más frecuente (independientemente del lema). Así que decidí realizar los mismos experimentos sólo con 25 iteraciones y ver cuáles eran los resultados.

Por supuesto, estos resultados son sólo una muestra de los datos disponibles que intentaré interpretar de la forma más objetiva posible. Sin embargo, puede haber algunos resultados que son oscurecidos por las visualizaciones elegidas.

8.4.1 Hipótesis 5.1

La hipótesis 5.1 establece que el modelo de red de escaleras mejora en comparación con los métodos puramente supervisados y otros métodos semisupervisados en un corpus de pruebas de resistencia. Para probar esta Hipótesis mido los resultados del experimento 8.1 que evalúa el impacto de usar un modelo de red de escaleras en la tarea de desambiguación de sentidos verbales del español. Estos resultados se comparan con los resultados de rendimiento de los algoritmos anteriores en el corpus de pruebas. La métrica para reportar los resultados es la puntuación F1 por clase. Como expliqué antes, mostraré los resultados entrenando la red de escaleras con 100 iteraciones y también con 25 iteraciones.

Como esto sólo se hace en el conjunto de datos de la prueba resistida, el rendimiento mostrado en esta sección se hace para la clase más frecuente, la segunda más frecuente y, si se aplica, la tercera más frecuente del lema. Recordemos que sólo dos de los seis

lemas tenían tres clases en el conjunto de datos de prueba: "llegar" y "pensar". Los otros cuatro lemas sólo tenían dos clases en el conjunto de datos.

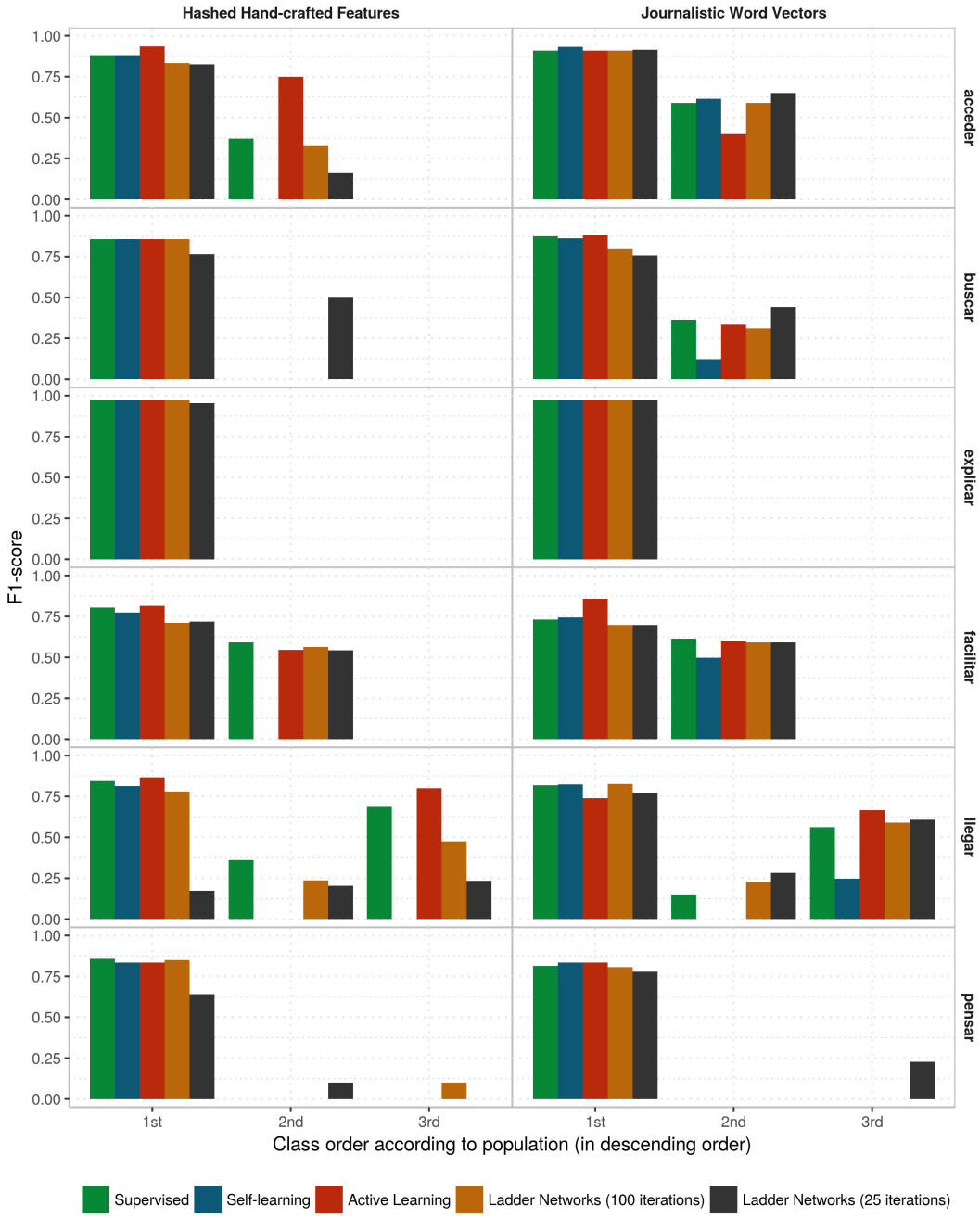


FIGURA 8.2: Comparación de los promedios macro y ponderado del F1-score para aprendizaje supervisado, autoaprendizaje, aprendizaje activo y redes en escalera (con 100 y 25 iteraciones respectivamente)

La figura 8.2 muestra la macro de la puntuación F1 y la media ponderada de las redes supervisadas, de autoaprendizaje, de aprendizaje activo y de escalera (usando 100 iteraciones) sobre el conjunto de datos de la prueba. En este caso, "supervisado" es la evaluación del modelo en la iteración inicial de cualquiera de los algoritmos de la envoltura (ya que es igual para ambos, es decir, sólo usando los datos etiquetados

manualmente). Las barras de redes de autoaprendizaje/aprendizaje activo/escalera representan el rendimiento del modelo sobre el conjunto de datos de prueba después de terminar las iteraciones del algoritmo correspondiente. La estructura del gráfico es la siguiente:

- Cada fila muestra los resultados para un lema testigo: "acceder", "buscar", "explicar", "facilitar", "llegar", y "pensar".
- Cada columna es un tipo de representación: atributos manuales y vectores de palabras periodísticos.
- Cada grupo de barras en cada gráfico representa una clase (i.e. un sentido) para ese lema. Estos están ordenado de acuerdo al número de ocurrencias de la clase en el conjunto de datos.
- Cada gráfico de barra de color distinto dentro de un grupo representa el algoritmo: supervisado (i.e. momento de evaluación de la iteración inicial), autoaprendizaje (i.e. momento de evaluación de la iteración final luego de que autoaprendizaje termine) y aprendizaje activo (i.e. momento de evaluación de la iteración final luego de finalizado aprendizaje activo), y dos verciones de las redes en escalera (i.e. para distintos momentos de parada en el algoritmo, 100 y 25 iteraciones respectivamente).
- La altura de la barra representa el valor de F1-score para cada clase.

Una vez más, recuerda que sólo las dos últimas filas del gráfico representan los lemas con 3 sentidos (es decir, "llegar" y "pensar"). Los primeros cuatro lemas pueden mostrar como máximo resultados para dos sentidos.

Tenga en cuenta que las redes de escaleras (en cualquier caso) funcionan igual o incluso mejor que el aprendizaje activo en la mayoría de los sentidos. Recuerde que hasta ahora el aprendizaje activo ha mostrado los mejores resultados, ya que puede apreciarse en la Figura.

Además, hay un par de casos en los que la red de escaleras (con 100 o 25 iteraciones) funciona mejor que el aprendizaje supervisado en una clase que el aprendizaje supervisado no reconoce en absoluto (por ejemplo, la tercera clase más frecuente del lema "pensar"). Por supuesto, esto puede suceder por casualidad, pero sigue siendo algo a considerar como redes de escalera, a diferencia del aprendizaje activo, es completamente automático (es decir, no hay participación humana). Además, ocurre de forma diferente en redes de escaleras que utilizan 100 iteraciones y 25 iteraciones.

Claramente, las redes de escaleras también son una mejor alternativa (o al menos tienen un mejor rendimiento) que el autoaprendizaje. Y también tiene un mejor rendimiento usando embeddings de palabras que usando características hechas a mano, como hemos visto a lo largo de esta tesis.

Nótese también que en general (especialmente para embeddings de palabras) detener redes de escaleras en 25 iteraciones tiene un mejor rendimiento que hacer las

100 iteraciones completas. Como mostraré más adelante, alrededor de 25 iteraciones comienzan la deriva hacia la clase más frecuente, lo que es una buena indicación de que esta es la razón por la que las redes de escaleras con sólo 25 iteraciones funcionan mejor que con 100 iteraciones. Nótese especialmente que las redes de escaleras con 25 iteraciones tienden a funcionar peor que las redes de escaleras con 100 iteraciones para el caso de la clase más frecuente. Esto es aún más evidencia de que detener el algoritmo cuando comienza a derivar a la clase más frecuente es una buena solución al problema de tener peor rendimiento para clases menos frecuentes.

En resumen, las redes de escaleras presentan algunos resultados impresionantes, especialmente en comparación con el aprendizaje activo y teniendo en cuenta que es un método puramente automático. Tiene resultados mucho mejores que el autoaprendizaje y parece que un diagnóstico más profundo sobre las iteraciones del algoritmo puede ayudar a equilibrar el rendimiento del clasificador en las clases además del más frecuente.

Aunque las redes de escaleras no son la mejor solución para todos los casos, está claro que al ser un método semi-supervisado que es completamente automático, mejora lo que otros métodos pueden lograr. Por ejemplo, el modelo tiene más información y, por tanto, más cobertura que un método puramente supervisado, y también evita el uso de un oráculo para el procedimiento de anotación. De estos resultados acepto la Hipótesis 5.1 de que las redes de escaleras mejoran con respecto a los métodos puramente supervisados u otros métodos semisupervisados. Se puede argumentar que, incluso si el rendimiento bruto del enfoque puramente supervisado y del enfoque de aprendizaje activo es comparable al de las redes de escaleras, estas últimas son menos costosas que el aprendizaje activo y en general hacen un mejor trabajo para mantener un buen rendimiento en todas las clases.

8.4.2 Hipótesis 5.2

Hipótesis 5.2 fue planeado para comparar cómo funcionan las redes de escaleras cuando se enfrentan a una tarea similar a la de los algoritmos de envoltura que exploré en los capítulos anteriores. La hipótesis establece que la representatividad de las clases si se utiliza el modelo para clasificar algunos datos no supervisados se mantendrá a través de las iteraciones. En particular, fue en la visualización de los resultados de este experimento que encontré que alrededor de la iteración número 25 había una deriva del algoritmo a la clase más frecuente. En esta sección hay dos gráficos de los mismos resultados, variando sólo en el número de iteraciones que el algoritmo de la red de escalera se da por terminado.

