Facultad de Matemática, Astronomía, Física y Computación

Universidad Nacional de Córdoba

Minería de Argumentos con Aprendizaje Profundo y Atención

Trabajo especial para la carrera de Licenciatura en Ciencias de la Computación

David Ignacio González

Directora: Milagro Teruel

Córdoba, Argentina 2019

Agradecimientos

Quiero agradecer principalmente a mis padres y hermanas por haberme apoyado incondicionalmente durante todos estos años, acompañándome en cada oportunidad.

A mis familiares y amigos de toda la vida, quienes siempre me brindaron fuerza para seguir adelante a lo largo de esta etapa.

A mi directora, Mili, por haberme guiado en este proceso con gran dedicación, alentándome en todo momento.

A mis compañeros de la facultad con los que he compartido todo este camino.

Resumen

En este trabajo agregamos un mecanismo de atención a una red neuronal del estado del arte, que consiste de una red BiLSTM con embeddings de caracteres y una capa de CRF. Este modelo no sólo ha sido previamente aplicado en minería de argumentos, sino en muchas otras tareas de etiquetado de secuencias, como el Reconocimiento de Entidades Nombradas, PoS tagging, chunking, y el reconocimiento de eventos. Se simplificó la red para poder lograr una mejor comparación de resultados, y agregamos dos enfoques distintos del mecanismo de atención, variando las funciones de activación disponibles. Luego se realizaron experimentos y se analizaron los resultados obtenidos para determinar el impacto de la atención en el rendimiento de la red. Los modelos con atención fueron evaluados con un conocido corpus de ensayos persuasivos. El mismo consta de 402 ensayos de estudiantes en inglés, anotados manualmente con componentes argumentativos y sus relaciones. Se observó que el rendimiento de uno de los enfoques del mecanismo de atención superó al modelo sin atención, mientras que el enfoque restante no obtuvo mejoras en el desempeño. De la misma manera se comprobó que las distintas funciones de activación del mecanismo de atención son determinantes para el mismo.

Abstract

In this paper, we add an attention mechanism to a state-of-the-art neural network, which consists of a BiLSTM with characters embeddings and a CRF layer. This model has not only been previously applied in argument mining, but in many other sequence tagging tasks, such as Named Entity Recognition, PoS tagging, chunking, and event recognition. The network was simplified in order to achieve a better comparison of results, and then two different attention mechanism approaches were added, varying between the availables activation functions. Experiments were then conducted, and the results obtained were analyzed to determine the impact of attention on the network performance. Attention models were evaluated with a well-known persuasive essays corpus. It consists of 402 students essays, manually annotated with argumentative components and their relations. It was observed that the performance of one of the attention mechanism approaches outperformed the no-attention model, while the remaining approach did not result in improvements of performance. Also, we proved that the different activation functions of the attention mechanism are decisive in the network performance.

Palabras claves: Modelo, Mecanismo, Minería, Argumento, Atención, Arquitectura, Red Neuronal, Clasificación, Puntaje.

Clasificación (ACM CCS 2012):

- Computing methodologies ∼ Artificial intelligence ∼ Natural language processing
- Computing methodologies ∼ Machine learning ∼ Machine learning approaches ∼ Neural networks

Índice general

Capítulo 1

Introducción y Motivación

La argumentación es una actividad verbal que apunta a aumentar o disminuir la aceptación de un punto de vista controversial, según van Eemeren et al. [1996, p. 5]. Es omnipresente en nuestra comunicación diaria. Los argumentos bien razonados no sólo son importantes para la toma de decisiones, sino que también juegan un papel crucial para obtener conclusiones con amplio consenso.

Desde un punto de vista computacional, nos interesan especialmente los argumentos estructurados. Sin embargo, no es posible dar una única definición formal y universalmente aceptada de argumento estructurado debido a que hay diferentes propuestas, suficientemente fundamentadas, que definen la argumentación estructurada [Besnard et al., 2014]. Walton [2009] da una definición intuitiva de argumento como un conjunto de declaraciones que consta de tres partes: un conjunto de premisas, una conclusión y una inferencia de las premisas a la conclusión. En la literatura, a veces se hace referencia a las conclusiones como afirmaciones, las premisas a menudo se denominan evidencia o razones (o dato, en el modelo de Toulmin [2003]), y el vínculo entre las dos, i.e., la inferencia, a veces se denomina como el argumento en sí.

Las expresiones de los argumentos suelen ser diversas y complejas, haciendo que su identificación automática en los textos sea una tarea muy desafiante. Mas allá de la complejidad del lenguaje, muchas de

las partes de un argumento no están marcadas claramente por formas lingüísticas específicas a nivel superficial. Por ello, muchas veces es necesario recurrir a la semántica para identificarlas, delimitarlas y entenderlas, y luego identificar las relaciones entre ellas.

Para poder identificar las estructuras argumentativas dentro del texto se requieren varias subtareas, tales como: (a) separar las unidades argumentativas de las unidades no argumentativas, llamado segmentación de componentes; (b) clasificar los componentes del argumento en clases como "Premisa" o "Afirmación"; (c) encontrar relaciones entre los componentes del argumento; (d) clasificar las relaciones en clases como "Apoyo" o "Ataque" [Persing and Ng, 2016], [Stab and Gurevych, 2014b].

Nuestro enfoque tiene como objetivo la clasificación de componentes argumentativos según su rol en el argumento. La estructura interna de un argumento consta de varios componentes argumentativos, pero como se menciona anteriormente, el tipo y las relaciones entre estos componentes se definen de distintas maneras de acuerdo al marco teórico utilizado.

Siguiendo a A Govier [2010] y la forma como se implementan estos conceptos en Stab and Gurevych [2014a], en este trabajo se considera que un componente argumentativo incluye al menos una afirmación y una o más premisas, tal como se ve en la Figura 1.1. La afirmación es una declaración polémica y el componente central de un argumento, mientras que las premisas son razones para justificar (o refutar) la afirmación. Además, los argumentos incluyen potencialmente relaciones argumentativas entre sus componentes. Cada una de estas relaciones indica que un componente justifica o refuta a otro. A su vez, distintos argumentos pueden combinarse entre sí con relaciones similares, dando origen a una argumentación compleja. La argumentación en su conjunto sostiene una posición principal del autor.

En este trabajo se abordará la tarea de clasificación de componentes argumentativos. Los componentes son:

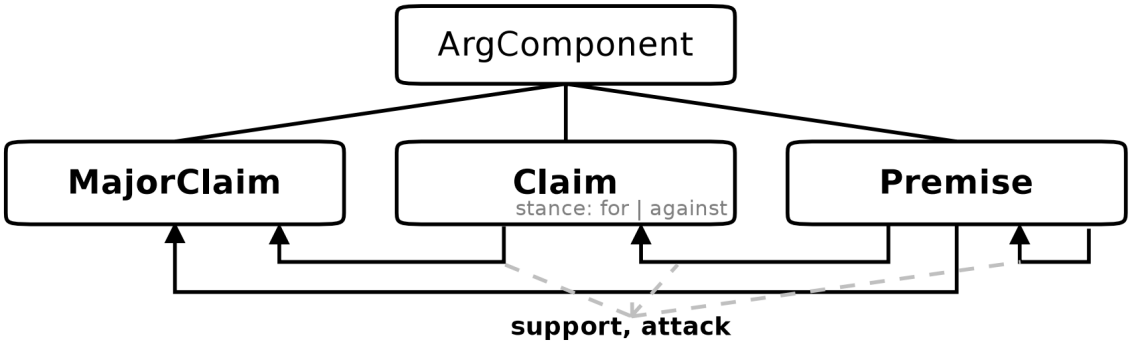


Figura 1.1: Componentes argumentativos

- Afirmación principal (*Major Claim*): Es la posición central de un autor respecto al tema del ensayo.

- Afirmación (*Claim*): Declaraciones controversiales que están en contra o a favor de la afirmación principal, que no deberían ser aceptadas por el lector sin una justificación adicional. Es apoyada o atacada por al menos una premisa.

- Premisa (*Premise*): Es una justificación dada por el autor para convencer a los lectores sobre su afirmación u otras premisas, apoyándolas o atacándolas.

Por ejemplo, el argumento del siguiente párrafo contiene cuatro componentes argumentativos: una afirmación (en negrita) y tres premisas (subrayadas).

> (1) **Los museos y las galerías de arte proveen un mejor entendimiento sobre el arte que Internet.** (2) En la mayoría de los museos y galerías de arte, se proveen descripciones detalladas sobre el background, la historia y el autor. (3) Ver una obra de arte online no es lo mismo que verlo con tus propios ojos, ya que (4) la imagen online no muestra la textura o la estructura tri-dimensional del trabajo, lo que es importante para estudiar

Otra tarea dentro la minería de argumentación es identificar las relaciones entre estos componentes. Esta tarea no será considerada en este trabajo, debido a que su nivel de complejidad es mayor.

1.1. ¿Qué es la minería de argumentos?

La minería de argumentos (también *minería de argumentación*) es un desafío relativamente nuevo en el análisis de texto basado en datos, una subárea del Procesamiento del Lenguaje Natural (PLN). Su objetivo es identificar e interpretar automáticamente estructuras argumentativas dentro del discurso, e identificar los componentes relevantes y sus relaciones en dicho argumento.

La argumentación se ha referido históricamente al proceso de construir argumentos y al proceso de determinar el conjunto de conclusiones justificadas de un conjunto de argumentos. Sin embargo, la minería de argumentos a menudo se usa indistintamente para estas u otras aplicaciones y en un sentido amplio, ya que el campo aún conserva un fuerte elemento de exploración conceptual.

Con la consolidación de las técnicas basadas en *datos* y las grandes cantidades de textos disponibles, cada vez es más factible recopilar argumentaciones de diversos entornos, las cuales generalmente vienen acompañadas de una postura del orador.

1.2. Posibles aplicaciones de la minería de argumentos

Hasta la fecha, se han investigado métodos para la minería de argumentos en áreas tales como toma de decisiones, documentos legales, debates en línea, revisiones de productos, literatura académica, comentarios de los usuarios sobre regulaciones propuestas, artículos de periódicos y casos judiciales, así como en dominios de diálogo, y la lectura, escritura y resumen de textos persuasivos desde un punto de

vista crítico.

La automatización del proceso de búsqueda de argumentos podría facilitar gran parte del esfuerzo manual involucrado en estas tareas, particularmente para manejar de manera robusta los argumentos de diferentes tipos de texto y temas. En este trabajo, exploraremos modelos capaces de encontrar y recuperar componentes argumentativos a partir de ensayos persuasivos producidos por estudiantes.

Una razón por la cual aplicar la minería de argumentos sobre ensayos persuasivos, se encuentra en el poder que tienen los argumentos para cambiar el parecer de las personas respecto a un tema en particular. Luego, detectar automáticamente argumentos que se inclinen hacia cierta posición respecto a un tema, permite obtener distintas razones fundamentadas por las que dicha posición es la correcta, sin hacer el esfuerzo de analizar manualmente dichos documentos.

Por otro lado, la argumentación también es un aspecto crucial en la adquisición de la habilidad en escritura. La habilidad de formular textos persuasivos no es sólo el fundamento de convencer una audiencia sobre ideas novedosas, sino que también juega un rol importante en la toma de decisiones en general y el análisis de diferentes posturas. Sin embargo, actualmente el apoyo en la escritura se limita a correcciones sobre la ortografía, gramática, o propiedades estilísticas, y no hay un sistema que proporcione comentarios sobre la argumentación escrita. Al integrar la minería de la argumentación en los entornos de escritura, los estudiantes podrán inspeccionar sus textos para mejorar la calidad de su argumentación.

1.3. La promesa del aprendizaje profundo

La minería de argumentaciones es una tarea desafiante debido a que el concepto de argumento depende del dominio que se quiera analizar. Un argumento en el dominio de documentos legales, difiere totalmente de lo que puede considerarse un argumento en el dominio de debates en línea, o de lo que puede considerarse un argumento dentro de un ensa-

yo. Por ejemplo, mientras que un ensayo contiene cinco o seis párrafos, la argumentación en sentencias de juicios usualmente se extiende a lo largo de varias páginas.