Al igual que para el autoaprendizaje y el aprendizaje activo en el capítulo anterior, para analizar estos resultados utilizo dos formas de visualización: una explora la distribución de las clases a lo largo de las iteraciones del algoritmo y la otra explora el conteo proporcional de cada clase añadida en cada iteración del algoritmo.

8.4.2.1 Distribución poblacional de las clases a través de las iteraciones

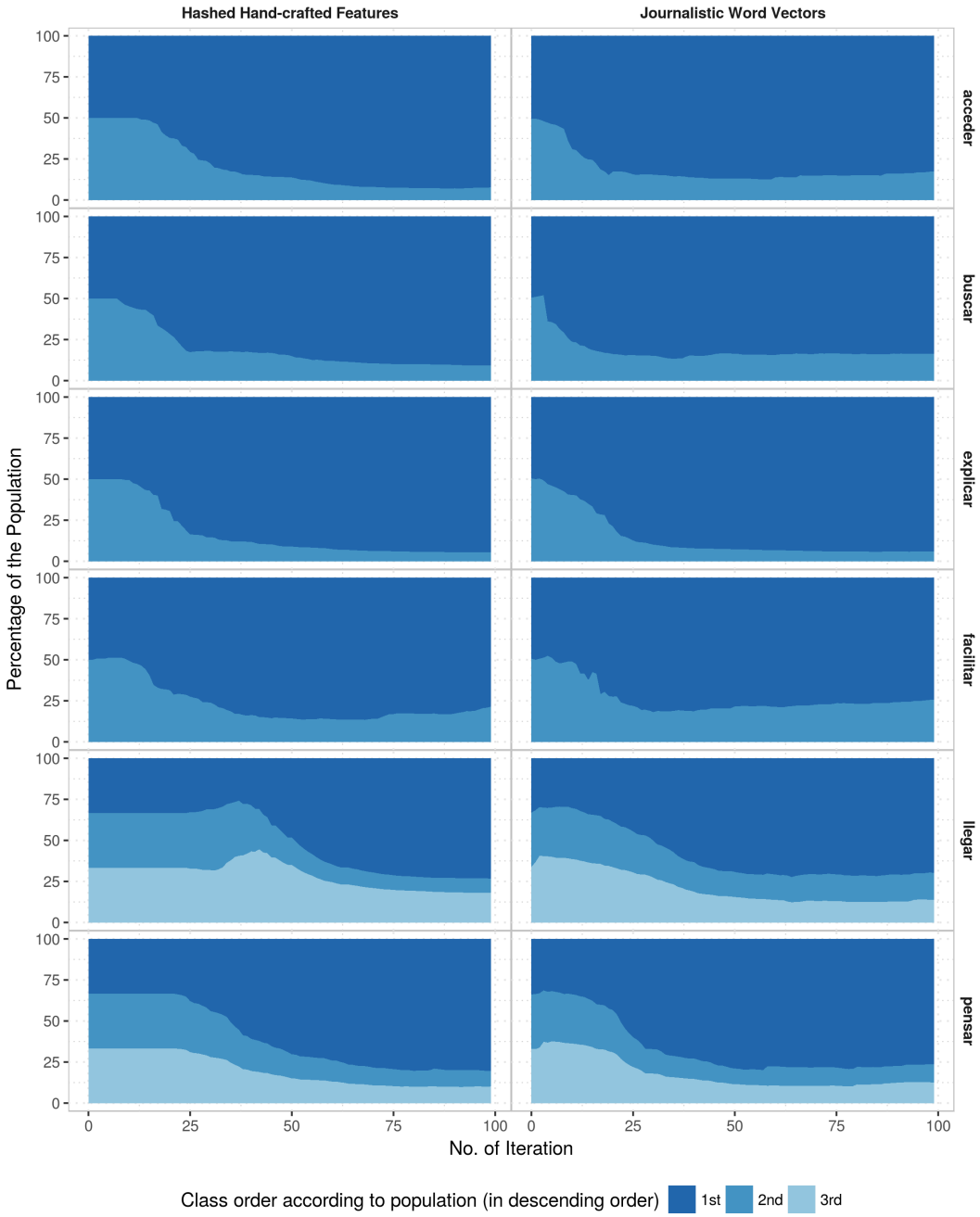


FIGURA 8.3: Distribución poblacional de las clases a través de las iteraciones del algoritmo de redes en escalera (con un máximo de 100 iteraciones) como una proporción de todo el conjunto de entrenamiento.

La figura 8.3 muestra la distribución de la población de las clases a través de iteraciones de red de escalera (con un máximo de 100 iteraciones) del algoritmo. La población de cada clase se representa como la proporción del número total de ejemplos en el conjunto de datos de formación para esa iteración. La parcela es una parcela de superficie apilada que sigue esta estructura:

- Cada fila muestra los resultados para un lema testigo: “acceder”, “buscar”, “explicar”, “facilitar”, “llegar”, y “pensar”.
- Cada columna es un tipo de representación: atributos manuales y vectores de palabras periodísticos.
- La coordenada x representa las iteraciones del algoritmo de redes en escalera.

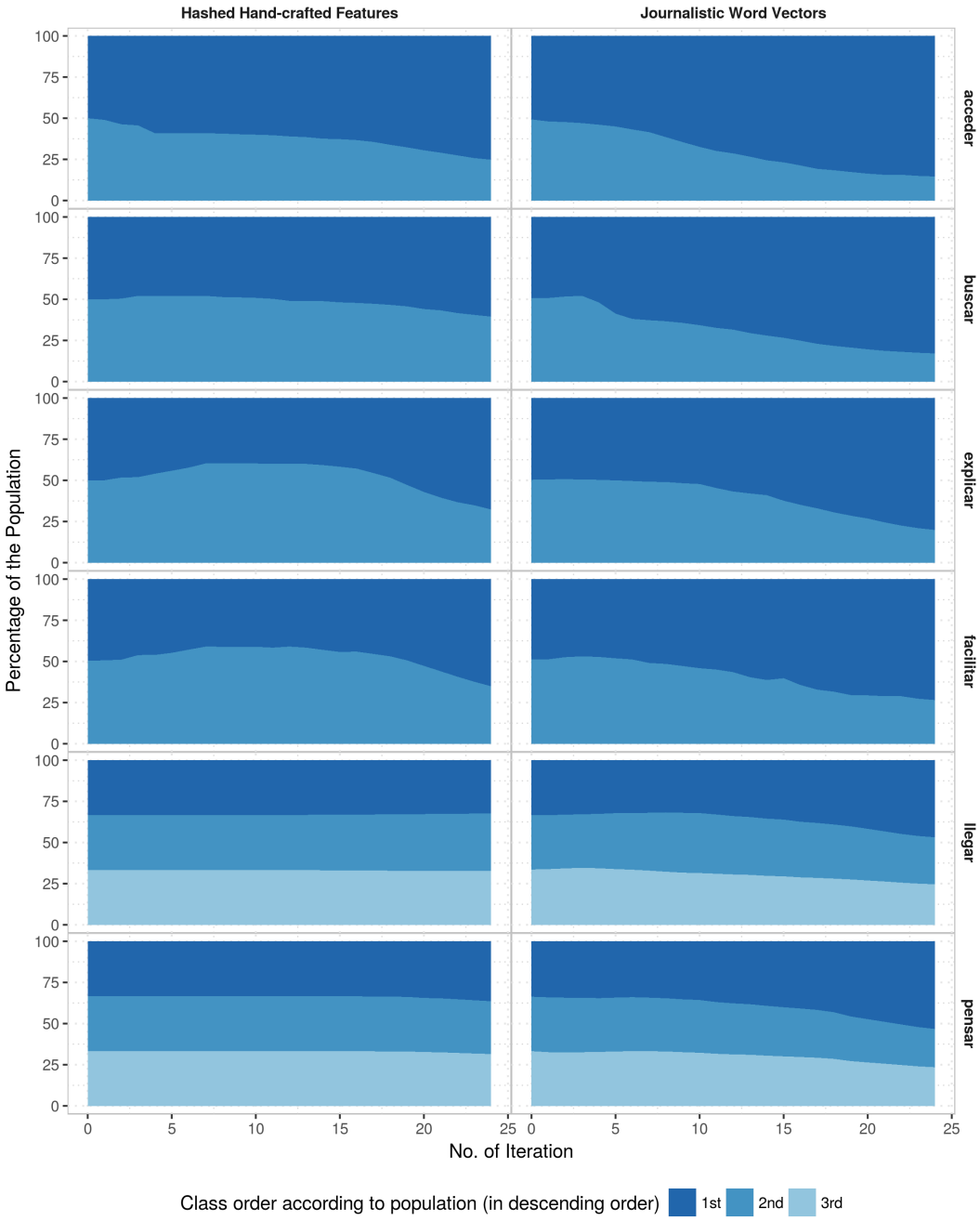


FIGURA 8.4: Distribución poblacional de las clases a través de las iteraciones del algoritmo de redes en escalera (con un máximo de 25 iteraciones) como una proporción de todo el conjunto de entrenamiento.

- La coordenada y representa el porcentaje de la población.
- Cada área de un color diferente representa la proporción de ejemplos para cada clase del conjunto de datos. Las clases nuevamente están ordenadas de acuerdo al número de ejemplos en el conjunto de datos supervisado original.

Como ya he explicado en secciones anteriores de este capítulo, es en torno a las iteraciones número 25 (aunque no en todos los casos, en algunos casos es mucho antes y en otros es más tarde) que la distribución de las clases desde las clases etiquetadas y anotadas automáticas del corpus sin etiquetar utilizado para la tarea (como se describe en el experimento 8.3) comienza a derivar hacia la clase más frecuente. Una vez más, la prevalencia de la clase más frecuente es más aguda para los rasgos artesanales que para las embeddings de palabras periodísticas.

Si decido detener el algoritmo por la iteración número 25, la distribución de clases se parece a la de la Figura 8.4. La figura tiene exactamente la misma estructura que la figura anterior con la única diferencia en el número máximo de iteraciones de entrenamiento del modelo de red de escaleras.

En este caso, sin embargo, la deriva del modelo para anotar automáticamente todo como la clase más frecuente no es tan fuerte como en la figura anterior. El algoritmo se detiene antes de que eso ocurra (aunque hay excepciones, después de todo, como todos, cada lemma tiene su propio conjunto de propiedades al que el algoritmo necesita adaptarse).

En cualquier caso, la deriva del modelo de red de escaleras a la clase más frecuente no es tan fuerte como en el autoaprendizaje, donde es prácticamente inmediato, justo después de la primera iteración. Esto bien puede ser una consecuencia de que la red de contactos aprenda el modelo a través de las iteraciones a medida que lentamente gana confianza en los límites de la decisión. A diferencia del autoaprendizaje, el uso de datos no etiquetados en este caso sirve para retrasar y evitar la deriva a la clase más frecuente.

Eventualmente, con más iteraciones, la clase más frecuente comienza a ganar más y más probabilidad, lo que se espera ya que los datos tienen una distribución en Zipfian. Sin embargo, el hecho de que los resultados del conjunto de datos del test sean mejores que los del autoaprendizaje parece indicar que la red de escaleras está clasificando correctamente los ejemplos de la clase más frecuente (mejor precisión), a diferencia del autoaprendizaje, que bien podría clasificarlos como clase mayoritaria sólo porque no puede distinguirlos de ninguna otra clase.

8.4.2.2 Población agregada por sentido por iteración

La figura 8.5 muestra la proporción de ejemplos añadidos por clase en cada iteración. Es un gráfico de barras apiladas donde cada barra representa el total de ejemplos añadidos en la iteración divididos por la proporción de clases anotadas automáticamente como tales. La estructura de la parcela es la siguiente:

- Cada fila muestra los resultados para un lema testigo: “acceder”, “buscar”, “explicar”, “facilitar”, “llegar”, y “pensar”.
- Cada columna es un tipo de representación: atributos manuales y vectores de palabras periodísticos.
- La coordenada x representa las iteraciones del algoritmo de redes en escalera.

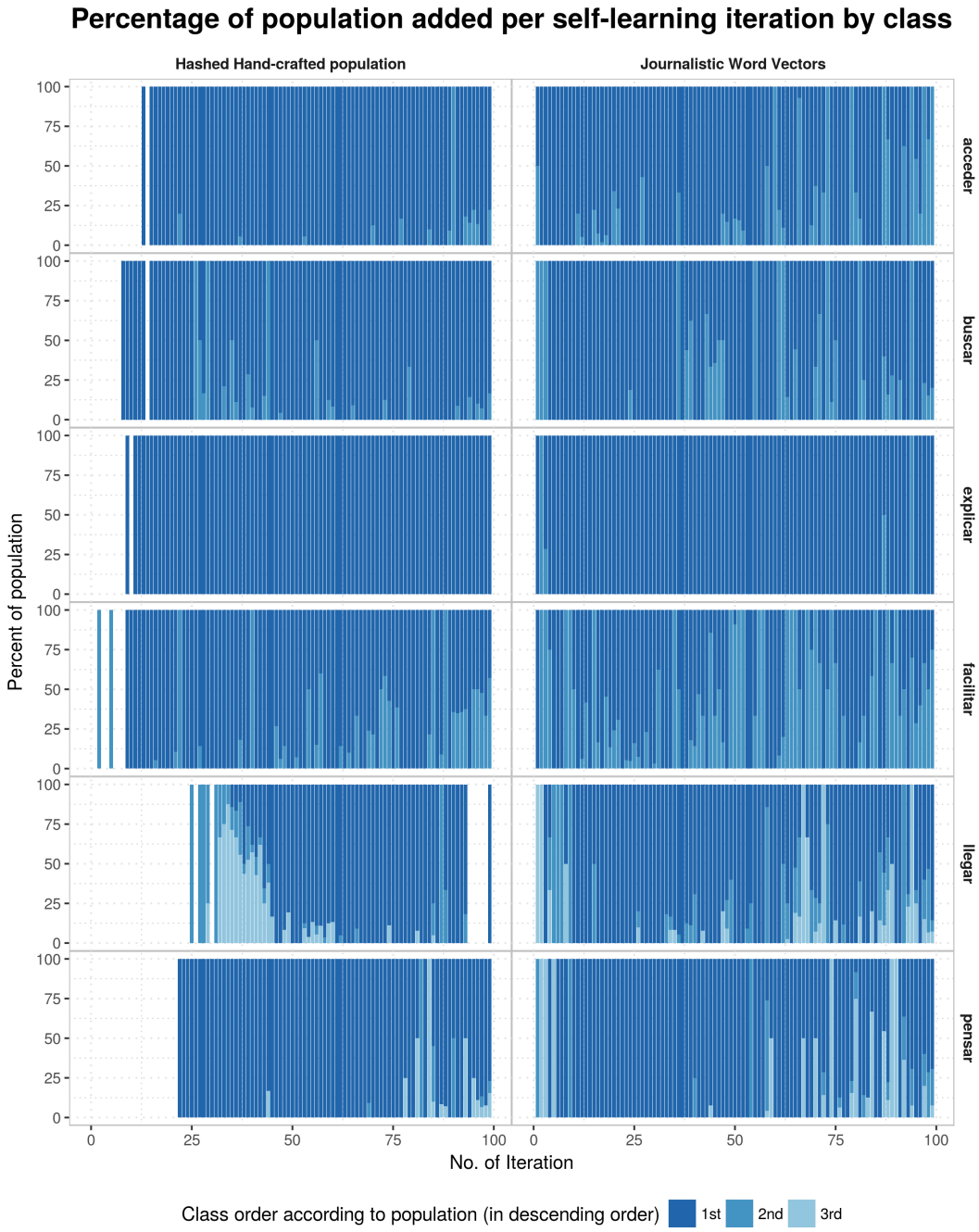


FIGURA 8.5: Población agregada para cada sentido en cada iteración de la red en escalera (con un máximo de 100 iteraciones) como la proporción de todos los ejemplos agregados en esa iteración).

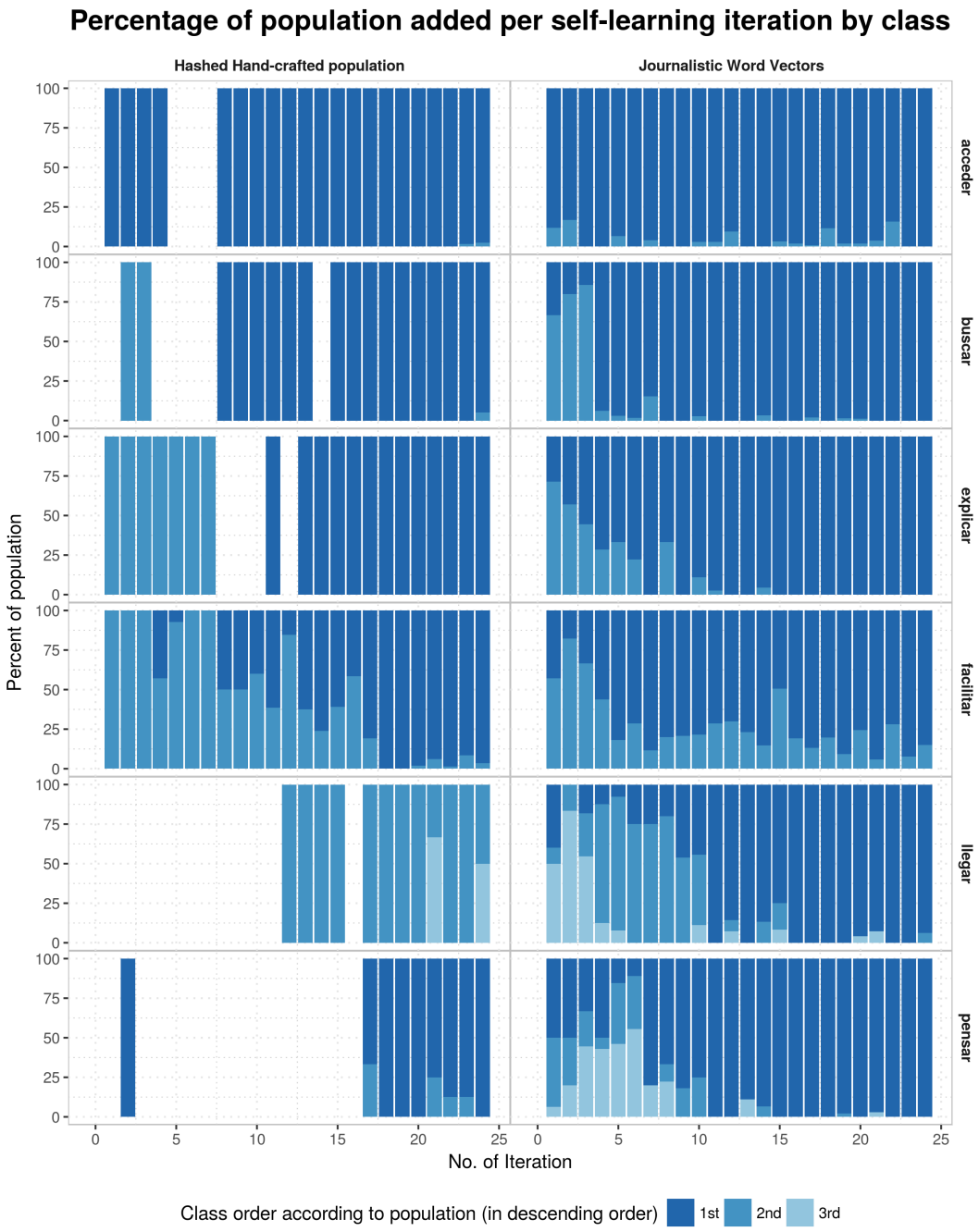


FIGURA 8.6: Población agregada para cada sentido en cada iteración de la red en escalera (con un máximo de 25 iteraciones) como la proporción de todos los ejemplos agregados en esa iteración).

- La coordenada y representa el porcentaje de ejemplos anotados automáticamente y agregados al modelo.
- Cada barra representa la distribución de ejemplos agregados in esa iteración. Cada color de la barra representa la clase con la cual fueron anotados los ejemplos.

Del mismo modo, la figura 8.6 muestra la misma información pero esta vez para el algoritmo de la red de la escalera que se detiene en 25 iteraciones.

Lo primero que hay que señalar de estas cifras es que, en este caso, a diferencia de lo que ocurre en las mismas cifras de capítulos anteriores, en algunos casos faltan barras. Esto se debe a que, a diferencia del autoaprendizaje, para el algoritmo de red de escalera no es obligatorio anotar ejemplos del conjunto de datos sin etiquetar. Así, en las primeras iteraciones, donde el algoritmo generalmente no anota nada, es porque el modelo no está lo suficientemente seguro acerca de las instancias anotadas.

Esto es consecuencia de que el modelo no sólo aprende del corpus etiquetado, sino también del corpus no etiquetado. El modelo tiene menos certeza a medida que los datos no etiquetados, que se añaden en lotes, añaden información que evita que el modelo converja, pero también que se sobreponga al conjunto de datos supervisados.

Hay algunos lemas en los que el modelo tiene más certeza que otros. En particular, el lema "llegar", para la representación de los rasgos artesanales, es el que más aprovecha el modelo para empezar a añadir ejemplos. Esto significa que el modelo no puede manejar la información del lema dada por los datos no supervisados en las primeras iteraciones y por lo tanto toma más tiempo para ajustar los parámetros a ese lema. Un rápido vistazo a Figure 8.2 muestra que es precisamente para "llegar" que, a 25 iteraciones, el algoritmo de red de la escalera muestra el peor rendimiento. Además, un rápido vistazo a la Figura 7.2 en el capítulo anterior muestra que "llegar" tiene el mayor número de sentidos diferentes que ocurren en el conjunto de datos sin etiquetar. Además, fue el ejemplo que utilicé cuando introduje el autoaprendizaje en el capítulo 6 para ejemplificar un lema que tiene un sentido tan frecuente en el corpus SenSem pero que no es precisamente el mismo que en un corpus general (por lo tanto, es sesgado de dominio). Recuerde que las instancias no etiquetadas se toman de un corpus de dominio general.