Además de la complejidad intrínseca de la argumentación y el número de componentes, distintos dominios del lenguaje natural utilizan vocabularios y estructuras propios. Por ejemplo, los marcadores de discurso que identifican los componentes argumentativos pueden diferir ampliamente. En consecuencia, los conjuntos de datos o *datasets* suelen utilizar marcos de anotación de la estructura de argumentos distintos, generalmente bastante complejos.

La tarea es tan compleja semánticamente y tan variada, que incluso los humanos tienen problemas para definirla formalmente. Cualquier propuesta debe ser lo suficientemente flexible como para tener en cuenta esta variabilidad y no estar limitada a un único dominio.

Al mismo tiempo, los propios corpus pueden llegar a diferir considerablemente respecto a los esquemas de argumentación utilizados para etiquetarlos, ya que no hay un consenso general sobre un esquema en particular. Inevitablemente, algunos esquemas representarán mucho mejor el fenómeno de argumentación, lo que lleva a que los modelos puedan realmente explotar las causas latentes expresadas en el texto. Por el contrario, si el esquema no es adecuado, podría ocultar las causas latentes al modelo.

A esto se le suma que los humanos suelen no coincidir a la hora de anotar un corpus bajo un esquema ya definido. Esto introduce discrepancias en las anotaciones de los documentos y dificulta el uso generalizado de éstos, llevando a los modelos a adaptarse más a ciertos conjuntos de datos que a otros. En otras palabras, el modelo se sobreajusta a un estilo particular de corpus-anotación.

Por otro lado, los métodos tradicionales utilizados se basan en la identificación manual de las características que indican un componente argumentativo. Por ejemplo, una posible característica sería que si el texto contiene la frase “ya que”, entonces la frase precedente es una afirmación (*Claim*), y la frase posterior es una premisa (*Premise*).

Sin embargo, se tiene la limitación de que estas características pueden variar completamente entre dominios, haciendo que la tarea dependa de éste. Además, si estas características se identifican manualmente, es probable que no se capturen todas las variaciones posibles de la estructura argumentativa, haciendo que la cobertura de las características no sea completa y pueda llegar a variar demasiado entre distintos corpus. En el ejemplo anterior podemos ver que, si cambiamos la frase "ya que" por la palabra "porque", se mantiene la estructura argumentativa mencionada. Si esta característica no fue identificada, entonces la frase no va a ser marcada como un argumento, evidenciando la falta de cobertura mencionada.

Es por esto que la identificación manual de las características no escala en nuestro problema, especialmente cuando tenemos en cuenta argumentaciones de distintos dominios. Adicionalmente, la ingeniería de características es un proceso costoso, tanto en tiempo y dinero, como en el nivel de conocimiento necesario del dominio que ésta implica.

En contraste con estos métodos, los modelos neuronales aprenden por sí mismos las características y cómo éstas se relacionan, a través de capas ocultas que logran capturar la información subyacente automáticamente. Estas representaciones jerárquicas tienen la capacidad de representar fenómenos muy complejos, y han alcanzado el estado del arte en otras tareas de PLN comparables, como por ejemplo etiquetado de secuencias [Ma and Hovy, 2016], análisis de dependencia [Andor et al., 2016], y traducción automática [Wu et al., 2016].

Por otro lado, arquitecturas tradicionales de minería de argumentos llamadas *pipelines*, solucionan independientemente cada tarea en distintas etapas. El resultado de cada etapa es la entrada de la etapa siguiente. Por ejemplo, primero se identifica si la oración contiene un componente argumentativo, luego se delimita y finalmente se le asigna una etiqueta (afirmación o premisa).

En contraste, las arquitecturas neuronales son capaces de aprender todas las tareas argumentativas juntas, aprovechando las relaciones entre las causas latentes. Esto implica un mejor aprendizaje para el

modelo, ya que las causas latentes que afectan a una de las tareas, también determinan las otras tareas. Sin embargo, al analizar las tareas por separado se pueden perder estas causas latentes, sin lograr que el modelo aprenda la tarea argumentativa completa.

Claramente otra desventaja sobre las arquitecturas de pipelines, es que conducen a la propagación de errores en las tareas anteriores, en lugar de explotar las interrelaciones entre variables.

En resumen, los métodos de aprendizaje profundo automatizan la extracción de características mientras que al mismo tiempo han mostrado ser capaces de tratar problemas complejos, como los datos esparzos, de forma eficiente y con muy buen rendimiento.

1.4. La promesa del mecanismo de atención

Debido a la complejidad de la tarea, los enfoques basados en datos requieren una gran cantidad de ejemplos para caracterizar adecuadamente los fenómenos, y encontrar patrones que pueden ser explotados por un analizador automático. Sin embargo, sólo hay pequeños corpus anotados disponibles para esta tarea, y no pueden usarse en combinación porque se basan en diferentes marcos teóricos (por ejemplo, argumentación abstracta, argumentación estructurada) o cubren diferentes géneros (por ejemplo, debates políticos, publicaciones en redes sociales, ensayos persuasivos). Luego necesitamos aprovechar la mayor información posible, por lo que es importante saber qué parte de ella es verdaderamente relevante, y separarla del "ruido".

En los últimos años, los modelos basados en atención han demostrado un gran éxito en muchas tareas de PLN, como la traducción automática [Bahdanau et al., 2014], [Sutskever et al., 2014], respuesta a preguntas [Sukhbaatar et al., 2015], reconocimiento de la vinculación textual [Rocktäschel et al., 2015] y minería de argumentos [Stab et al., 2018b].

Al momento de querer clasificar oraciones, nos encontramos con que hay palabras dentro de las mismas que podrían ser determinantes pa-

ra ello. Debido a esto, proponemos utilizar un *mecanismo de atención* que focalice al clasificador en tales palabras y que "ignore" aquellas que no son relevantes para la tarea, para así llevar su entrenamiento a una convergencia más rápida con resultados más acertados. El comportamiento esperado por este mecanismo sería que frases como "Es por esto que", "Pienso que", etc., obtengan mayor peso y que así el clasificador logre una mejor interpretación de las oraciones a lo largo del entrenamiento. Por lo tanto, se eliminaría información residual de características menos determinantes, lo cual disminuye posibles ruidos o información aleatoria de los ejemplos de entrenamiento, que no contribuyen a generalizar adecuadamente a otros ejemplos.

Por otro lado, este mecanismo de atención ayudaría a hacer un mejor análisis de los modelos neuronales entrenados, ya que nos permite ver los distintos puntajes de atención que reciben las palabras en las oraciones. Al poder reconocer qué partes de la entrada reciben más atención, es posible, por ejemplo, mejorar los procesos de anotación y así proporcionar mejores ejemplos de entrenamiento.

En esta sección, propusimos utilizar un mecanismo de atención que mejorará el rendimiento del modelo en comparación con los enfoques sin atención. Además, ayudará a identificar palabras que son importantes para la tarea de minería de argumentos. Esperamos poder usar los puntajes de atención para llevar a cabo un análisis de los errores con discrepancias entre los humanos y los clasificadores automáticos. De esta manera, se pueden refinar los esquemas de anotación en base a patrones extraídos de los datos. Esto ayudaría a usar un esquema argumentativo más adecuado y a eliminar el ruido generado por discrepancias entre humanos. En una exploración preliminar se vio que los mecanismos de atención identifican palabras fuertemente asociadas con tipos específicos de componentes de argumentos. En la figura 1.2 podemos observar que el mecanismo de atención destaca palabras que están fuertemente asociadas con componentes de tipo "afirmación", como "*admissible*" o "*inadmissible*".

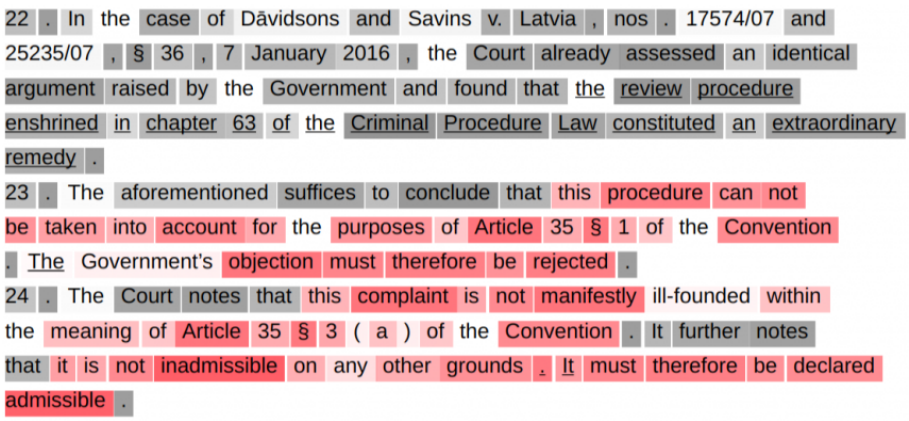


Figura 1.2: Ejemplo de un texto donde el mecanismo de atención da diferentes intensidades a las palabras según la fuerza de asociación de cada una con un tipo de componente argumentativo. El rojo identifica asociación con "afirmación", el gris con "premisa".

1.5. Objetivos

El objetivo de este trabajo es evaluar el impacto del mecanismo de atención en la tarea de minería de argumentaciones, y analizar cómo varía dicho impacto en base a las distintas funciones de activación utilizadas.

Para ello se simplificará una red neuronal del estado del arte y se le agregará un mecanismo de atención con dos enfoques distintos. Luego se entrenará dicha red con un conjunto de ensayos de estudiantes. Se analizará el comportamiento de los dos enfoques de atención sobre la red, y cómo las distintas funciones de activación afectan su rendimiento. Luego, a través de los resultados obtenidos por los distintos modelos, podremos concluir si el mecanismo de atención ayuda a la red a obtener un mejor rendimiento o no.

1.6. Contribuciones

Como contribución, en este trabajo se pudo consolidar la implementación del mecanismo de atención sobre una arquitectura del estado

del arte en tareas de clasificación de secuencias. Se proponen dos enfoques distintos para aplicar dicho mecanismo: atención basada en la representación de cada palabra y atención basada en el contexto. Para cada uno de ellos, se utilizan distintas funciones de activación que aportan distintas interpretaciones al valor de los puntajes de atención.

A través de una serie de experimentos, se evaluó el impacto de ambos mecanismos de atención en una tarea de minería de argumentaciones, y se comparó su rendimiento con el de la arquitectura sin atención. Esta evaluación involucró un proceso de búsqueda y selección de hiperparámetros aleatorio, que permitió realizar un análisis de cómo afectan el rendimiento de la red.

Finalmente, concluimos que los resultados obtenidos por el mecanismo de atención basado en las palabras fueron buenos, ya que logró ayudar a la red a generalizar mejor ante la escasez de datos, enfocándola en las palabras importantes. Consecuentemente, el mecanismo de atención podría aplicarse para lograr ayudar al anotador al momento de etiquetar argumentos, llevando a una disminución de bias en el proceso de anotación de los datasets.

1.7. Estructura de la tesis

Este trabajo está estructurado de la siguiente manera:

En el Capítulo 2 se analizarán distintos trabajos de investigación, en los cuales se exploran arquitecturas y posibles soluciones para el problema de la minería de argumentación, haciendo foco en las principales diferencias respecto a nuestro trabajo.

En el Capítulo 3 se explicará de forma detallada el mecanismo de atención y cómo se compone la arquitectura de nuestra red neuronal.

En el Capítulo 4 analizamos los resultados de distintos experimentos, planteados con el propósito de poder analizar el impacto del mecanismo de atención y de los distintos hiperparámetros de la red, contrastándolo con el desempeño de la red sin el mecanismo.

Finalmente, en el Capítulo 5 concluimos con un resumen del trabajo

realizado, un análisis de los resultados y su relevancia, y una propuesta de posibles trabajos futuros a partir del mismo.

Capítulo 2

Trabajo Relacionado

2.1. Minería de argumentos

Como dijimos en la sección 1.1, la minería de argumentos es un área naciente del procesamiento de lenguaje natural que busca el reconocimiento automático de argumentos en documentos. Automatizar el proceso de búsqueda de argumentos puede facilitar mucho el esfuerzo manual involucrado en tareas como toma de decisiones [Svenson, 1979], razonamiento legal [Wyner et al., 2010], y la lectura, escritura y resumen de textos persuasivos [Wingate, 2012]. La importancia de esta tarea también ha sido recalcada por trabajos recientes en asistencia para redacción [Zhang et al., 2016] y correcciones de ensayos [Persing and Ng, 2015].