El lema "llegar" es un caso particular, y vemos que el uso de rasgos artesanales socava el rendimiento del modelo sobre un lema que está tan sesgado. Sin embargo, es importante notar que este no es el caso para todos los lemas o incluso que lo mismo sucede con una representación menos dependiente del dominio, como es la que proporcionan los embeddings de palabras.

Siguiendo las cifras, tanto para 100 como para 25 iteraciones, queda claro que en la mayoría de los casos la clase más frecuente empieza a dominar los ejemplos anotados automáticamente. También es importante notar cómo esto es más claro para las características hechas a mano. Por lo tanto, la idea de detener las iteraciones una vez que las redes de escaleras empiecen a flotar no es tan injustificada después de todo, especialmente teniendo en cuenta los resultados dados en la sección anterior. Sin embargo, es importante notar algo: en el primer par de iteraciones hay más instancias de las clases menos frecuentes, algo que se va volviendo gradualmente. Sin embargo, es más notorio en las características artesanales que la distribución es todo o nada, ya que las barras comienzan a ser de un color y luego de repente se convierten en el otro color. Esto no ocurre con los embeddings de palabras. En ese caso, la representatividad

de cada una de las clases es más uniforme y más similar a la distribución original de las clases (es decir, la clase más frecuente sigue siendo la más frecuente, pero no la única). Dado lo que he discutido en este trabajo de tesis, es seguro asumir que es una consecuencia directa de que los embeddings de palabras pueden proporcionar una mejor generalización a un modelo.

De los resultados mostrados en esta y en la sección anterior, hay suficiente evidencia para aceptar la hipótesis 5.2 para embeddings de palabras. Sin embargo, no hay pruebas suficientes para aceptarlo para los rasgos hechos a mano.

8.4.3 Hipótesis 5.3

La hipótesis final del capítulo explora cómo los datos no etiquetados ayudan al modelo de red de escaleras a reducir la tendencia a la sobreadaptación. Hipótesis 5.3 establece que el exceso del modelo de red de escaleras sobre los datos etiquetados se evita mediante el uso de datos no etiquetados para calcular la función de coste global, que se compone de las funciones de coste etiquetadas y no etiquetadas.

Es importante notar que los experimentos y resultados mostrados en esta sección capturan el espíritu de los experimentos de los capítulos anteriores, los cuales también exploran cómo la adición de datos a un modelo impacta en la tendencia a sobredimensionarse. Sin embargo, lo que podría lograr aquí no se compara directamente con lo que hice con otros algoritmos, como el autoaprendizaje o incluso los métodos supervisados. Esto se debe a que, a diferencia de lo que sucede en esos casos, el algoritmo de las redes de escalera integra la información del conjunto de datos sin etiquetar de una manera diferente a la de los métodos anteriores. En este caso, la información no supervisada se integra en los pesos de la red.

La figura 8.7 muestra el gráfico de la curva de aprendizaje en función del número de ejemplos del conjunto de datos de formación. La estructura de la curva de aprendizaje es la siguiente:

- Cada fila muestra los resultados para un lema testigo: “acceder”, “buscar”, “explicar”, “facilitar”, “llegar”, y “pensar”.
- Cada columna es un tipo de representación: atributos manuales y vectores de palabras periodísticos.
- La coordenada x representa el número de ejemplos no anotados agregados en cada iteración por la red en escalera.
- La coordenada y representa el error de clasificación.
- Hay dos colores representando los conjuntos de datos: entrenamiento y validación (en este caso es el conjunto de validación).
- La línea sólida más oscura representa la media del error de clasificación a través de las distintas iteraciones de los conjuntos de datos sobre todos los modelos.

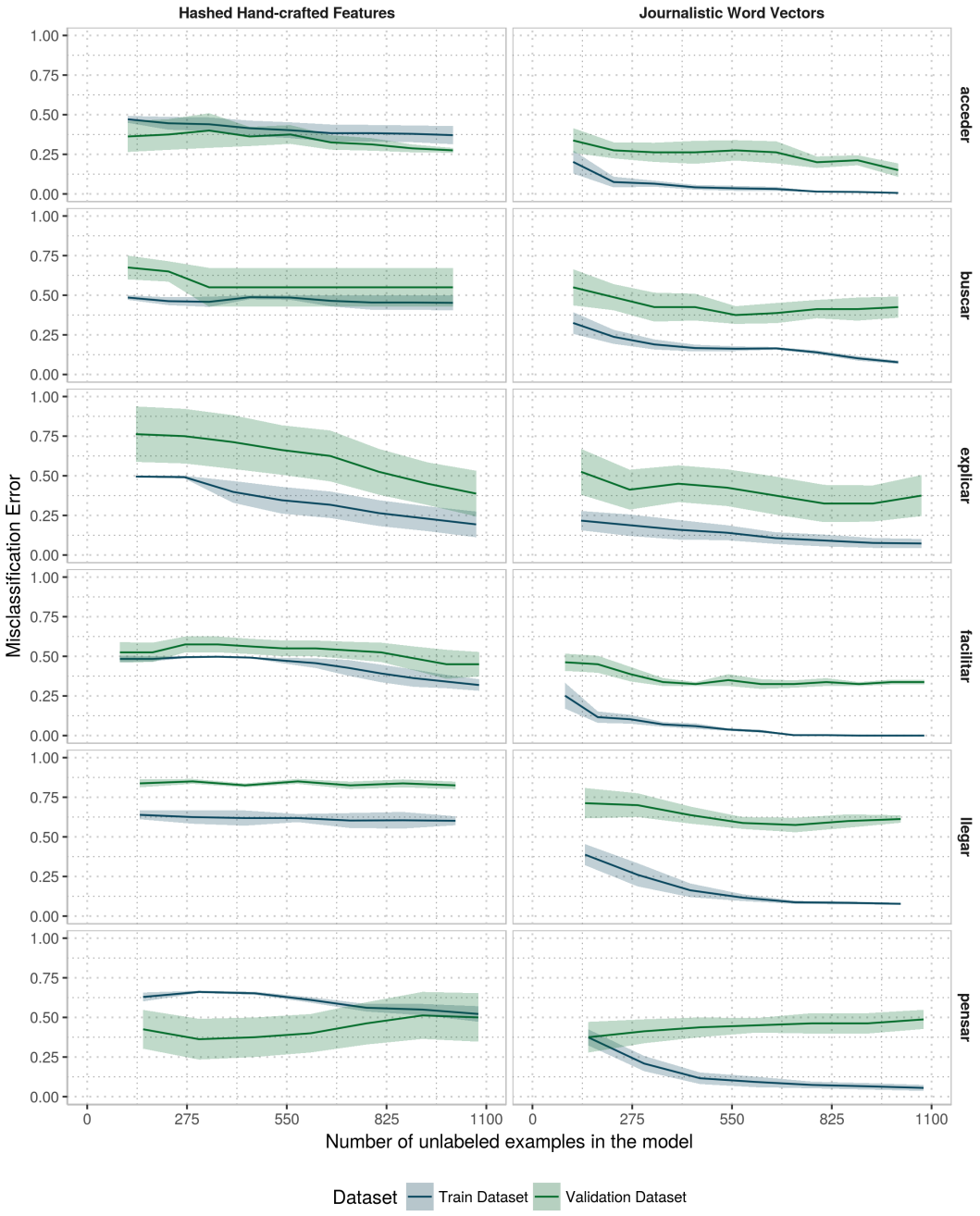


FIGURA 8.7: Curva de aprendizaje como función del número de ejemplos no anotados agregados por la red en escalera en cada iteración del algoritmo

- El área sombreada, con un color más claro, representa el error estándar de la media del error de clasificación.

Recuerde que en este caso, el Experimento se detiene una vez que todo el conjunto de datos sin etiquetar es atravesado una vez. Como el tamaño del lote es igual al número de instancias etiquetadas, que son alrededor de 100 instancias por lema y el número máximo de datos sin etiquetar es 1000, entonces el algoritmo se detiene

cerca de la iteración número 10, porque ya cubre todos los datos sin etiquetar por esa iteración.

La figura muestra una imagen muy diferente de lo que han mostrado hasta ahora diagramas similares para otros enfoques. A diferencia de lo que ocurrió con los métodos supervisados o semisupervisados, el error de clasificación errónea del conjunto de datos de formación no se acerca a cero esta vez. En este caso, el error de clasificación disminuye lentamente y en la mayoría de los casos se acompaña del error de validación. Sin embargo, hay más errores debido a la varianza (representada por el área sombreada más amplia) del conjunto de datos de validación. Sin embargo, parece estar disminuyendo lentamente a medida que se añaden más ejemplos no etiquetados al modelo. Además, excepto quizás en los dos últimos lemas, el error de validación converge lentamente también en el error de entrenamiento.

Las características hechas a mano parecen tener un error mayor debido a la varianza que los embeddings de palabras. Como se ve a través de esta tesis, esto parece un producto de la dificultad de los rasgos hechos a mano para generalizar sobre su sesgo de dominio. Los embeddings de palabras, al ser más suaves, modelan los datos de una manera más generalizada.

Es importante notar que, sin importar la representación, la progresión del error para la mayoría de los casos es bastante uniforme, es decir, incluso si el error de entrenamiento es menor que el error de validación, la forma en que ambos progresan es similar. Además, como el modelo de red de escalera está añadiendo datos no etiquetados para ayudar a la red a generalizarse mejor, a partir de estos resultados parece que en muchos casos lo hace precisamente evitando una gran caída en el error de los datos de formación mientras que tiene un error mayor de los datos de validación. Una vez más, esto también depende del lema, ya que cada lema tiene su propio conjunto de propiedades. Estos experimentos se realizaron en un corpus limitado sin etiquetar debido a limitaciones de tiempo y recursos, una línea de trabajo futura sería explorar la curva de aprendizaje del algoritmo con un número mucho mayor de ejemplos no anotados.

La evidencia mostrada por estos resultados es suficiente para aceptar la Hipótesis 5.3 de que el uso de datos no etiquetados impide que el modelo se ajuste demasiado.

8.5 Conclusiones

En este capítulo presenté un método semi-supervisado conjunto que, que yo sepa, no tenía aplicaciones previas en el área de desambiguación de sentidos: la red de escaleras.

Los experimentos y resultados mostrados en este capítulo han demostrado ser muy interesantes por sus implicaciones y sus posibilidades en el ámbito del desambiguación de sentidos verbales del español, pero también como un método general semi-supervisado, aplicable a muchas áreas diferentes.

The ladder network is a semi-supervised algorithm that uses a cost function which is a combination of the supervised cost function given by a feed-forward neural network

(in this case a multilayer perceptron) and an unsupervised cost function that results in the layer by layer reconstruction of an autoencoder. The first is used to train on labeled data while the second is used to train on unlabeled data. The use of a cost functions that has an unsupervised part in the training of the classifier helps smoothing the fitting of the labeled data in the same network.

La red de escalera es un algoritmo semi-supervisado que utiliza una función de coste que es una combinación de la función de coste supervisado dada por una red neuronal de retroalimentación (en este caso un perceptrón multicapa) y una función de coste no supervisado que resulta en la reconstrucción capa por capa de un autoencoder. El primero se usa para entrenar con datos etiquetados mientras que el segundo se usa para entrenar con datos no etiquetados. El uso de funciones de coste que tienen una parte no supervisada en la formación del clasificador ayuda a facilitar el ajuste de los datos etiquetados en la misma red.

Como consecuencia, la red de escaleras mejora otros métodos al superar algunas de las deficiencias que tenían. Al integrar un conjunto de datos sin etiquetar, tiene más cobertura que un método puramente supervisado. También tiene mejor rendimiento que el autoaprendizaje porque mantiene bajo control el sesgo de la clase más frecuente. Por último, es más barato que el aprendizaje activo al evitar la necesidad de que un anotador humano añada nuevos ejemplos al conjunto de datos de formación.

Hipótesis 5.1 establece que el modelo de red de escaleras mejora en comparación con los métodos puramente supervisados y otros métodos semisupervisados. La hipótesis es aceptada como se muestra en los resultados de la Sección 8.4.1. Los experimentos reportaron que la red de escaleras efectivamente alcanzó el desempeño de otros métodos como el aprendizaje activo o puramente supervisado, y en algunos casos incluso los superó. Lo más importante de estos resultados fue ver cómo las redes de escaleras podían sacrificar algo de rendimiento en la clase más frecuente para representar mejor las clases menos frecuentes dadas las condiciones adecuadas (es decir, detenerse en 25 iteraciones). Además, las redes de escaleras podrían funcionar mejor en sentidos que el enfoque supervisado no podría reconocer en absoluto en el corpus de pruebas. Incluso si la red de escaleras no puede funcionar tan bien como el aprendizaje activo en algunos casos, es importante notar que las redes de escaleras no dependen de un humano para la parte no supervisada.

La hipótesis fue parcialmente aceptada. La hipótesis establecía que si el modelo se utilizaba para anotar automáticamente las instancias de un corpus sin etiquetar, se mantendría la representatividad de las clases en esas instancias. Los resultados de Section 8.4.2 muestran cómo evolucionó la distribución de las clases a través de iteraciones. En estos resultados, el número de iteraciones también tuvo un efecto importante en el resultado, ya que expliqué que cuando se ejecutó el algoritmo para 100 iteraciones fue alrededor de la iteración número 25 que el modelo comenzó a derivar a la clase más frecuente. Por eso decidí truncar los modelos en 25 iteraciones y comparar los resultados. Estos resultados mostraron que para las características artesanales la Hipótesis no podía ser aceptada ya que era demasiado extrema. En

cada iteración, el modelo de características artesanales marcaría todos los ejemplos no etiquetados como parte de una sola clase, primero las clases menos frecuentes y finalmente la más frecuente. Sin embargo, el modelo basado en embeddings de palabras mostró mejores resultados en línea con lo que Hipótesis 5.2 indicaba. En este último, la representatividad de las clases se mantuvo de manera más uniforme a través de las iteraciones. Una línea de trabajo futuro sería utilizar la diferencia entre la distribución del corpus original y la distribución del lote previsto como criterio de parada, similar al trabajo de Zhao et al. [Zhao et al., 2017].

Hipótesis 5.3 estableció que el uso de datos no etiquetados ayuda al modelo de red de escaleras a reducir la tendencia a sobredimensionar. Discutimos los resultados de los experimentos con respecto a esta hipótesis en la Sección 8.4.3. En los capítulos anteriores exploré cómo evolucionó la curva de aprendizaje añadiendo datos etiquetados al modelo. En este caso, sin embargo, decidí explorar cómo la adición de datos no etiquetados al modelo repercute en la tendencia a modificar el modelo. Aunque no sea directamente comparable a lo que he visto en capítulos anteriores, la idea básica de ver la tendencia al overfit es la misma: cómo la nueva información sobre el modelo afecta a la forma en que se toman las decisiones. Los resultados mostraron que la curva de aprendizaje se mantuvo uniforme mientras que los datos no etiquetados se agregaron al modelo. Tanto el error de entrenamiento como el error de validación disminuyeron lentamente con el tiempo en la mayoría de los lemas, y el error debido a la varianza también se redujo.

Los resultados de este capítulo muestran el potencial de las redes de escaleras como un enfoque semi-supervisado, no sólo para desambiguación de sentidos verbales del español, sino como un método en general. Todavía queda trabajo por hacer con respecto a este método.

En el trabajo futuro, la primera línea sería explorar cómo evoluciona la curva de aprendizaje cuando no se limita a un número limitado de instancias no etiquetadas. El enfoque ideal sería tener un método de aprendizaje en línea [Bottou, 1998] en el que los nuevos datos no supervisados pudieran añadirse indefinidamente.

Otra línea de trabajo sería hacer una evaluación manual de las instancias anotadas automáticamente y compararlas con lo que hace el autoaprendizaje. Por último, como este método es un algoritmo por derecho propio, podría utilizarse en algunos de los métodos de envoltura anteriores. Sería interesante entonces ver cómo evolucionaría una combinación de la red de escaleras envuelta por el aprendizaje activo.

En cuanto a la personalización del algoritmo de la red de escaleras, hay mucho trabajo futuro por hacer: la exploración de diferentes tipos de funciones de combinación, el uso de otros tipos de redes neuronales como recurrentes o convolucionales, y el diseño de un enfoque más integral que sostiene la red de escaleras como

Parte IV

Conclusiones

Capítulo 9

Conclusiones

9.1 Contribuciones

La experimentación de esta tesis mostró que la falta de datos produce una tendencia a encajar en modelos de aprendizaje puramente supervisados, junto con una pequeña cobertura. Para abordar estos problemas exploré diferentes métodos semi-supervisados, cada uno con ventajas y desventajas propias. Se encontró que el modelo de *redes neuronales en escalera* era el más prometedor en términos de aprovechar los datos no etiquetados para mejorar el rendimiento de los modelos puramente supervisados.

Mi objetivo propuesto para esta tesis era estudiar cómo las diferentes técnicas semi-supervisadas podrían mejorar una tarea que se beneficiaría enormemente de ellas. A lo largo de los experimentos y resultados de este trabajo investigué las propiedades, los beneficios y las deficiencias de los diferentes enfoques semi-supervisados en una tarea en particular. Para seleccionar la tarea decidí explorar un dominio que efectivamente encontraría útil el uso de una técnica semi-supervisada. No quería utilizar un conjunto de datos de juguetes en el que se aplicarían las propiedades del aprendizaje semisupervisado sólo porque se diseñó para obtener tales resultados. Lo que me interesaba era una tarea que tuviera su propio conjunto de desafíos y, en particular, una tarea que ya tuviera la configuración ideal que hiciera del aprendizaje semi-supervisado una solución lógica: la tarea debería tener una pequeña cantidad de datos etiquetados, lo suficientemente buena para una solución de línea de base, y una gran cantidad de datos sin etiquetar.

La desambiguación de sentidos, como se discutió en la introducción de este trabajo, es una tarea fundamental en el campo del procesamiento del lenguaje natural. Es lo que se conoce como tarea intermedia, necesaria para tareas más complejas como la traducción automática o la extracción de información. En particular, la tarea de desambiguación de sentidos verbaleses útil en la extracción de relaciones, ya que un verbo es la pieza léxica que establece las relaciones entre los participantes en una oración. Mi interés particular dentro del área de desambiguación de sentidos verbalesfue el área de desambiguación de sentidos verbales del español. En esta área ha habido sólo un trabajo previo mínimo, ya que la mayor parte del trabajo en desambiguación es para sustantivos. En particular, me interesaba la tarea de desambiguación de sentidos verbales del español por la disponibilidad del corpus SenSem, que es un corpus desambiguado manualmente para los verbos españoles. Este recurso proporciona una

línea de base supervisada y la semilla inicial necesaria para los métodos semisupervisados. Por otra parte, los recursos disponibles para los datos no etiquetados son más que suficientes para la parte no supervisada de las tareas semisupervisadas.

El capítulo 4 comienza explorando los diferentes algoritmos supervisados para enfocar sus defectos y planificar cómo podría abordarlos como desafíos. El capítulo hace una investigación sobre diferentes técnicas usando lo que yo llamé *atributos manuales*, es decir, características tomadas del propio corpus etiquetado. En primer lugar, el objetivo principal del capítulo era establecer una base común para la experimentación evitando una explosión exponencial de posibles experimentos y resultados de los que habría sido imposible analizar y sacar conclusiones. Este punto de partida se estableció para explorar diferentes maneras de representar los datos y reducir los recursos consumidos por esas representaciones utilizando técnicas de reducción de la dimensionalidad. El capítulo también explora diferentes clasificadores, lineales y no lineales, para descartar grandes diferencias antes de seleccionar uno de ellos. Una vez establecida la estructura base de los experimentos, el capítulo explora cómo el número de datos etiquetados en un modelo puede impactar en el rendimiento final del modelo. Lo que los resultados de la experimentación muestran es que el principal problema del enfoque supervisado es la falta de datos etiquetados para entrenar un buen modelo que pueda generalizar bien. En particular, el modelo se entrena sobre un conjunto de datos etiquetado de un dominio específico, lo que empeora aún más el desafío de la sobreadaptación. Por otro lado, el modelo tiene poca información: la cobertura se ve muy afectada por los pocos ejemplos disponibles para capacitar. Una vez que los resultados de este capítulo fueron claros, en capítulos sucesivos exploré el impacto de las diferentes técnicas de aprendizaje semisupervisadas sobre estos dos desafíos: la adaptación del modelo en el conjunto de datos de capacitación, y la cobertura del modelo mediante la adición de más información de nuevos ejemplos.