La minería de argumentos comprende diversas tareas que pueden ser resueltas individualmente o en conjunto. Por un lado, es necesario identificar y clasificar los componentes argumentativos. Como se mencionó en el capítulo anterior, la definición de un componente argumentativo varía de un modelo argumentativo a otro. Además de ello, para describir la argumentación como un todo es necesario también encontrar las relaciones entre los distintos componentes argumentativos.

La mayoría de los trabajos se centran en subtareas de la minería de argumentaciones, y relativamente pocos trabajan sobre el problema

de una manera integral, identificando los componentes y también las relaciones entre ellos. Por ello, es importante tener en cuenta qué tarea se intenta resolver al analizar cada uno de los trabajos relacionados.

Por otra parte, estas subtareas pueden ser abordadas individualmente o en conjunto. Un tipo de arquitectura habitual es la de *pipeline*, donde se resuelve cada tarea antes de pasar a la próxima. Por ejemplo, un modelo primero identifica las oraciones argumentativas, un segundo modelo las clasifica y un tercero encuentra las relaciones entre ellas.

Sin embargo, también es posible resolver todas las tareas en simultáneo. Este tipo de arquitecturas es llamada *end-to-end*, ya que toma el texto original y devuelve, utilizando un mismo modelo, toda la estructura argumentativa.

Cabe destacar que, en este trabajo, sólo nos centramos en la identificación de componentes. Sin embargo, el método que utilizamos puede ser generalizado a cualquier tarea de clasificación, incluyendo la detección y clasificación de relaciones entre dichos componentes.

2.1.1. Sistemas no neuronales

Los primeros trabajos en minería de argumentaciones se centraban en el problema de detección de argumentos, siendo tratado inicialmente mediante métodos simbólicos. Un trabajo donde podemos observar esto es el de Kang and Saint-Dizier [2014], donde se desarrolla una gramática para analizar argumentos en texto. Esta gramática está fuertemente basada en claves léxicas superficiales (marcadores del discurso) y esto hace que su rendimiento, aunque tenga buena precisión, falle por baja cobertura.

Este tipo de método simbólico está limitado exclusivamente al conjunto de reglas con el que fue diseñado originalmente. Esto implica que nunca vamos a tener asegurada la completitud de los mismos respecto a su conjunto de reglas, ya que las mismas son introducidas manualmente, posiblemente omitiendo otros casos.

A esto se le suma que las claves léxicas son independientes del tamaño del argumento, por lo que no nos garantiza capturar cada argumento en su completitud, sino que puede dar lugar a que queden partes del mismo sin clasificar. Ante el evidente problema de escalabilidad de este método, es clara la necesidad de un método que no dependa de un conjunto de reglas fijas.

En contraste, otra forma de abordar problemas del procesamiento del lenguaje natural es el aprendizaje automático o *machine learning*. Con estos métodos se obtienen modelos que aprenden automáticamente a partir de la experiencia, sin ser programados explícitamente.

Este proceso de aprendizaje se basa en buscar patrones sobre los datos de ejemplo provistos, para tomar mejores decisiones en el futuro. El objetivo principal es permitir que el modelo aprenda sin intervención humana. Esto evita estar limitados a la poca cobertura que ofrecen las gramáticas o conjuntos de reglas hechos a mano, o sólo a los patrones reconocidos por un humano.

A continuación repasaremos algunos trabajos en los que se aplican distintas técnicas con aprendizaje automático.

En el trabajo de Ajjour et al. [2017], se estudian sistemáticamente los parámetros de la segmentación de unidades. Esta es una de las primeras tareas dentro de un pipeline para hacer minería de argumentos, y consiste en dividir un texto en segmentos argumentativos y no argumentativos. Para dicho estudio, se explora la efectividad de distintas representaciones o *features*, ya sea utilizando las palabras separadas, junto a sus vecinas, o el texto entero.

Otra técnica, aplicada en el trabajo de Al-Khatib et al. [2016], es la de supervisión distante. El objetivo es construir un corpus de gran tamaño con segmentos de texto sobre distintos dominios, anotados en base a su argumentación, i.e., si es argumentativo o no. Se utiliza supervisión distante ya que es bien conocida por entrenar clasificadores estadísticos robustos. La idea detrás de ella es generar anotaciones mediante el mapeo automático de datos sin etiquetar a un conjunto de etiquetas predefinidas. Para obtener el corpus, se recolectaron datos

de portales de debates online que contienen segmentos de texto argumentativos y no argumentativos sobre distintos temas. Estos están organizados de forma semi estructurada, permitiendo que se deriven las anotaciones a partir de ellos. Luego de obtener el corpus, entrenan un clasificador robusto para argumentos partiendo de features manuales.

En estos últimos dos enfoques, a diferencia de nuestra propuesta, la clasificación está limitada a sólo indicar si un texto es argumentativo o no, sin distinguir de qué tipo de argumento se trata.

Por otra parte, el trabajo de Nguyen and Litman [2016] se basa en el minado de relaciones argumentativas utilizando información contextual. Se extraen features a partir de la temática de los textos escritos, así como también de las oraciones que rodean a los distintos componentes argumentativos. El objetivo de esto es mejorar la predicción en las relaciones entre ellos. Se realizaron experimentos en ensayos de estudiantes, llegando a la conclusión de que las features contextuales ayudan significativamente al desempeño de dos tareas de clasificación de relaciones argumentativas: Support vs. Non-support y Support vs. Attack.

Otros ejemplos del uso de aprendizaje automático se encuentran en Persing and Ng [2015] y Stab and Gurevych [2017]. Estos trabajos utilizan arquitecturas de *pipeline*. Primero entrenan modelos independientemente para cada sub-tarea. Algunas de ellas pueden ser análisis de la estructura del discurso, identificación de componentes argumentativos, clasificación de componentes argumentativos, e identificación de relaciones entre componentes. Luego, la salida de cada una de estas sub-tareas es utilizada como entrada en la siguiente tarea. El modelo propuesto optimiza globalmente los tipos de los componentes argumentativos y las relaciones entre ellos utilizando programación lineal entera. Adicionalmente se pueden agregar restricciones globales como, por ejemplo, que cada premisa tiene un padre, etc.

Las arquitecturas de pipeline utilizadas por estos trabajos son problemáticas, ya que solucionan las tareas independientemente. Esto lle-

va a que se propaguen y acumulen errores, en vez de explotar las relaciones entre las causas latentes de cada tarea.

Aparte de esta arquitectura de pipeline, los modelos mencionados también tienen en común que dependen fuertemente de las features manuales (hand-crafted features). Estas tienen una limitación, ya que la minería de argumentos es un problema en el que la noción de "argumento" depende críticamente de la teoría de argumentación subyacente. Luego, si los features son diseñados a mano, es probable que no se capturen todas las variaciones posibles de la estructura argumentativa.

Es por esto que utilizar features manuales no escala en nuestro problema cuando tenemos en cuenta argumentaciones de distintos dominios. En contraste, las arquitecturas neuronales aprenden los features y cómo éstos se relacionan a través de las distintas capas que las componen, logrando capturar la información subyacente por sí misma. Luego, las arquitecturas neuronales suponen una mejor opción para la minería de argumentos en general.

2.1.2. Arquitecturas neuronales

Una alternativa a los modelos basados en hand-crafted features son las arquitecturas neuronales. Este tipo de modelo ha sido utilizado exitosamente en muchas tareas del procesamiento del lenguaje natural. Algunas de ellas son la clasificación de documentos en temas, la minería de entidades nombradas, y la repuesta automática a preguntas.

Las arquitecturas neuronales se caracterizan por una serie de capas internas que transforman la entrada sucesivamente hasta llegar a la última capa de clasificación. De esta manera, se construyen múltiples representaciones de los mismos datos, aplicando un función no lineal a la representación de la capa anterior. Con estos mecanismos se puede aprender una jerarquía de representaciones no trivialmente relacionadas, que potencialmente capturan fenómenos lingüísticos complejos como las argumentaciones.

Además del potencial de las redes neuronales en general, un tipo particular, llamado redes recurrentes, es capaz de procesar datos secuenciales (por ejemplo, el texto) y reconocer patrones entre elementos alejados en la secuencia. Una de las variantes más exitosas, la LSTM, tiene mecanismos para mantener y olvidar la información histórica relevante de la secuencia que ya ha sido procesada. Esto las hace especialmente útiles para analizar texto, ya que al "recordar" lo visto previamente, se puede obtener una cierta noción de contexto. Consecuentemente, logran capturar distintas propiedades semánticas que no podrían capturarse sin tal contexto.

Un trabajo basado en arquitecturas neuronales para minería de argumentaciones es el de Eger et al. [2017]. Esta propuesta ha sido la base a partir de la cual construimos el modelo presentado en este trabajo. Además de ampliar la arquitectura utilizada, evaluaremos los resultados sobre el mismo conjunto de datos de ensayos argumentativos que este trabajo, el cual se describirá en la sección de experimentos.

Los autores proponen que el análisis de dependencias puede considerarse una opción natural para la tarea de minería de argumentaciones (AM), porque las estructuras de discusión a menudo forman árboles. Sin embargo, concluyen que interpretar a la tarea de AM sólo como un análisis de dependencias conduce a resultados de rendimiento inferiores. Proponen entonces una arquitectura de etiquetado de secuencias que resuelve todas las sub-tareas de AM simultáneamente, utilizando un modelo neuronal estándar para múltiples entornos.

Los resultados obtenidos muestran que la arquitectura neuronal logra superar el rendimiento de un modelo basado en características manuales, eliminando la necesidad de hacer ingeniería sobre ellas. También se demuestra que la BiLSTM tiene muy buen desempeño para la clasificación de componentes.

Basados en esta hipótesis expuesta por Eger et al. [2017], en este trabajo también modelamos el problema como un etiquetado de secuencias. En los últimos años, otros problemas de etiquetado de secuencias han sido abordados exitosamente con arquitecturas neuro-

nales combinadas. Un ejemplo es el trabajo de Ma and Hovy [2016], donde implementan una arquitectura *end-to-end* que toma como input word embeddings pre-entrenados sobre corpus sin etiquetar, y utiliza una combinación de modelos BiLSTM, CNNs y CRF para tratar los problemas de POS tagging y NER.

El funcionamiento e interacción de los distintos componentes del sistema implementado en Eger et al. [2017], se describirán en detalle en el capítulo 3. A continuación, se dará una breve intuición sobre el funcionamiento del modelo.

La secuencia de la palabras que conforman el texto es pasada a una red neuronal convolucional (CNN), que codifica la información a nivel caracteres de las palabras. Estas redes son efectivas para generalizar las palabras fuera del vocabulario, ya que extraen información morfológica de los caracteres de las palabras y dan una representación neuronal de esta información.

Luego, se combinan estas representaciones basadas en caracteres con los embeddings de palabras. El resultado es introducido a la capa de BiLSTM (*Bidirectional LSTM*) para que modele la información contextual de cada palabra. La idea básica de usar una capa bidireccional, es unir la información contextual posterior y anterior a la palabra que se está analizando.

Finalmente, se usa un clasificador CRF (*conditional random field*) para decodificar las etiquetas de la oración completa, tomando como entrada el vector proveniente de la BiLSTM. Éste obtiene una familia de probabilidades condicionales sobre todas las posibles etiquetas, decodificando conjuntamente la mejor cadena de etiquetas para la secuencia dada. Así, se consideran las correlaciones entre etiquetas de las distintas secuencias, sanitizando posibles errores de la capa recurrente.

Este modelo también ha sido extensamente evaluado en Reimers and Gurevych [2017], donde fue aplicado sobre varias tareas de etiquetado de secuencias de distintos idiomas y dominios. Esta flexibilidad está dada ya que no depende de características manuales o del pre-procesamiento de los datos.