En el capítulo 5 se introdujo el uso de *word embeddings* como una representación alternativa de las instancias utilizadas para entrenar al clasificador. Esto se conoce como *aprendizaje semisupervisado disjunto*, donde previamente se realiza una tarea no supervisada (es decir, embeddings de palabras de entrenamiento) y el resultado de esta tarea se integra en una tarea supervisada (es decir, desambiguación de sentidos verbales). Este capítulo explora cómo el uso de embeddings de palabras impacta en un clasificador supervisado. En particular, el capítulo experimentó con diferentes tipos de embeddings de palabras entrenadas con diferentes dominios. Los embeddings formados en el mismo dominio que el conjunto de datos etiquetado (es decir, el dominio periodístico) mejoran el rendimiento del clasificador supervisado. Sin embargo, el clasificador supervisado entrenado con representaciones de embeddings de palabras no alcanzó los niveles de rendimiento del modelo entrenado con características hechas a mano. Aunque esto podría interpretarse como un problema, los capítulos sucesivos muestran lo contrario. El modelo puramente supervisado obtuvo mejores cifras de rendimiento que el modelo basado en embeddings de palabras porque las características hechas a mano estaban más cerca de los datos, por lo que tenía una mayor

tendencia a sobredimensionarse. Otros experimentos en este capítulo y el análisis de los resultados mostraron que, en contraste, los embeddings de palabras son una buena representación para reducir la tendencia del modelo a sobredimensionar los datos de entrenamiento. Esto está más fuertemente apoyado por la evidencia en el Capítulo 6, con experimentos en un corpus más general.

El capítulo 6 fue el primero en introducir un algoritmo de *aprendizaje conjunto* en el que tanto los datos etiquetados como los no etiquetados contribuyen a la tarea de aprendizaje semi-supervisado. El capítulo explora el autoaprendizaje, un algoritmo de envoltura sobre un clasificador supervisado. En este esquema, el conjunto de datos etiquetado sirve como una semilla inicial para entrenar un modelo base para un clasificador. Este clasificador se utiliza para aumentar la información disponible en el modelo mediante la anotación automática de nuevas instancias a partir de un conjunto de datos sin etiquetar basado en la certeza y su adición como instancias etiquetadas para formar un nuevo modelo. Este proceso se repite iterativamente para que cada vez más ejemplos de capacitación se tomen de los datos no etiquetados. Los experimentos en este capítulo exploraron cómo estos nuevos datos afectaron el desempeño del modelo. En particular, quise ver cómo los nuevos datos aumentaron efectivamente el modelo inicial con más información recopilada de las nuevas instancias. Esta nueva información podría ayudar a ampliar la cobertura del modelo. La adición de nuevos ejemplos de un nuevo conjunto de datos también ayudaría al modelo a prevenir la tendencia a sobredimensionarse. Sin embargo, lo que encontré fue que la expansión de la cobertura del modelo se hizo a expensas de la calidad del modelo. El algoritmo de autoaprendizaje tenía problemas especialmente cuando se trataba de un conjunto de datos muy desequilibrado, es decir, con una clase claramente mayoritaria. Esta configuración es común en una tarea como la de desambiguación de sentidos verbales del español, así como para cualquier tarea de procesamiento de lenguaje natural, debido a la distribución del lenguaje natural en Zipfian. Entonces, mientras más información parece ser agregada al modelo, lo que realmente está sucediendo es que el modelo está a la deriva para clasificar nuevos datos casi exclusivamente como parte de la clase más frecuente en el conjunto de datos. Entonces, el modelo no amplió su cobertura, sino que simplemente difuminó los límites de decisión del clasificador supervisado original. A partir de este capítulo, sin embargo, hubo un importante hallazgo dado por los resultados generales y el rendimiento obtenido con los embeddings de palabras. Se comprobó que estas últimas se comportan mucho mejor que las características hechas a mano. Representaban mejor a las clases minoritarias, mitigando la deriva hacia la clase mayoritaria. Como son entrenados desde un dominio más general que las características hechas a mano, también trataron mejor con el cambio de dominio dado por las instancias no etiquetadas del algoritmo de auto-aprendizaje usado para expandir el modelo supervisado.

El capítulo 7 exploró una segunda técnica de aprendizaje conjunto: *aprendizaje activo*. Al igual que el autoaprendizaje, también es un algoritmo de envoltura. Se entrena sobre la base de una semilla inicial y luego agrega ejemplos de un conjunto no

etiquetado de instancias disponibles. Estas instancias sirven para aumentar el conjunto de ejemplos etiquetados que se utilizan para capacitar a un nuevo modelo que se utiliza de nuevo para obtener nuevos datos para agregar. La principal diferencia entre este algoritmo y el autoaprendizaje es la forma en que se anota el conjunto de datos sin etiquetar. En el autoaprendizaje, la anotación es automática, y las instancias anotadas automáticamente se incluyen como formación si el clasificador tiene una gran confianza en la instancia. En contraste, el aprendizaje activo utiliza a un humano (generalmente un experto en el dominio) para anotar ejemplos no etiquetados seleccionados de manera "inteligente". Esto significa que los casos deben ser aquellos que, una vez anotados, tienen el mayor impacto en el nuevo modelo, aumentando su certeza o reduciendo el error. La experimentación realizada para este capítulo fue meramente exploratoria, porque el enfoque de aprendizaje activo es en sí mismo un método costoso que requiere anotaciones manuales. Sin embargo, los resultados fueron lo suficientemente prometedores como para tener algunas ideas sobre cómo este método podría ayudar a mejorar la tarea de desambiguación de sentidos verbales del español. Los resultados del capítulo fueron interesantes porque el modelo no sólo mejoró en las clases más frecuentes sino también en las clases menos frecuentes. Además, debido a la forma en que el algoritmo selecciona las instancias para la anotación, no sufrió la deriva a la clase más frecuente que mostró el autoaprendizaje. De hecho, se demostró que el aprendizaje activo aumentaba la información del modelo con menos ejemplos proporcionados en comparación. Pero, como requiere un anotador humano para funcionar, el aprendizaje activo se convierte en un enfoque costoso para el aprendizaje semi-supervisado.

Finalmente, el capítulo 8 exploró una técnica de aprendizaje semi-supervisado bastante nueva: *la red neuronal en escalera*. En esta red neuronal, la arquitectura está diseñada para entrenar los parámetros optimizando una función de coste que es una combinación de una función de coste supervisado y una función de coste no supervisado. La red neuronal tiene una arquitectura que se asemeja a la de un autoencoder, con un encoder y una ruta de decodificación. La ruta del codificador viene dada por una red neuronal de alimentación con una capa de salida que se entrena utilizando una función de coste supervisado. La ruta del decodificador se utiliza para reconstruir las instancias de entrenamiento y, por lo tanto, ayuda a suavizar la red neuronal al evitar el sobreequipamiento de un pequeño conjunto de datos supervisado. En este modelo, los ejemplos no etiquetados no se etiquetan ni se añaden al modelo como en los algoritmos anteriores. Sin embargo, las instancias no etiquetadas todavía ayudan a expandir la cobertura del modelo, ya que la información que proporcionan se integra a la del conjunto de datos etiquetado. Los experimentos en este capítulo exploran el uso de redes de escaleras como un enfoque alternativo semi-supervisado que supera la deficiencia del autoaprendizaje que añade nuevas instancias sólo como si fueran parte de la clase más frecuente. Además, el modelo de red de escaleras supera el problema del aprendizaje activo, es decir, el costo de la anotación manual. Los resultados del capítulo fueron prometedores, ya que la red de escaleras tuvo buenos resultados, no

sólo en la clase más frecuente sino en todas las clases en general. Muy a menudo superó los resultados obtenidos por un enfoque puramente supervisado o por un aprendizaje activo. El uso de datos no etiquetados para ayudar a la formación del modelo añadió más información que ayudaría a ampliar la cobertura del modelo. Además, el modelo demostró ser capaz de evitar el sobreequipamiento, ya que los ejemplos no etiquetados y la función de coste no supervisada impedían que la pieza supervisada sobreequipara los datos etiquetados. Las redes de escaleras mostraron resultados impresionantes que provienen de un proceso puramente automático para el aprendizaje semi-supervisado y es un buen candidato para seguir explorando.

9.2 Trabajo futuro

El principal obstáculo a través de los experimentos realizados en esta tesis fue el conjunto de datos desequilibrado. La presencia de una clase mayoritaria que desequilibra la distribución de las clases es algo común para una tarea en el procesamiento del lenguaje natural. Este desequilibrio tuvo un gran impacto en la forma en que las técnicas semi-supervisadas podían funcionar en la tarea de desambiguación de sentidos verbales del español. Como he visto en el capítulo 6, el problema del desequilibrio, aunque al principio fuera pequeño, puede crecer rápidamente y acabar afectando a la calidad del algoritmo, haciendo inútil el modelo final. Las clases desequilibradas son un problema en sí mismo, que no podría cubrir más a fondo en este trabajo. Existen técnicas, además del sobremuestreo o submuestreo, que ayudan a hacer frente a este tipo de situaciones y que requerirían una mayor exploración en trabajos futuros. Sin embargo, el mejor enfoque hasta ahora para tratar el problema del desequilibrio y el sesgo en la clase más frecuente ha sido el aprendizaje activo con muestreo de incertidumbre, que por su diseño tendería a añadir aquellos casos sobre los que el modelo tenía menos información, es decir, las clases menos frecuentes.

El dominio del corpus rotulado también tuvo un gran impacto en los experimentos y resultados que analicé a través de esta tesis. El corpus etiquetado fue el recurso base para trabajar en esta tesis y como tal la calidad del recurso impactó directamente en la calidad de los modelos. Es importante notar que como era un recurso fuertemente basado en dominios, también podría oscurecer algunos de los resultados, por ejemplo, cuando mostraba un mejor desempeño para las características hechas a mano que los embeddings de palabras sólo porque estas últimas generalizaban mejor. Esto pone en consideración cómo interpreto los resultados: un modelo que muestra una caída en el rendimiento en comparación con otro modelo no es automáticamente un mal modelo. A veces hay otras métricas y otras vistas de los datos que ayudan a identificar lo que un modelo está omitiendo para ganar rendimiento, por ejemplo, la representatividad de las clases minoritarias o una mejor generalización a los datos de otros dominios. Además, la fuerte influencia en el dominio mostró la importancia de contar con datos más heterogéneos para buscar un buen modelo. De hecho, las técnicas de aprendizaje semi-supervisadas, que adquirían información de un corpus

de un dominio general, mejoraron algunos aspectos del modelo original, que estaba demasiado cerca del conjunto de datos etiquetado.

Los dos desafíos principales que se abordaron en este trabajo fueron la cobertura, observada como la información accionable por el modelo, así como la tendencia a sobredimensionarse o generalizarse. Estos desafíos fueron el resultado directo de tener una pequeña cantidad de datos etiquetados y de que dichos datos formaran parte de un dominio muy específico. Se exploraron técnicas semi-supervisadas para superar estos dos desafíos. Se deja para el trabajo futuro llevar a cabo experimentos que son más impulsados por el dominio y ver cómo el dominio realmente afecta los resultados finales cuando se exploran otros dominios. Por ejemplo, es interesante explorar cómo el uso de datos no etiquetados del mismo dominio o de un dominio contrastado afecta a los modelos semisupervisados.