Reimers and Gurevych [2017] hacen un análisis extenso sobre el desempeño de clasificadores neuronales con conclusiones interesantes. Demuestran que, para tareas comunes en etiquetado de secuencias como POS tagging, Chunking, Named Entity Recognition, Entity Recognition, y Event Detection, el valor de la semilla para el generador de números aleatorios puede resultar en diferencias de desempeño estadísticamente significativas para sistemas que son de el estado del arte. Lo mismo sucede con la selección de los hiperparámetros. Como consecuencia, esto podría llevar a obtener conclusiones erróneas sobre los modelos. Luego, presentan arquitecturas que producen muy buenos resultados en cuanto a desempeño máximo, así como también arquitecturas que producen resultados más estables, dependiendo de los hiperparámetros.

Así concluyen que en modelos de aprendizaje no determinístico, como las redes neuronales, reportar únicamente puntajes de las ejecuciones no es suficiente, por lo que se deben comparar las distribuciones de los puntajes. Aunque este trabajo no está directamente relacionado con la tarea a resolver, nos orienta sobre cómo evaluar efectivamente el desempeño del modelo.

2.2. Mecanismos de atención

Se han realizado diferentes investigaciones sobre el problema de minería de argumentos agregándole un mecanismo de atención.

En el trabajo de Stab et al. [2018a] se busca encontrar argumentos relevantes a un tema dado. Para ello, se procesan grandes cantidades de documentos y se recuperan las porciones relacionadas a una consulta original. Se utiliza un modelo de BiLSTM con *inner attention*, que trata de medir la importancia de cada palabra dentro de una oración, a partir de la similitud entre cada una de ellas y las palabras del tema dado. Luego, cada palabra es pesada por su puntaje de importancia.

Este modelo es aplicado a textos heterogéneos que tratan sobre distintos temas, al contrario de la mayoría de modelos que son más

dependientes a un tipo de texto en particular. Sin embargo, requiere que el tema de la argumentación sea predefinido. Este no es siempre el caso para las tareas de argumentación, especialmente en documentos que contienen argumentaciones complejas, como transcripciones de juicios. Por ello, en este trabajo proponemos un método más general, que no requiere información sobre el contexto de la argumentación, como podría ser el tema.

Los mecanismos de atención también han sido utilizados para otras tareas de procesamiento de lenguaje natural con una semántica compleja.

Por ejemplo, en Koreeda et al. [2016], se construye una red neuronal con atención que, dado un par tema-valor (por ejemplo apuestas-crimen), clasifica si cierta evidencia contiene un argumento de relación promover/suprimir (por ejemplo el texto "casino aumenta el robo" contiene un argumento de relación "las apuestas promueven el crimen"). Se usa una técnica con embeddings de palabras que generaliza para temas-valores nunca vistos antes, por lo que el modelo aprende como clasificar evidencias que apoyan una relación, en vez de aprender la relación misma.

La atención de cada palabra es calculada teniendo en cuenta tanto el tema como el valor. Para la atención con respecto al tema se utiliza el embedding de cada palabra del tema y la salida correspondiente de la BiLSTM para cada palabra de la oración. Para la atención en el valor se utiliza el embedding de las palabras del valor, junto a lo obtenido en la atención del tema y la salida de la BiLSTM. El desempeño de este modelo supera a los modelos neuronales sin atención y efectivamente el modelo aprendió cómo clasificar las evidencias de las relaciones, en vez de la relación en sí.

Sin embargo, este problema está planteado de una manera distinta. El dominio de los datos es distinto, ya que se cuenta con los pares tema-valor. El puntaje de atención aplicado se calcula en relación a la importancia del tema, y luego se aplica a la salida de la BiLSTM. Por lo tanto, al contrario de nuestro trabajo, la atención es utilizada luego

de la BiLSTM y no antes. Además, los valores no poseen un puntaje de atención en sí mismos, sino sólo en relación al tema que se esté analizando.

Adicionalmente, el texto sobre el cual se trabaja consiste de oraciones mucho más cortas que las de nuestro corpus, por lo que tampoco podría garantizarnos resultados iguales para nuestro problema, ya que nuestro objetivo es capturar relaciones de un mayor alcance.

En el trabajo de Wang et al. [2016] se utiliza el método de *inner attention* en redes recurrentes para la tarea de selección de respuestas. Esta tarea consiste en, dada una pregunta, elegir la respuesta entre un conjunto de oraciones pre-seleccionadas. Los autores identifican que los pesos de atención en las oraciones están sesgados, ya que las últimas palabras de cada oración siempre tiene puntajes altos. La hipótesis propuesta es que esto sucede por la acumulación semántica y, para resolver este problema, se construyeron tres modelos recurrentes que agregan atención antes de la representación oculta en la RNN, a diferencia de modelos anteriores donde se agrega la atención después de la representación oculta. El primer modelo usa atención en la pregunta para ajustar la representación (embedding) de las palabras directamente en la respuesta. En el segundo modelo, se le suma a la atención del primero la representación contextual de cada palabra en la respuesta, i.e., el último estado oculto. En el tercer modelo se le pone atención en las compuertas de activación internas de la RNN para influenciar el cálculo de la representación oculta, y se agrega regulación en la sumatoria de los pesos de atención para imponer esparcidad.

Nuevamente, este problema se encuentra en un dominio distinto al nuestro, ya que se aplica atención sobre una pregunta, para luego utilizarla sobre las respuestas. Por otro lado, utilizan una celda de tipo GRU en la capa recurrente, lo cual difiere de nuestro trabajo, donde utilizaremos BiLSTM siguiendo a Eger et al. [2017].

Como podemos ver en estos dos últimos trabajos, la atención se aplica antes y después de la capa recurrente, obteniendo buenos resultados de ambas formas. Sin embargo, la interpretabilidad de la salida

de la capa recurrente no es tan clara, y tenemos la hipótesis de que la decisión de poner la atención antes o después de esta capa tendrá distintos efectos en ella. Es por esto que elegimos implementar la atención antes de la BiLSTM, dejando como un posible trabajo futuro el análisis de la atención después de la BiLSTM. Un análisis más profundo sobre esta hipótesis se puede encontrar en la sección 3.2.3, en el capítulo de Arquitectura.

Por otro lado, si bien estos trabajos se basan en dominios distintos al nuestro, han demostrado que la atención efectivamente ayuda al modelo a obtener mejores resultados. Esto nos motiva a llevar la atención a nuestro dominio, donde hay pocos conjuntos de datos anotados, las oraciones son mucho más largas, y se busca explotar relaciones de mayor alcance en el texto.

Capítulo 3

Arquitectura

3.1. Diseño de arquitectura

Nuestra arquitectura se basa en el trabajo de Nils Reimers, cuya implementación está disponible en GitHub [1]. Este sistema fue diseñado para tareas de etiquetado de secuencias tales como *part of speech tagging*, *named entity recognition*, y *chunking*. En cada una de ellas, al igual que en la minería de argumentaciones, el objetivo del clasificador es obtener una etiqueta para cada una de las palabras. Esto contrasta con las tareas de clasificación de secuencias, donde el modelo predice una única etiqueta para toda la secuencia. Esto determina el tipo de arquitectura recurrente que debemos utilizar.

En el paper Neural End-to-End Learning for Computational Argumentation Mining [Eger et al., 2017] se experimentó sobre esta red con distintos hiperparámetros, llegando a la conclusión de que el uso de el clasificador CRF (*Conditional Random Field*), el cual se basa en el contexto (i.e., las etiquetas de las palabras vecinas) al momento de clasificar, no mejora los resultados.

La arquitectura original consta de: una capa de embeddings de palabras y caracteres, una capa BiLSTM, y una capa CRF. Para evaluar mejor el impacto del mecanismo de atención, y siguiendo los resultados de Eger et al. [2017], hemos simplificado el modelo dejando sólo

[1] https://github.com/UKPLab/emnlp2017-bilstm-cnn-crf

una capa de embeddings de palabras y la capa de BiLSTM. De esta forma, podemos asegurar con mayor certeza de que las diferencias que encontremos en los resultados sean consecuencia de la atención y no de otro factor.

3.1.1. Embeddings de palabras

Para poder aplicar un clasificador neuronal recurrente a una oración, es necesario primero representar sus palabras numéricamente, a través de un vector numérico de características o *features*. A estas representaciones se les llama *embeddings*, y son generadas para cada una de las palabras o expresiones multipalabra del vocabulario. Existen diversas técnicas para optimizar esta representación, y en los últimos años han introducido mejoras revolucionarias en las tareas de PLN [Mikolov et al., 2013].

La forma más sencilla de representar una palabra es asignarle un identificador i y construir un vector del largo del vocabulario donde todos los valores son ceros, excepto por la posición i. De esta forma, cada palabra tiene una representación única. Sin embargo, este método, llamado *one-hot encoding*, tiene varias desventajas. Por un lado, cada vector tiene una alta dimensionalidad (la misma crece junto al vocabulario), y por otro, son ortogonales al resto de las representaciones, es decir, no podemos capturar similaridad entre palabras.

Los embeddings utilizados en este trabajo, por otra parte, generan una representación densa de las palabras que captura las regularidades semánticas y sintácticas del lenguaje, sobre la cual podemos aplicar operaciones matemáticas. Además de ello, estas representaciones son obtenidas a partir de grandes volúmenes de datos no etiquetados y pueden ser utilizados en diversas tareas. Los one-hot encodings no pueden capturar estas regularidades por sí solas, y dependen de lo que el modelo aprenda durante el entrenamiento, sólo en base a los ejemplos provistos. Otra diferencia es que los embeddings de palabras son usualmente de dimensionalidad baja (usualmente entre 50 y 600).

Los embeddings de palabras, al ser vectores densos, modelan también las relaciones semánticas como operaciones en dicho espacio multidimensional. Por ejemplo, si tomamos los vectores correspondientes y calculamos Rey - Hombre + Mujer, vamos a obtener un vector similar al de Reina. Este tipo de relaciones puede verse representado, a través de una simplificación en dos dimensiones, en la Figura 3.1[2].

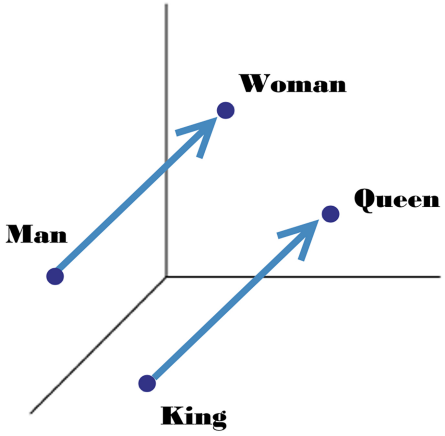


Figura 3.1: Word embeddings para hombre, mujer, rey y reina

En nuestros experimentos, usaremos los embeddings pre-entrenados de GloVe "Common Crawl" Pennington et al. [2014], los cuales tienen 42 mil millones de tokens y un vocabulario de 1.9 millones de palabras en vectores de 300 dimensiones.

Cada ejemplo que ingresa a esta capa es un ensayo completo, donde las palabras están separadas (tokenizadas) con el mismo método que utiliza Eger et al. [2017]. El output de esta capa serán los vectores con las representaciones numéricas de cada palabra del ensayo dado por los embeddings pre-entrenados de GloVe, mencionados anteriormente.

[2] https://www.oreilly.com/learning/capturing-semantic-meanings-using-deep-learning

3.1.2. Capa de BiLSTM

Luego de la capa de embeddings, el modelo utiliza una capa recurrente para procesar las secuencias. Las capas neuronales recurrentes tienen la propiedad de recordar los ejemplos vistos anteriormente. Mantienen un *estado oculto* en el cual se condensa toda la información de la secuencia necesaria para la tarea de clasificación. Las operaciones realizadas para calcular este estado oculto dependen del tipo de celda utilizado en la capa recurrente. Una red recurrente común utiliza las siguientes funciones para calcular la salida y el estado interno:

$$h_t = tanh(W^{(xh)}x^{(t)} + W^{(hh)}h^{(t-1)} + b^{(h)}) \tag{3.1}$$

$$y^{(t)} = softmax(W^{(hy)}h^{(t)} + b^{(y)})$$

donde el sufijo t indica el momento del tiempo o *timestep*, x es el vector que representa cada palabra, h es el vector de estado oculto de la recurrencia, y es el vector de salida, W son matrices con los parámetros de cada capa y b son los vectores de bias correspondientes.

Se puede observar que los parámetros para el cálculo del estado interno son el estado anterior $h^{(t-1)}$ y el nuevo elemento ingresado a la red $x^{(t)}$.

La función *tanh*, o tangente hiperbólica, es llamada en este contexto función de activación. Introduce una no-linearidad luego de la multiplicación del input por la matriz de pesos. Básicamente decide si una neurona debe ser activada o no, en base a si la información que esta neurona está recibiendo es relevante o debería ser ignorada. En general, se usa la función sigmoide o la tangente hiperbólica, aunque la función ReLU ha obtenido resultado empíricos muy favorables.

Una celda de LSTM (*Long short-term memory*) está compuesta por una compuerta de entrada, una de salida, y una de olvido. La celda es responsable de recordar valores durante intervalos de tiempos arbitrarios y cada compuerta puede pensarse como una neurona convencional (i.e., computan la activación de una suma ponderada). La compuerta

de entrada controla la medida en que un nuevo valor ingresa a la celda; la compuerta de olvido controla la medida en que un valor permanece en la celda, y la compuerta de salida controla la medida en el que el valor de la celda es usado para computar la activación de salida de la unidad de LSTM. Por lo tanto, una LSTM tiene las mismas entradas y salidas que una RNN, variando su estructura interna con la noción de "olvido".

Las redes recurrentes, y en particular las LSTMs, son adecuadas para clasificar, procesar, y predecir secuencias donde el tiempo y el número de eventos transcurrido entre eventos importantes es desconocido.

Esto se debe a que logran solucionar los problemas de *exploding* y *vanishing gradient* cuando las secuencias de entrada son muy largas. Estos problemas son propios del algoritmo de *back propagation through time*, donde se multiplican los gradientes en cada paso de la secuencia. El primer problema se da al multiplicar sucesivamente números mayores a 1, lo que causa que los gradientes tiendan a infinito (exploten). El segundo caso se da cuando la multiplicación es entre números menores a 1. Esto causa que los gradientes sean muy pequeños, impidiendo encontrar secuencias de largo alcance. El exploding gradient se soluciona con el uso de *clipping gradient*, el cual es un hiperparámetro que se encarga de "cortar" los gradientes y mantenerlos bajo un límite. Por otro lado, el problema de vanishing gradient es solucionado por las celdas LSTM gracias a la compuerta de olvido mencionada anteriormente, la cual se encarga de ir "apagando" celdas pasadas.

Es por esto que las redes recurrentes son utilizadas para minería de argumentaciones, ya que en principio debería ser capaz de darse cuenta de qué características (en este caso, palabras) son valiosas para la tarea o no, ya sea para mantenerlas a lo largo del tiempo, o para olvidarlas.

3.1.3. Relación de LSTM con el mecanismo de atención

Teniendo en cuenta lo mencionado anteriormente, podemos notar que el mecanismo de atención es intuitivamente parecido a lo que se espera que haga la LSTM, i.e., darse cuenta qué hay que "mirar" (atención) o qué hay que mantener en el tiempo (LSTM). Luego nos surge la pregunta de si la atención termina siendo redundante en una red recurrente.

No podemos asegurar que el mecanismo de atención provee a la arquitectura básica recurrente de la capacidad de representar más funciones. Esto es, en teoría, con y sin atención se podrían conseguir modelos adecuados de los mismos fenómenos. Sin embargo, es claro a partir de los resultados obtenidos por el estado del arte que el problema de minería de argumentaciones no es fácilmente resuelto por este tipo de modelos. Por ello, proponemos una mejora que puede permitir capturar el mismo fenómeno a partir de menos datos, identificando factores relevantes. Esta inyección de conocimiento de dominio es la que nos permite decirle a la red que se fije en qué palabras son importantes, porque intuitivamente vemos que son determinantes a la hora de identificar componentes argumentativos, ayudando a que el modelo converja más rápido. Por este motivo, nuestra hipótesis es que la atención en esta red no es redundante, y tendrá un impacto positivo en el rendimiento del clasificador. Si éste llegara a ser redundante (debido a la concentración de información que ya realiza la LSTM propiamente), entonces añadir este mecanismo a la arquitectura no va a mejorar el desempeño del modelo obtenido. Podremos responder a esta incógnita comparando el desempeño de clasificadores con y sin atención.

3.1.4. Entrenamiento de la red

Para entrenar una red neuronal, es decir, encontrar buenos valores para los parámetros del modelo, en general se utilizan diversos algoritmos basados en descenso por la gradiente o *gradient descent*, con una retropropagación de errores. Durante este procedimiento se realiza una

optimización iterativa de los parámetros a través de la minimización de la función de pérdida o función de *loss*.

Otra posibilidad es utilizar el descenso por el gradiente estocástico (*stochastic gradient descent*), una variación del algoritmo de descenso por el gradiente que actualiza el modelo utilizando el valor de la función de costo sobre un único ejemplo de entrenamiento. Esto implica una mayor eficiencia computacional, ya que no es necesario procesar el conjunto de entrenamiento en su totalidad antes de actualizar el modelo.

En nuestro caso utilizamos descenso por el gradiente por lotes pequeños (*mini batch gradient descent*), el cual divide el dataset de entrenamiento en pequeños lotes que son utilizados para calcular el error del modelo y actualizar los coeficientes del mismo. Este algoritmo busca encontrar un balance entre la robustez del descenso por el gradiente y la eficiencia del descenso por el gradiente estocástico.

Como resultado, el tamaño del batch de entrenamiento determina la rapidez del mismo. En el paso de optimización, se obtienen los gradientes de cada uno de los ejemplos del batch y se promedian antes de ser aplicados a los parámetros del modelo. Si los gradientes se calculan sobre varios ejemplos, se reduce la cantidad de actualizaciones realizadas, y por lo tanto el tiempo de entrenamiento. Sin embargo, un batch demasiado grande también tiene desventajas. Por un lado, puede consumir rápidamente los recursos de cómputo. Por el otro, al realizar pocas actualizaciones, se realentiza potencialmente la convergencia del clasificador, requiriendo más iteraciones hasta llegar a un punto de convergencia.

3.2. Mecanismo de atención

Para implementar el mecanismo de atención descripto en 1.4, agregamos una capa densa, utilizando *TimeDistributed* para aplicar el mismo conjunto de operaciones a cada timestep. Las fórmulas dependerán del enfoque de atención que se esté utilizando.

3.2.1. Word Attention

Para este enfoque de atención, la capa densa implementa la operación

$$attention\_score = activation(XW^{(ax)} + b)$$

donde X es nuestro input, i.e., una matriz con una fila por palabra en el ejemplo (timestep) y una columna por feature, a diferencia de la Ecuación 3.1, donde el vector x hace referencia a la representación de una única palabra. W es una matriz de pesos, y b es un vector llamado *bias*. La función de activación *activation*, se aplica a cada valor (*element-wise*).

Por lo tanto, el puntaje de atención es un valor numérico asignado a las palabras por la capa de atención, y su rango de valores depende de la función de activación utilizada. Nuestra interpretación sobre estos valores es que representan la importancia de las palabras y qué tanto deberían influir al momento de clasificar el componente argumentativo de la que forman parte.

Con el resultado de la capa densa obtenemos el puntaje de atención para cada feature de cada palabra. Luego, promediamos los puntajes de los features de cada palabra para obtener el puntaje de atención final de dicha palabra.

Finalmente, "pesamos" cada palabra con su puntaje de atención, multiplicando el vector original de la palabra con dicho puntaje.

$$averaged\_attention\_score_i = \frac{\sum_{j=1}^{n} attention\_score_{ij}}{n}$$

$$output\_vector = X \odot averaged\_attention\_score$$

donde $\odot$ hace referencia a la multiplicación punto a punto (o producto Hadamard) entre dos matrices, y n es la cantidad de features que tienen las palabras.

De esta manera, los valores de los vectores de cada palabra son aumentados o disminuidos de acuerdo a su puntaje de atención. A través del proceso de optimización, se espera que las palabras indicativas para la tarea de clasificación no sean apagadas, mientras que las palabras no indicativas podrían ser disminuidas para evitar el ruido.

Como podemos ver, las operaciones anteriores son diferenciables y por lo tanto se pueden optimizar sus parámetros durante el entrenamiento del modelo. Como resultado, el mecanismo de atención se puede agregar como una capa más, obteniendo la arquitectura presentada en la Figura 3.2.

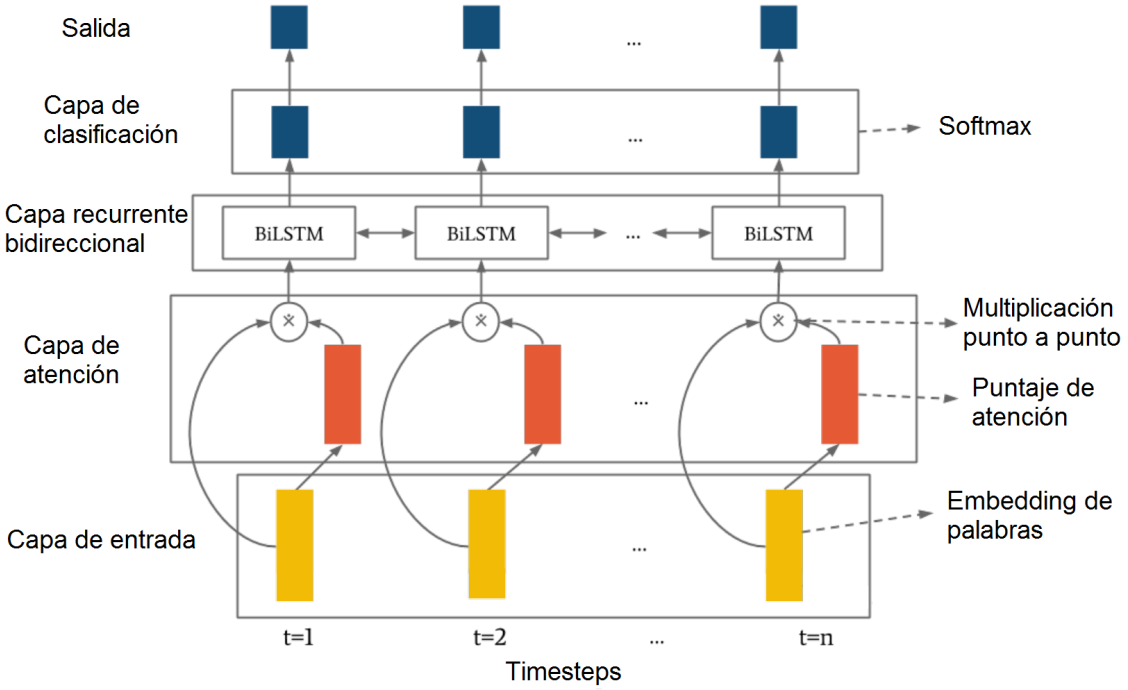


Figura 3.2: Arquitectura de nuestro modelo neuronal con atención

3.2.2. Context Attention

La diferencia entre *Word Attention* y *Context Attention* se basa en la forma en la que calculamos los puntajes de atención.

En el método de Word Attention, como se describió en la sección anterior, cuando aplicamos la capa densa, el puntaje de atención de cada palabra se calcula usando únicamente los valores de los features de esa palabra, es decir, en el mismo timestep. Esto significa que no se tienen en cuenta los features de las otras palabras. Como consecuencia, cada palabra tendrá el mismo puntaje de atención en todas sus menciones.

Por ejemplo, a partir de los experimentos realizados, hemos encontrado que los modelos detectan que la palabra "We" es importante en la clasificación de componentes argumentativos. Sin embargo, los modelos con Word Attention no pueden distinguir la relevancia de esta palabra cuando es usada de distintas formas. Si analizamos los ejemplos 3.2.2, podemos notar que en primer ejemplo, la palabra "We" es claramente indicativa de una *Claim*, mientras que en el segundo ejemplo el autor está refiriendo a una experiencia pasada no necesariamente relacionada a una argumentación.

Ejemplo 1 We should all vaccinate our kids

Ejemplo 2 When we were kids, vaccines were scary things.