Con más recursos disponibles, algunos de los trabajos futuros más importantes que tengo son el uso de corpus más grandes sin etiqueta con diferentes técnicas semi-supervisadas. Esto va desde la formación de embeddings de palabras con más corpus procedentes de diferentes recursos disponibles en línea, hasta el uso de más instancias de formación no etiquetadas en la red de escaleras. Para el aprendizaje activo, la anotación manual fue hecha mayormente por mí con la ayuda de mi supervisor de tesis, pero beneficiaría a un experto en el dominio que también podría trabajar con más ejemplos que la muy pequeña cantidad que podría llegar a anotar manualmente. Por último, me gustaría hacer el análisis de más de los lemas disponibles, ya que debido a las limitaciones de tiempo sólo pude explorar una pequeña muestra de todas las posibilidades disponibles que el área de desambiguación de sentidos verbales del español tiene para ofrecer. Con más recursos también, trabajaré en un análisis de errores más exhaustivo de los experimentos, especialmente los relacionados con el sobreequipamiento y la distribución de las clases. Estoy muy interesado en ver cómo los modelos semi-supervisados tratan con la anotación de nuevos ejemplos haciendo una evaluación manual de los mismos, especialmente para los casos de autoaprendizaje y redes de escaleras.

Por último, desde un punto de vista más técnico, en el futuro habrá que explorar diferentes combinaciones de las técnicas semisupervisadas, como el uso de un algoritmo combinado entre el autoaprendizaje y el aprendizaje activo o, además, la combinación de uno de los algoritmos de la envoltura con la propia red de escaleras. En particular, la red de escaleras es un enfoque muy prometedor que requeriría una mayor exploración para afinar sus componentes. Una línea de trabajo futuro sería utilizar la diferencia entre la distribución del corpus original y la distribución del lote previsto como criterio de parada. Además, muchos de los hiperparámetros pueden modificarse intentando conseguir un mejor rendimiento del algoritmo en la tarea: desde las ponderaciones de costes de denotar hasta la arquitectura de la capa de codificador, cambiando también la función de combinador dada por los autores. Aún así, la línea de trabajo más interesante en redes de escaleras sería comprobar cómo funcionan integrando una red neuronal más compleja que un simple perceptrón multicapa. Sería interesante ver

cómo una red de escalera construida sobre una red neuronal convolucional o una red neuronal recurrente puede ayudar al rendimiento final de los modelos. En términos generales, las redes de escaleras, siendo el enfoque más novedoso estudiado en esta tesis, ofrecen un amplio margen de mejora y exploración.

Bibliografía

[Agirre et al., 2014] Agirre, E., López de Lacalle, O., and Soroa, A. (2014). Random walks for knowledge-based word sense disambiguation. *Comput. Linguist.*, 40(1):57–84.

[Agirre and Soroa, 2009] Agirre, E. and Soroa, A. (2009). Personalizing pagerank for word sense disambiguation. In *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics*, EACL '09, pages 33–41, Stroudsburg, PA, USA. Association for Computational Linguistics.

[Alonso et al., 2007] Alonso, L., Capilla, J. A., Castellón, I., Fernández, A., and Vázquez, G. (2007). The SenSem project: Syntactico-Semantic Annotation of Sentences in Spanish. In et al, N. N., editor, *Selected Papers from RANLP 2005*, pages 89–98. John Benjamins.

[Bengio, 2008] Bengio, Y. (2008). Neural net language models. 3(1):3881. revision #91566.

[Bengio et al., 2003] Bengio, Y., Ducharme, R., Vincent, P., and Janvin, C. (2003). A neural probabilistic language model. *J. Mach. Learn. Res.*, 3:1137–1155.

[Bengio et al., 1994] Bengio, Y., Simard, P., and Frasconi, P. (1994). Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, 5(2):157–166.

[Berger et al., 1996] Berger, A. L., Pietra, V. J. D., and Pietra, S. A. D. (1996). A Maximum Entropy Approach to Natural Language Processing. *Comput. Linguist.*

[Bloodgood and Vijay-Shanker, 2009] Bloodgood, M. and Vijay-Shanker, K. (2009). Taking into account the differences between actively and passively acquired data: The case of active learning with support vector machines for imbalanced datasets. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics, Companion Volume: Short Papers*, pages 137–140. Association for Computational Linguistics.

[Bottou, 1998] Bottou, L. (1998). Online algorithms and stochastic approximations. In Saad, D., editor, *Online Learning and Neural Networks*. Cambridge University Press, Cambridge, UK. revised, oct 2012.

[Brown et al., 1992] Brown, P. F., deSouza, P. V., Mercer, R. L., Pietra, V. J. D., and Lai, J. C. (1992). Class-based n-gram models of natural language. *Comput. Linguist.*, 18(4):467–479.

[Cardellino, 2016] Cardellino, C. (2016). Spanish Billion Words Corpus and Embeddings.

[Cardellino and Alonso i Alemany, 2017] Cardellino, C. and Alonso i Alemany, L. (2017). Disjoint semi-supervised spanish verb sense disambiguation using word embeddings. In *ASAI - 46 JAIIO*.

[Cardellino et al., 2015] Cardellino, C., Teruel, M., and Alonso Alemany, L. (2015). Combining semi-supervised and active learning to recognize minority senses in a new corpus. In *Workshop on Replicability and Reproducibility in Natural Language Processing: Adaptive Methods, Resources and Software at IJCAI*.

[Chapelle et al., 2010] Chapelle, O., Schlkopf, B., and Zien, A. (2010). *Semi-Supervised Learning*. The MIT Press, 1st edition.

[Chen and Palmer, 2009] Chen, J. and Palmer, M. S. (2009). Improving English verb sense disambiguation performance with linguistically motivated features and clear sense distinction boundaries. *Language Resources and Evaluation*.

[Cohn and Cohn,] Cohn, T. and Cohn, T. Performance metrics for word sense disambiguation.

[Collobert and Weston, 2008] Collobert, R. and Weston, J. (2008). A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th International Conference on Machine Learning*, ICML '08, pages 160–167, New York, NY, USA. ACM.

[Cortes and Vapnik, 1995] Cortes, C. and Vapnik, V. (1995). Support-vector networks. *Machine Learning*, 20(3):273–297.

[Culotta and McCallum, 2005] Culotta, A. and McCallum, A. (2005). Reducing Labeling Effort for Structured Prediction Tasks. In *Proceedings of the 20th National Conference on Artificial Intelligence - Volume 2*, AAAI'05, pages 746–751, Pittsburgh, Pennsylvania. AAAI Press.

[Culp and Michailidis, 2008] Culp, M. and Michailidis, G. (2008). An iterative algorithm for extending learners to a semi-supervised setting. *Journal of Computational and Graphical Statistics*, 17(3):545–571.

[Dligach and Palmer, 2011] Dligach, D. and Palmer, M. (2011). Good seed makes a good crop: Accelerating active learning using language modeling. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*.

[Druck et al., 2009] Druck, G., Settles, B., and McCallum, A. (2009). Active Learning by Labeling Features. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 81–90. ACL.

[Ertekin et al., 2007] Ertekin, S. E., Huang, J., Bottou, L., and Giles, C. L. (2007). Learning on the border: Active learning in imbalanced data classification. In *In Proc. ACM Conf. on Information and Knowledge Management (CIKM '07.*

[Fisher, 1921] Fisher, R. (1921). On the probable error of a coefficient of correlation deduced from a small sample. *Metron*, 1:3–32.

[Fleiss et al., 1971] Fleiss, J. et al. (1971). Measuring nominal scale agreement among many raters. *Psychological Bulletin*, 76(5):378–382.

[Gella et al., 2016] Gella, S., Lapata, M., and Keller, F. (2016). Unsupervised visual sense disambiguation for verbs using multimodal embeddings.

[Goodfellow et al., 2016] Goodfellow, I., Bengio, Y., and Courville, A. (2016). Deep Learning. Book in preparation for MIT Press.

[Gutmann and Hyvärinen, 2012] Gutmann, M. U. and Hyvärinen, A. (2012). Noise-contrastive estimation of unnormalized statistical models, with applications to natural image statistics. *J. Mach. Learn. Res.*, 13:307–361.

[He and Carbonell, 2007] He, J. and Carbonell, J. G. (2007). Nearest-Neighbor-Based Active Learning for Rare Category Detection. In Platt, J. C., Koller, D., Singer, Y., and Roweis, S. T., editors, *NIPS*. Curran Associates, Inc.

[Hinton and Salakhutdinov, 2006] Hinton, G. E. and Salakhutdinov, R. R. (2006). Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507.

[Iacobacci et al., 2016] Iacobacci, I., Pilehvar, M. T., and Navigli, R. (2016). Embeddings for word sense disambiguation: An evaluation study. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 897–907, Berlin, Germany. Association for Computational Linguistics.

[Ide and Véronis, 1998] Ide, N. and Véronis, J. (1998). Introduction to the Special Issue on Word Sense Disambiguation: The State of the Art. *Comput. Linguist.*, 24(1):2–40.

[Ioffe and Szegedy, 2015] Ioffe, S. and Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift.

[Johnson, 2009] Johnson, M. (2009). How the statistical revolution changes (computational) linguistics. In *Proceedings of the EACL 2009 Workshop on the Interaction Between Linguistics and Computational Linguistics: Virtuous, Vicious or Vacuous?*,

ILCL '09, pages 3–11, Stroudsburg, PA, USA. Association for Computational Linguistics.

[Kawahara and Palmer, 2014] Kawahara, D. and Palmer, M. (May 26-312014). Single Classifier Approach for Verb Sense Disambiguation based on Generalized Features. In Chair), N. C. C., Choukri, K., Declerck, T., Loftsson, H., Maegaard, B., Mariani, J., Moreno, A., Odijk, J., and Piperidis, S., editors, *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, Reykjavik, Iceland. European Language Resources Association (ELRA).

[Krippendorff, 1970] Krippendorff, K. (1970). Estimating the reliability, systematic error and random error of interval data. *Educational and Psychological Measurement*, 30(1):61–70.

[Lewis and Catlett, 1994] Lewis, D. D. and Catlett, J. (1994). Heterogeneous Uncertainty Sampling for Supervised Learning. In *In Proceedings of the Eleventh International Conference on Machine Learning*, pages 148–156. Morgan Kaufmann.

[Maeireizo et al., 2004] Maeireizo, B., Litman, D., and Hwa, R. (2004). Co-training for predicting emotions with spoken dialogue data. In *Proceedings of the ACL 2004 on Interactive Poster and Demonstration Sessions*, ACLdemo '04, Stroudsburg, PA, USA. Association for Computational Linguistics.

[Manning et al., 2008] Manning, C. D., Raghavan, P., and Schütze, H. (2008). *Introduction to Information Retrieval*. Cambridge University Press, New York, NY, USA.

[Manning et al., 2014] Manning, C. D., Surdeanu, M., Bauer, J., Finkel, J., Bethard, S. J., and McClosky, D. (2014). The Stanford CoreNLP Natural Language Processing Toolkit. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 55–60.