Por otra parte, en Context Attention, el puntaje de atención de cada feature se calcula utilizando el valor de ese feature en todos los timesteps de las palabras en el contexto. Por lo tanto, nuestras nuevas ecuaciones para este tipo de atención, serán:

$$attention\_score = activation(X^T W^{(ax)} + b)^T$$

$$output\_vector = X \odot averaged\_attention\_score$$

En principio, nuestra intuición nos dice que la importancia de una palabra no debería estar determinada por sí misma, sino que también por las palabras con las que co-ocurre. Por lo tanto, parecería que tiene más sentido usar Context Attention, es decir *que tenga en cuenta el*

valor del mismo feature en todos los timesteps, y en base a eso decida qué tanta atención darle a ese feature. Esto permitiría a la atención tener información de las otras palabras en su contexto y por ende más poder discriminativo.

3.2.3. Antes o después de la recurrencia

El mecanismo de atención podría aplicarse antes de la capa de BiLSTM, o después. Nuestra intuición es que el mecanismo de atención tendrá efectos diferentes si se aplica en cada uno de estos dos contextos:

Antes de la LSTM Podemos interpretar mejor cómo impacta el input en la clasificación. En este caso, la atención se concentrará en las palabras individuales que más contribuyen a la clasificación de un ejemplo.

Después de la LSTM Podemos interpretar mejor qué parte de la información ya procesada es relevante para el cálculo del output. En este caso, la atención estaría pesando la salida de la capa recurrente, con lo cual se concentrará en los elementos que hayan concentrado información relevante. Aunque su impacto en el desempeño del modelo es incierto, la interpretabilidad que podemos darle a esta alternativa no es tan clara como en el caso anterior.

Debido a la diferencia de interpretabilidad que existe entre ambos contextos, en este trabajo evaluaremos el impacto de los mecanismos de atención aplicados antes de la capa recurrente, y dejaremos el segundo caso como trabajo futuro.

3.2.4. Funciones de activación

La atención cumple la función de distinguir las palabras más relevantes y aumentar relativamente el valor de la señal que producen.

Esperamos ver que un mecanismo de atención exitoso polarice los puntajes de atención asignados entre palabras importantes y palabras no importantes. El rango y la distribución de los valores dependerán del tipo de activación que se le aplica a la capa de atención.

Proponemos utilizar las siguientes funciones, que se encuentran entre las más comunes para modelos neuronales:

Lineal: La función de activación más simple es la función lineal, que corresponde a no realizar ninguna transformación. Este tipo de función no agrega expresividad a la red, ya que una combinación de funciones lineales continúa siendo una función lineal. Por lo tanto, la red no ganaría la capacidad de representar una mayor cantidad de funciones. Sin embargo, resta evaluar si sólo agregando una noción de puntajes de atención es suficiente para que los clasificadores converjan a un mínimo local más deseable.

$$f(x) = x$$

Sigmoid: Es una función no lineal y continuamente diferenciable. El rango de salida es (0,1), lo cual tiene la desventaja de que el gradiente se sigue actualizando cerca de estos límites, complicando la optimización. Sufre del problema de vanishing gradient, i.e. el gradiente toma valores muy pequeños aproximándose demasiado a 0 y la red termina no aprendiendo. Esto disminuye la intensidad de las activaciones y en principio no facilitaría obtener una buena interpretabilidad de la atención. Son de convergencia lenta. Aún así, es una de las funciones más utilizadas en Aprendizaje Profundo.

$$f(x) = \frac{1}{(1 + e^{-x})}$$

Tanh: Es una versión escalada de la sigmoide. Su rango de salida es (-1,1). A diferencia de la sigmoide, su salida es centrada en

0 y por lo tanto su optimización es más fácil, razón por la que en general se prefiere su uso. También sufre del problema del vanishing gradient.

$$tanh(x) = \frac{2}{(1 + e^{-2x})} - 1$$

ReLU: Es como una activación lineal pero sin valores negativos. Se probó que su convergencia es seis veces mejor que la de tanh. Evita y rectifica el problema de vanishing gradient. Es la más usada en los modelos de deep learning actuales. Sin embargo, su limitación es que debe ser utilizada solamente en las capas ocultas del modelo. Otra desventaja es que ciertos gradientes pueden ser demasiado frágiles durante el entrenamiento y terminar apagándose. El inconveniente que tenemos con esta función de activación, es que al no tener una idea concisa sobre como interpretar la atención, no sabemos que tan correcto puede ser "apagar" un valor cuando éste es negativo

$$r(x) = max(0, x)$$

Softmax: Es una generalización de la función logística. Se emplea para "comprimir" un vector K-dimensional, z, de valores reales arbitrarios en un vector K-dimensional, $\sigma(z)$, de valores reales en el rango [0, 1] con norma 1. La salida de la función softmax puede ser utilizada para representar una distribución categórica, i.e., la distribución de probabilidad sobre K diferentes posibles salidas. Es por esto que es ampliamente utilizada en tareas de clasificación.

$$softmax(x_i) = \frac{\exp(x_i)}{\sum_j \exp(x_j)}$$

3.3. Detalles de implementación

Para poder calcular la atención tuvimos que padear las secuencias a un largo fijo, donde padear es la técnica de "rellenar" con ceros los vectores que representan a los ensayos, hasta completar el tamaño máximo que hemos fijado (el ensayo más largo). El padeo fue necesario debido a restricciones de implementación de la librería *Tensorflow*.

Este padeo no afecta el resultado final tanto de predicciones como de gradientes para la optimización: se inserta un capa de máscara (*Masking*) en la primera capa de embeddings, que se propaga hasta el cálculo de la función loss y asegura que las computaciones que involucran a los elementos padeados sean ignoradas.

Por otro lado, para el mecanismo de Context Attention, una de las operaciones (permute) no permite la propagación de la capa de Masking. Para contrarrestar esto, agregamos la capa de Masking luego de la aplicación de la atención. El resultado final es el mismo, ya que como las secuencias fueron padeadas con ceros, al multiplicar los vectores de palabras con los puntajes de atención, el resultado sigue siendo nulo al ingresar a la capa de Masking.

Capítulo 4

Experimentos

4.1. Hipótesis

Para evaluar el impacto de la atención en nuestros modelos, se plantean las siguientes hipótesis:

Primera hipótesis: Los mecanismos de atención propuestos mejorarán el desempeño de los clasificadores en la tarea de minería de argumentaciones.

Segunda hipótesis: Las distintas funciones de activación generarán clasificadores con distinto desempeño. Nuestra intuición es que las funciones no lineales, debido a su mayor poder de expresividad, serán las que mejores rendimientos obtengan.

4.2. Corpus

En Stab and Gurevych [2017] se introduce un corpus de ensayos persuasivos anotados con estructuras argumentativas. Dicho corpus contiene ensayos de estudiantes escritos en respuesta a temas controversiales como "¿Competición o cooperación? ¿Cuál es mejor?". El mismo fue etiquetado siguiendo un esquema y guía de anotación que facilita el acuerdo entre humanos. Tal esquema se basa en anotar los componentes argumentativos en base a las etiquetas de MajorClaim, Claim, y Premise, siguiendo las definiciones introducidas en el capítu-

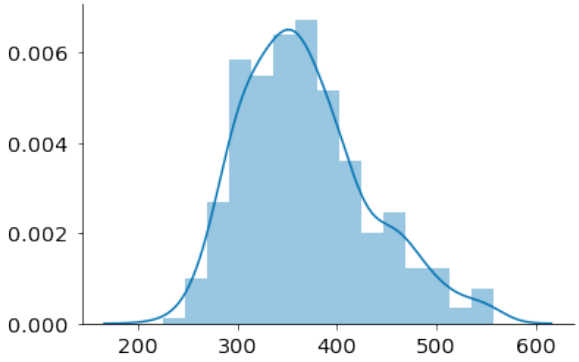


Figura 4.1: Distribución de la cantidad total palabras para cada ensayo.

Figura 4.2: Estadísticas del corpus.

	Train	Test
Essays	322	80
Paragraphs	1786	449
Tokens	118648	29538

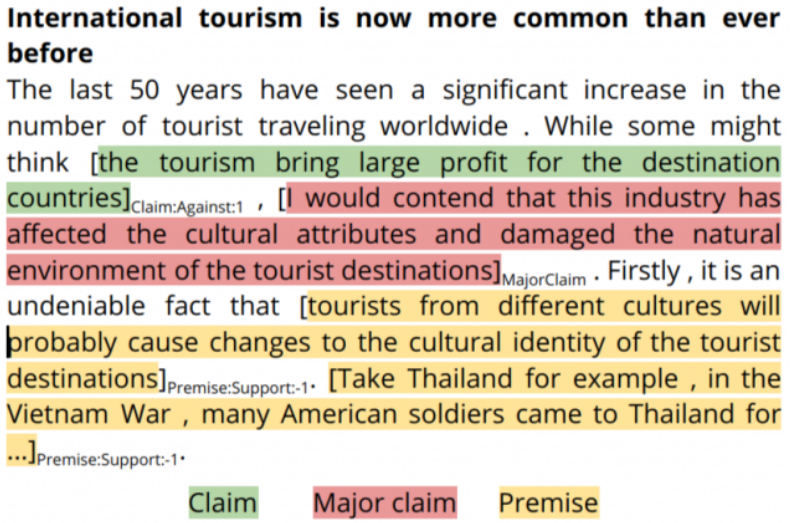
lo 1. Los ensayos fueron obtenidos de un foro online, el cual consiste de una comunidad activa que provee correcciones y comentarios sobre diferentes textos, como papers de investigación, ensayos, o poesía. El dataset está disponible para descargar libremente [1].

Este corpus y otras versiones anteriores han sido utilizados para diferentes tareas de minería de argumentos desde diversas perspectivas. En Nguyen and Litman [2015] se analiza un método para reducir el espacio de características esparzas, tanto léxicas como sintácticas. En Nguyen and Litman [2016] se presenta un modelo independiente de contexto para el minado de relaciones argumentativas, y en Eger et al. [2017] se utiliza un modelo neuronal *end-to-end*, tratando el problema como un parseo de dependencias y etiquetado de secuencias.

Utilizaremos este corpus para analizar los efectos del mecanismo de atención, mientras que para ello adaptamos el modelo desarrollado en Eger et al. [2017] para evaluar nuestro enfoque propuesto.

[1] https://www.informatik.tu-darmstadt.de/ukp/research_6/data/argumentation_mining_1/argument_annotated_essays_version_2/index.en.jsp

Figura 4.3: Fragmento de un ensayo con sus anotaciones originales.



Como podemos ver en la Figura 4.2, el corpus consiste de 402 ensayos y 148,587 palabras. De estos, 286 se usan para *training*, 36 para *dev* y 80 para *testing*. La distribución de palabras por cada ensayo responde a la distribución graficada en la Figura 4.1. Cada ensayo se divide en párrafos, de los cuales hay 2235 en total, y la estructura argumentativa está contenida completamente dentro de cada párrafo. Éstos tienen una cantidad promedio de 66 palabras cada uno, mientras que los ensayos tienen un promedio de 368 palabras. Por lo tanto, en promedio, cada ensayo contiene entre 5 y 6 párrafos.

La anotación distingue entre MajorClaim, Claim y Premise. En total hay 751 MajorClaims, 1506 Claims y 3832 Premises. Adicionalmente, incluye información sobre las relaciones entre ellas, de ataque (*Against*) o de apoyo (*Support*).

En nuestra tarea de clasificación, solamente utilizaremos como etiquetas el tipo de componente argumentativo, es decir, Claim, Premise y MajorClaim. En la Figura 4.3 mostramos como ejemplo un fragmento de un ensayo con sus respectivas anotaciones.

4.3. Comparación con otros sistemas

4.3.1. Baselines

Para evaluar el impacto del mecanismo de atención, comparamos los resultados del método propuesto con los obtenidos por el modelo BiLSTM base, adaptado al problema de minería de argumentos. Este modelo proviene del paper Neural End-to-End Learning for Computational Argumentation Mining [Eger et al., 2017], con las simplificaciones propuestas en la Sección 3.1.

4.3.2. Métricas

Existen diversas métricas para medir el desempeño de modelos en problemas de clasificación. Entre las más utilizadas se encuentran las siguientes:

Accuracy: Es la proporción de ejemplos correctamente etiquetados, con respecto a la cantidad de ejemplos de evaluación totales.

Esta métrica es ampliamente utilizada en tareas de clasificación, aunque es más adecuada para clasificaciones binarias con etiquetas balanceadas. Una desventaja es que la misma opaca información referente a la distribución de las etiquetas correctas e incorrectas pertenecientes a cada clase.

Es por esto que utilizar sólo esta métrica no sería de mucha utilidad, ya que podría darnos resultados poco informativos. Por ejemplo, supongamos que tomamos un fragmento de ensayo de 10 palabras, donde 1 es una premisa y las restantes no tienen etiqueta, y el clasificador no etiqueta ninguna premisa. Luego, tendremos que el accuracy de ese fragmento de ensayo es 0.9, ya que acertó 9 de 10 veces en no etiquetar ninguna palabra como premisa.

Por lo tanto, elegimos también trabajar con las siguientes métricas más detalladas.

Precision: Indica con qué certeza un resultado obtenido es correcto. La precisión se calcula clase por clase, viendo el problema como

one-vs-all, donde la clase en cuestión se toma como positiva y todas las restantes clases como negativas. Está definida por la siguiente fórmula:

$$TP/(TP + FP)$$

donde TP se refiere a *True positive* (se etiqueta una palabra que debe etiquetarse) y FP significa *False positive* (se etiqueta una palabra que no debe etiquetarse).

Supongamos que en el ejemplo anterior el clasificador etiqueta como premisa a la palabra correcta y a otra palabra que no lo es. La precisión de la etiqueta Premise sería de 0.5, ya que de las premisas etiquetadas, sólo la mitad es correcta.

Recall: Indica el porcentaje de cobertura sobre los elementos que debe etiquetar correctamente. Corresponde a la fórmula:

$$TP/(TP + FN)$$

donde FN hace referencia a *False negative* (no se etiqueta una palabra que debería etiquetarse). Supongamos que en el ejemplo anterior se clasifican todas las palabras como premisas. Luego tendríamos un recall de 1 para Premise, ya que se etiquetó correctamente la única premisa que contenía el fragmento de ensayo.

F1: Para combinar precision y recall en un solo valor numérico, se utiliza el F1. El resultado ideal sería un F1 de 1, mientras que el peor resultado se daría con un F1 de 0.

$$2 * (precision * recall/precision + recall)$$

La métrica utilizada para reportar resultados es el **F1 con un promedio macro**, que pesa el F1 individual de cada clase de la misma manera. El promedio macro es el más adecuado para comparar resultados obtenidos en diferentes particiones del conjunto de datos[2].

En el trabajo de Reimers and Gurevych [2017] se evalúa la misma arquitectura en cuatro tareas de etiquetado de secuencias: etiquetado

[2] https://medium.com/@ramit.singh.pahwa/micro-macro-precision-recall-and-f-score-44439de1a044

gramatical, reconocimiento de entidades nombradas, fragmentación de texto y reconocimiento de eventos. En este trabajo se hace hincapié en que reportar el mejor F1 no garantiza una evaluación adecuada, y es necesario graficar la distribución de tales resultados. Esto se debe a que modelos con los mismos hiperparámetros pueden generar resultados distintos, porque dependen de la inicialización de sus parámetros. Siguiendo esto, también reportaremos las distribuciones de todas las combinaciones de hiperparámetros que hemos experimentado.

4.4. Entorno de experimentación

Los experimentos se realizaron seleccionando distintos hiperparámetros de forma aleatoria, los cuales se describirán en detalle en la sección 4.5 Resultados. Los hiperparámetros restantes que no han sido mencionados tomaron los valores recomendados por el framework *Keras*.

Todos los modelos fueron entrenados con 100 épocas. Sin embargo, si el modelo no registra un aumento del F1 en 10 épocas, el entrenamiento es finalizado prematuramente. Dicho F1 fue tomado sobre el corpus de dev, para evitar el sobreajuste indirecto.

Cada modelo se entrenó sobre el conjunto de datos de entrenamiento. Luego, seleccionamos el mejor modelo de cada arquitectura (sin atención, Word Attention y Context Attention) en base al valor de F1 más alto durante el entrenamiento sobre el conjunto de datos de test. Es importante notar que este valor no corresponde necesariamente a la última época de entrenamiento, sino a la mejor de todas.

Cada ejemplo provisto al clasificador es un ensayo completo, del cual se pueden extraer dependencias de largo alcance entre componentes argumentativos pertenecientes a distintas oraciones.

4.4.1. Optimizador

El sistema original del estado del arte utiliza el optimizador Adam. Sin embargo, en estudios preliminares realizados con el mismo conjun-

to de datos se observó que el optimizador Nadam obtenía resultados similares en mucho casos y superiores en otros. Como resultado, durante la experimentación todos los modelos son optimizados con la implementación de Nadam, de Keras[3].

4.5. Resultados

En esta sección presentamos los resultados de los experimentos llevados a cabo, para responder a las preguntas planteadas en la sección 4.1.

4.5.1. Rendimiento en la tarea de clasificación

Cuadro 4.1: Mejores resultados de la métrica F1 obtenidos con las tres arquitecturas propuestas, sobre los datasets de dev y test.

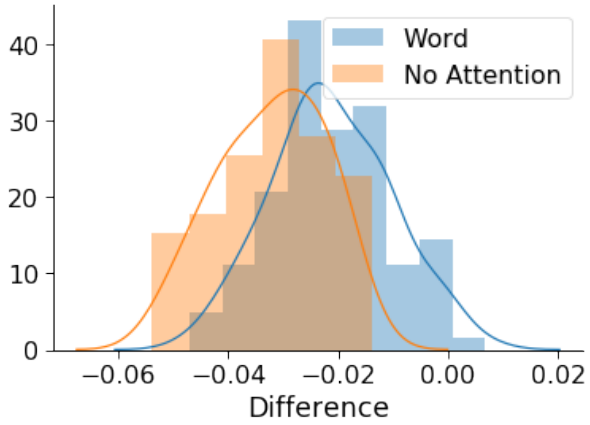
	Dev F1	Test F1
Sin Atención	0.716	0.748
Word Attention	**0.728**	**0.754**
Context Attention	0.707	0.729

Como podemos ver en 4.1, la arquitectura con el mecanismo de Word Attention efectivamente mejora el rendimiento del modelo. Sin embargo, la arquitectura con Context Attention no sólo parece no introducir mejoras en el desempeño, sino que lo perjudica.

Por otro lado, cabe recalcar que en algunos modelos, los resultados en el corpus de test no siguen el mismo patrón que los resultados en el corpus de dev, es decir, que el valor del F1 aumenta en mayor proporción en uno que en el otro. En la Figura 4.4 se muestran las distribuciones de la diferencia entre ambos conjuntos de datos. En general, observamos que la diferencias entre ambos valores de F1 son menores en los modelos con Word Attention.

[3] https://keras.io/optimizers

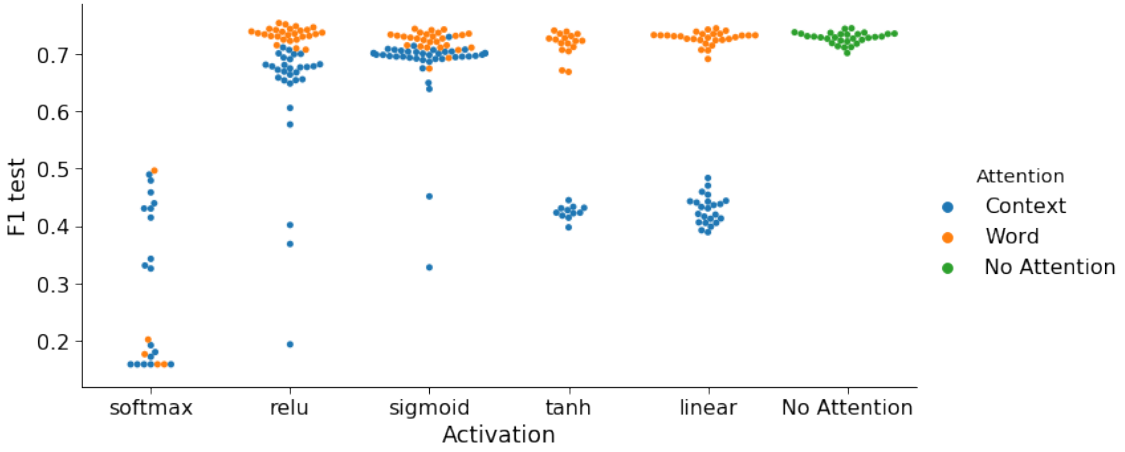
Figura 4.4: Distribución de la diferencia entre el puntaje F1 en el corpus de dev y en el corpus de test, de acuerdo al tipo de atención utilizado.



Realizando un análisis más detallado de los valores de F1 de cada clase para los modelos sin atención (lo cual se profundiza en la sección 4.7.1), notamos que la diferencia en F1 total se debe a una considerable mejora en el precision y recall de la clase Claim, que es el punto más débil en todos los modelos entrenados. No tenemos una hipótesis sobre este comportamiento, pero podría ser evidencia de que los modelos que no utilizan atención son menos robustos.

En la Figura 4.5 reportamos el desempeño de todas las combinaciones de hiperparámetros entrenadas. Podemos observar que, en general, en modelos con Word Attention el impacto de la función de activación es menos relevante, ya que la mayoría de los resultados tienen una distribución similar, con excepción de la función softmax. Analizaremos más en profundidad este fenómeno en las secciones siguientes. Por otro lado, los modelos con Context Attention sólo obtienen resultados comparativos para las activaciones ReLU y sigmoide. Por ello, podemos concluir que los modelos con Word Attention son mucho más robustos. Más aún, podemos observar que existen varios modelos con distintas configuraciones que superan a los modelos sin atención, determinando que no es un fenómeno aleatorio.

Figura 4.5: Resultados de F1 para todos los modelos entrenados durante la optimización de hiperparámetros.



4.5.2. Impacto de la función de activación

En esta sección analizaremos el desempeño de las distintas funciones de activación y sus causas. Ya hemos analizado los clasificadores con Context y Word Attention desde un punto de vista general, concluyendo que los mejores resultados se obtienen con Word Attention. Analizaremos ahora como se comportan estos tipos de atención con las distintas funciones de activación, y sólo incluiremos los resultados en los que se hallaron patrones interesantes para analizar, excluyendo aquellas que no aportaban información adicional.

Activación Sigmoide

A partir de la Figura 4.6 podemos ver que el modelo de Word Attention más exitoso con activación sigmoide, asigna un puntaje de atención cercano a 1 a la mayoría de las palabras, lo que podría indicar que hayan palabras que no sean importantes y aún así obtengan demasiada atención. Sin embargo, esta alta diferencia de atención con el resto de las palabras podría ser positiva al momento de darle peso a las palabras que realmente importan, obteniendo el comportamiento

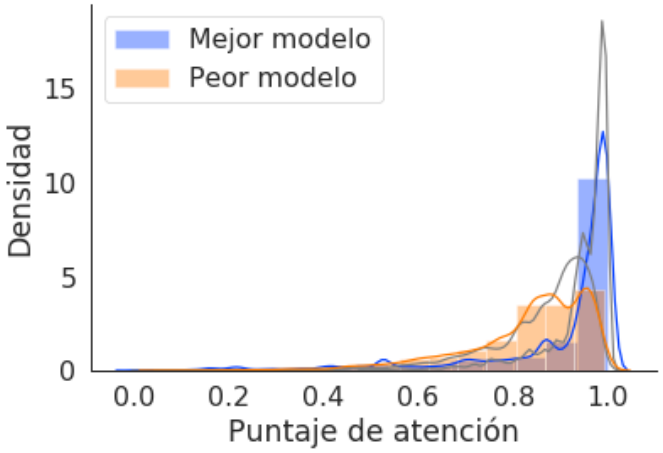


Figura 4.6: Distribución de los puntajes de atención para el mejor y peor modelo con Word Attention y activación sigmoide, con modelos promedio en gris.

que esperamos. También observamos pequeños aumentos de la densidad de probabilidad para puntajes bajos, lo cual indica que el modelo está efectivamente apagando las señales de las palabras que considera ruidosas.