[Màrquez et al., 2007] Màrquez, L., Padró, L., Surdeanu, M., and Villarejo, L. (2007). UPC: Experiments with Joint Learning within SemEval Task 9. In *Proceedings of the 4th International Workshop on Semantic Evaluations (SemEval-2007)*.

[McCarthy and Carroll, 2003] McCarthy, D. and Carroll, J. (2003). Disambiguating Nouns, Verbs, and Adjectives Using Automatically Acquired Selectional Preferences. *Comput. Linguist.*, 29(4):639–654.

[Mihalcea, 2004] Mihalcea, R. (2004). Co-training and self-training for word sense disambiguation. In Ng, H. T. and Riloff, E., editors, *HLT-NAACL 2004 Workshop: Eighth Conference on Computational Natural Language Learning (CoNLL-2004)*, pages 33–40, Boston, Massachusetts, USA. Association for Computational Linguistics.

[Mihalcea et al., 2004] Mihalcea, R., Chklovski, T., and Kilgarriff, A. (2004). The senseval-3 english lexical sample task. In *Proceedings of Senseval-3: Third International Workshop on the Evaluation of Systems for the Semantic Analysis of Text, Barcelona, Spain, July 2004*. Association for Computational Linguistics (ACL).

[Mikolov et al., 2013a] Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013a). Efficient Estimation of Word Representations in Vector Space.

[Mikolov et al., 2013b] Mikolov, T., Sutskever, I., Chen, K., Corrado, G., and Dean, J. (2013b). Distributed Representations of Words and Phrases and their Compositionality.

[Mitchell, 1997] Mitchell, T. (1997). *Machine Learning*. McGraw-Hill international editions - computer science series. McGraw-Hill Education. LCCB: 97007692.

[Mnih and Hinton, 2009] Mnih, A. and Hinton, G. E. (2009). A scalable hierarchical distributed language model. In Koller, D., Schuurmans, D., Bengio, Y., and Bottou, L., editors, *Advances in Neural Information Processing Systems 21*, pages 1081–1088. Curran Associates, Inc.

[Montoyo et al., 2011] Montoyo, A., Palomar, M., Rigau, G., and Suárez, A. (2011). Combining Knowledge- and Corpus-based Word-Sense-Disambiguation Methods. *CoRR*.

[Montraveta et al., 2008] Montraveta, A. F., Vázquez, G., and Fellbaum, C. (2008). The spanish version of wordnet 3.0. In Storrer, A., Geyken, A., Siebert, A., and Würzner, K., editors, *Text Resources and Lexical Knowledge. Selected Papers from the 9th Conference on Natural Language Processing, KONVENS 2008, Berlin, Germany*, pages 175–182. Mouton de Gruyter.

[Morin and Bengio, 2005] Morin, F. and Bengio, Y. (2005). Hierarchical probabilistic neural network language model. In Cowell, R. G. and Ghahramani, Z., editors, *Proceedings of the Tenth International Workshop on Artificial Intelligence and Statistics*, pages 246–252. Society for Artificial Intelligence and Statistics.

[Padró and Stanilovsky, 2012] Padró, L. and Stanilovsky, E. (2012). FreeLing 3.0: Towards Wider Multilinguality. In *Proceedings of the Language Resources and Evaluation Conference*.

[Pradhan et al., 2007] Pradhan, S. S., Loper, E., Dligach, D., and Palmer, M. (2007). Semeval-2007 task 17: English lexical sample, srl and all words. In *Proceedings of the 4th International Workshop on Semantic Evaluations*, SemEval '07, pages 87–92, Stroudsburg, PA, USA. Association for Computational Linguistics.

[Quinlan, 1986] Quinlan, J. R. (1986). Induction of decision trees. *Machine learning*, 1(1):81–106.

[Rasmus et al., 2014] Rasmus, A., Raiko, T., and Valpola, H. (2014). Denoising autoencoder with modulated lateral connections learns invariant representations of natural images.

[Rasmus et al., 2015a] Rasmus, A., Valpola, H., Honkala, M., Berglund, M., and Raiko, T. (2015a). Semi-Supervised Learning with Ladder Networks.

[Rasmus et al., 2015b] Rasmus, A., Valpola, H., and Raiko, T. (2015b). Lateral connections in denoising autoencoders support supervised learning.

[Řehůřek and Sojka, 2010] Řehůřek, R. and Sojka, P. (2010). Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 45–50, Valletta, Malta. ELRA. `http://is.muni.cz/publication/884893/en`.

[Rijsbergen, 1979] Rijsbergen, C. J. V. (1979). *Information Retrieval*. Butterworth-Heinemann, Newton, MA, USA, 2nd edition.

[Riloff et al., 2003] Riloff, E., Wiebe, J., and Wilson, T. (2003). Learning subjective nouns using extraction pattern bootstrapping. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003 - Volume 4*, CONLL '03, pages 25–32, Stroudsburg, PA, USA. Association for Computational Linguistics.

[Rosenberg et al., 2005] Rosenberg, C., Hebert, M., and Schneiderman, H. (2005). Semi-supervised self-training of object detection models. In *Application of Computer Vision, 2005. WACV/MOTIONS '05 Volume 1. Seventh IEEE Workshops on*, volume 1, pages 29–36.

[Rosenblatt, 1962] Rosenblatt, F. (1962). *Principles of neurodynamics: perceptrons and the theory of brain mechanisms*. Report (Cornell Aeronautical Laboratory). Spartan Books.

[Rothe and Schütze, 2015] Rothe, S. and Schütze, H. (2015). Autoextend: Extending word embeddings to embeddings for synsets and lexemes. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1793–1803, Beijing, China. Association for Computational Linguistics.

[Russell and Norvig, 2003] Russell, S. J. and Norvig, P. (2003). *Artificial Intelligence: A Modern Approach*. Pearson Education, 2 edition.

[Scudder, 1965] Scudder, H. (1965). Probability of error of some adaptive pattern-recognition machines. *IEEE Transactions on Information Theory*, 11(3):363–371.

[Settles, 2009] Settles, B. (2009). Active Learning Literature Survey. Computer sciences technical report, University of Wisconsin–Madison.

[Settles and Craven, 2008] Settles, B. and Craven, M. (2008). An Analysis of Active Learning Strategies for Sequence Labeling Tasks. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1069–1078. ACL.

[Sudarikov et al., 2016] Sudarikov, R., Dušek, O., Holub, M., Bojar, O., and Kríž, V. (2016). Verb sense disambiguation in machine translation.

[Suddarth and Kergosien, 1990] Suddarth, S. C. and Kergosien, Y. L. (1990). *Rule-injection hints as a means of improving network performance and learning time*, pages 120–129. Springer Berlin Heidelberg, Berlin, Heidelberg.

[Taghipour and Ng, 2015] Taghipour, K. and Ng, H. T. (2015). Semi-Supervised Word Sense Disambiguation Using Word Embeddings in General and Specific Domains. In *NAACL*.

[Tomanek and Hahn, 2009] Tomanek, K. and Hahn, U. (2009). Reducing Class Imbalance During Active Learning for Named Entity Annotation. In *Proceedings of the Fifth International Conference on Knowledge Capture*, K-CAP '09, pages 105–112, New York, NY, USA. ACM.

[Tong and Koller, 2002] Tong, S. and Koller, D. (2002). Support Vector Machine Active Learning with Applications to Text Classification. *J. Mach. Learn. Res.*, 2:45–66.

[Turian et al., 2010] Turian, J., Ratinov, L., and Bengio, Y. (2010). Word representations: A simple and general method for semi-supervised learning. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, ACL '10, pages 384–394, Stroudsburg, PA, USA. Association for Computational Linguistics.

[Valpola, 2014] Valpola, H. (2014). From neural pca to deep unsupervised learning.

[Vincent et al., 2010] Vincent, P., Larochelle, H., Lajoie, I., Bengio, Y., and Manzagol, P.-A. (2010). Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *J. Mach. Learn. Res.*, 11:3371–3408.

[Walker and Duncan, 1967] Walker, S. H. and Duncan, D. B. (1967). Estimation of the probability of an event as a function of several independent variables. *Biometrika*, 54(1-2):167–179.

[Weinberger et al., 2009] Weinberger, K., Dasgupta, A., Langford, J., Smola, A., and Attenberg, J. (2009). Feature hashing for large scale multitask learning. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, pages 1113–1120, New York, NY, USA. ACM.

[Weld et al., 2009] Weld, D. S., Hoffmann, R., and Wu, F. (2009). Using wikipedia to bootstrap open information extraction. *SIGMOD Rec.*, 37(4):62–68.

[Weston et al., 2008] Weston, J., Ratle, F., and Collobert, R. (2008). Deep Learning via Semi-supervised Embedding. In *Proceedings of the 25th International Conference on Machine Learning*, ICML '08, pages 1168–1175, New York, NY, USA. ACM.

[Wilks and Stevenson, 1996] Wilks, Y. and Stevenson, M. (1996). The Grammar of Sense: Is word-sense tagging much more than part-of-speech tagging? *CoRR*, cmp-lg/9607028.

[Yarowsky, 1995] Yarowsky, D. (1995). Unsupervised Word Sense Disambiguation Rivaling Supervised Methods. In *ACL-95*, pages 189–196, Cambridge, MA. ACL.

[Ye and Baldwin, 2006] Ye, P. and Baldwin, T. (2006). Verb Sense Disambiguation Using Selectional Preferences Extracted with a State-of-the-art Semantic Role Labeler. In *Proceedings of the Australasian Language Technology Workshop 2006*, pages 139–148, Sydney, Australia.

[Yuan et al., 2016] Yuan, D., Richardson, J., Doherty, R., Evans, C., and Altendorf, E. (2016). Semi-supervised word sense disambiguation with neural models.

[Zhao et al., 2017] Zhao, J., Wang, T., Yatskar, M., Ordonez, V., and Chang, K.-W. (2017). Men also like shopping: Reducing gender bias amplification using corpus-level constraints.

[Zhong and Ng, 2010] Zhong, Z. and Ng, H. T. (2010). It makes sense: A wide-coverage word sense disambiguation system for free text. In *Proceedings of the ACL 2010 System Demonstrations*, ACLDemos '10, pages 78–83, Stroudsburg, PA, USA. Association for Computational Linguistics.

[Zhu and Hovy, 2007] Zhu, J. and Hovy, E. H. (2007). Active Learning for Word Sense Disambiguation with Methods for Addressing the Class Imbalance Problem. *EMNLP-CoNLL*, 7:783–790.

[Zhu, 2005] Zhu, X. (2005). Semi-supervised learning literature survey. Technical Report 1530, Computer Sciences, University of Wisconsin-Madison.

[Zhu and Goldberg, 2009] Zhu, X. and Goldberg, A. B. (2009). Introduction to semi-supervised learning. *Synthesis lectures on artificial intelligence and machine learning*, 3(1):1–130.

[Zipf, 1949] Zipf, G. (1949). *Human Behavior and the Principle of Least Effort.* Addison–Wesley, Cambridge, MA.