Por otro lado, para el peor modelo vemos que los puntajes de atención se distribuyen de una forma más uniforme, la mayoría tomando valores entre 0.8 y 1. Sin embargo, que haya tanta uniformidad en el rango de estos puntajes indica que la atención podría no dar pesos suficientemente discriminatorios para afectar a las decisiones futuras de la red, lo cual se aleja del comportamiento deseado. Si analizamos los modelos intermedios, vemos que tienden a replicar el comportamiento del mejor modelo; uno de ellos asignando mayoritariamente una atención muy cercana a 1, y otro siendo un término medio entre el peor y el mejor modelo.

En 4.7a y 4.7b vemos que las palabras que más atención obtuvieron en el mejor modelo, fueron palabras como “I”, “My”, “We”, que indican el inicio de alguna opinión, por lo que tiene sentido que obtengan un puntaje alto. Notar que la palabra “Disagree” también obtuvo un puntaje alto de atención. Por otro lado, las palabras que

word	true	prediction	attention
i	Premise	Premise	0.999839
me	Premise	Premise	0.999802
we	Premise	Premise	0.999500
disagree	O	O	0.999345
you	Premise	Premise	0.999343
do	Premise	Premise	0.999133
my	Premise	Premise	0.999082
see	Premise	Premise	0.998916
him	Premise	Premise	0.998811
they	Premise	Premise	0.998775

(a) Palabras con mayor atención

word	true	prediction	attention
cattles	Premise	Premise	0.067110
indepedent	O	O	0.115994
adventurousness	Premise	Premise	0.140423
transportations	O	O	0.142618
equips	O	O	0.152127
polices	Premise	Premise	0.168429
snuffing	Premise	Premise	0.168937
unresponsiveness	O	Premise	0.174562
salesmanship	Premise	Premise	0.176234
handcraft	Claim	Premise	0.184342

(b) Palabras con menor atención

Figura 4.7: Puntajes de atención utilizados por el mejor modelo con Word Attention y activación sigmoide.

menos atención obtuvieron fueron palabras como “cattles”, “handcraft”, “transportations”, que son sustantivos y claramente aportan poca información a la hora de saber si se trata de un argumento o no.

La distribución de atención del mejor y peor modelo de Context Attention obtenido con activación sigmoide puede observarse en la Figura 4.9. Respecto al mejor modelo, vemos que la atención se distribuye principalmente entre 0.4 y 0.6, siendo 0.55 el puntaje al que más palabras se le asignó. En este caso, podemos ver en 4.8a que los puntajes de atención más altos, son asignados a palabras que indican argumentación, como lo son “and”, “if” y “as”, aunque también aparecen sustantivos con atención similar. Por otro lado, en 4.8b podemos observar las palabras con menor puntaje de atención. Llamativamente, la palabra “should” obtuvo el puntaje más bajo de todos a pesar de ser claramente una palabra que indica argumentación. Aún así, el resto de los puntajes más bajos de atención fue para palabras no determinantes. Luego, podría suponerse que en este caso la atención no mejoró el rendimiento de nuestra red debido a que se asignó mayoritariamen-

word	true	prediction	attention
and	Premise	Premise	0.692373
in	Premise	Premise	0.691403
elderly	Claim	Claim	0.691118
mobile	Premise	Premise	0.688843
out	Premise	Premise	0.687474
time	Premise	Premise	0.683305
this	Premise	Premise	0.681893
related	Premise	Premise	0.681460
their	Premise	Premise	0.680959
,	Premise	Premise	0.676798
the	Premise	Premise	0.676176
if	Premise	Premise	0.675859
increasing	Premise	Premise	0.675100
as	Premise	Premise	0.674765

(a) Palabras con mayor atención

word	true	prediction	attention
should	Claim	Claim	0.353015
advantage	Claim	Claim	0.354706
on	Claim	Claim	0.357389
,	Claim	Claim	0.362528
people	Claim	Claim	0.365307
personally	O	MajorClaim	0.366905
.	O	O	0.367478
endangered	Claim	Claim	0.369075
places	MajorClaim	Claim	0.371673
breed	MajorClaim	MajorClaim	0.371874
an	Claim	Claim	0.372629
will	Claim	Claim	0.373153
in	Claim	Claim	0.373244
completely	O	O	0.373459

(b) Palabras con menor atención

Figura 4.8: Puntajes de atención utilizados por el mejor modelo con Context Attention y activación sigmoide.

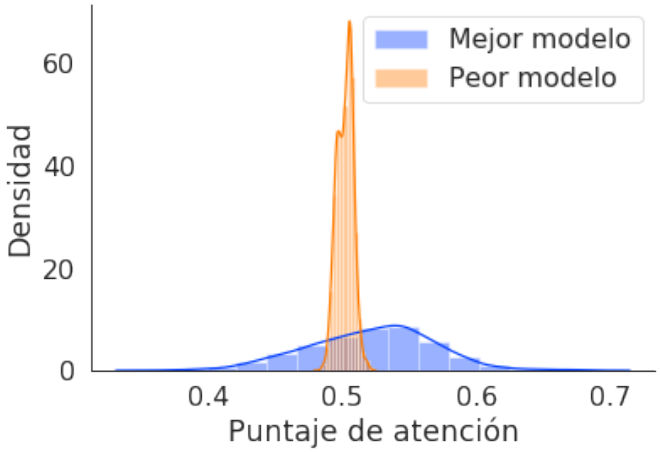


Figura 4.9: Distribución de los puntajes de atención para el mejor y peor modelo con Context Attention y activación sigmoide

te un puntaje medio de atención a las palabras. Para el peor modelo podemos ver que los puntajes de atención no lograron converger, ya que la distribución observada se corresponde con la inicialización de la atención con activación sigmoide.

Activación Lineal

En la Figura 4.10 podemos ver las distribuciones de atención del mejor y peor modelo de Word Attention con activación lineal, así como también otros dos modelos promedio. Lo que se puede observar acerca del mejor modelo, es que la mayoría de los puntajes de atención son negativos, siendo -2 el promedio del puntaje más obtenido. Es importante notar que el signo de la atención no es relevante para el modelo, sino el valor absoluto de la misma. Esto se debe a que dicho valor se corresponde con la intensidad de la activación de la celda.

En 4.11a podemos ver que las palabras con puntaje de mayor valor absoluto, son las más determinantes, como “I”, “Me”, “We”, “You”, y las de puntaje con menor valor absoluto, son las menos determinantes, como “cattles” o “snuffing”

Por otro lado, el peor modelo se centró en los puntajes positivos,

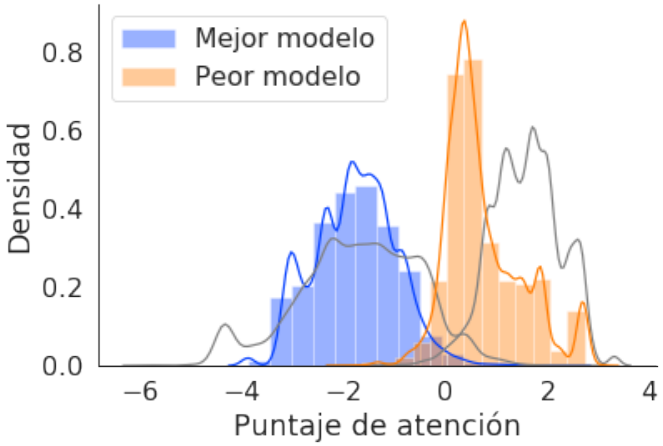


Figura 4.10: Distribución de los puntajes de atención para el mejor y peor modelo con Word Attention y activación lineal, con modelos promedio en gris.

word	true	prediction	attention
i	Premise	Premise	-3.866696
me	Premise	Premise	-3.257548
we	Premise	Premise	-3.247421
you	Premise	Premise	-3.234020
do	Premise	Premise	-3.197834
not	Premise	Premise	-3.162609
think	Premise	Premise	-3.138895
they	Premise	Premise	-3.093038
.	Premise	O	-3.056715
reason	Premise	Premise	-3.055725

(a) Palabras con mayor atención

word	true	prediction	attention
cattles	Premise	Premise	2.459534
indepedent	O	O	1.471166
toefl	Premise	Premise	1.448701
ibt	Premise	Premise	1.262604
unresponsiveness	O	O	1.241308
snuffing	Premise	Premise	1.195614
transportations	O	O	1.186324
adventurousness	Premise	Claim	1.122684
organising	O	MajorClaim	1.103902
stegosaurus	Premise	Premise	1.047559

(b) Palabras con menor atención

Figura 4.11: Puntajes de atención utilizados por el mejor modelo con Word Attention y activación lineal.

pero obteniendo un comportamiento similar al mejor modelo, donde se asignaron los menores puntajes a palabras como “cattles”. La diferencia entre este modelo y el mejor, es que la distribución de los puntajes fue menos uniforme, centrándose en valores cercanos a cero.

Es posible que al apagar tantas señales, el modelo no haya podido converger correctamente, explicando su bajo rendimiento.

Los modelos promedio obtuvieron distribuciones similares al mejor y peor modelo. Estos resultados nos indican que, en este caso, un mejor comportamiento de nuestro modelo no depende de si la atención se centra en valores negativos o positivos, sino en que esté distribuida en intervalos con valor absoluto mayor a uno. Contrariamente a la activación sigmoide, una distribución con mayor desviación estándar obtiene mejores resultados, ya que la imagen de la función no está limitada a valores entre 0 y 1.

Podemos notar un paralelismo entre las palabras que obtienen mayor puntaje en los modelos con activaciones distintas. Este fenómeno indicaría que el mecanismo de atención no depende de la forma de la función de activación para discriminar palabras importantes, sino de que los valores estén lo suficientemente separados.

Activación ReLU

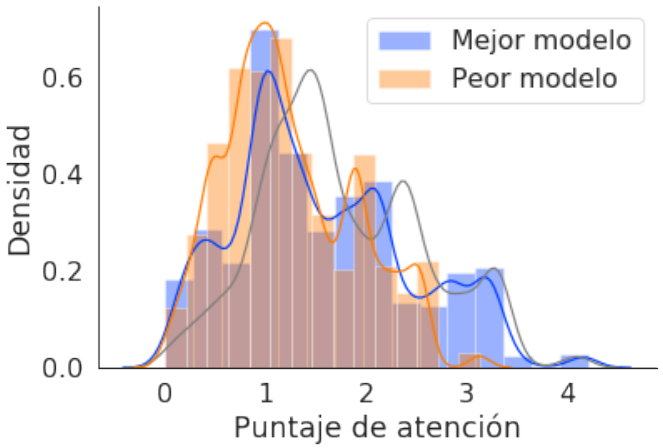


Figura 4.12: Distribución de los puntajes de atención para el mejor y peor modelo con Word Attention y activación ReLU, con modelo promedio en gris.

La Figura 4.12 indica que, en general, los puntajes de atención de los modelos con esta activación se distribuyen uniformemente entre 0 y

word	true	prediction	attention
disagree	O	O	4.221425
i	Premise	Premise	4.171544
reasons	O	O	4.132561
reason	Premise	Premise	3.805883
wrong	Premise	O	3.539518
arguments	O	O	3.536747
conclusion	O	O	3.500906
argued	O	O	3.457768
however	O	O	3.439439
factors	O	MajorClaim	3.416589

(a) Palabras con mayor atención

word	true	prediction	attention
incoming	Premise	Premise	0.000000
cattles	Premise	Premise	0.000000
ibt	Premise	Premise	0.001160
prepares	Premise	Premise	0.001359
dormitories	Premise	Premise	0.005190
dormitory	Premise	Premise	0.007910
normality	Premise	Premise	0.008621
toefl	Premise	Premise	0.010927
bounded	Premise	Premise	0.011645
legendary	Premise	Premise	0.020044

(b) Palabras con menor atención

Figura 4.13: Puntajes de atención utilizados por el mejor modelo con Word Attention y activación ReLU.

5. Al momento de analizar los puntajes de atención asignados del peor y mejor modelo, si bien ambos parecen asignar un mayor puntaje a las palabras más determinantes, podemos ver que la principal diferencia es el puntaje máximo que asigna cada uno. Por un lado, el mejor modelo llega a asignar 4.22 de atención a la palabra "disagree", mientras que el peor modelo asigna un máximo de 3.13 a la palabra "I". También podemos ver que ambos modelos asignan un valor pequeño a palabras poco determinantes, como "cattles" o "ibt".

De acuerdo a estos resultados, la diferencia del puntaje máximo que se asigna a las palabras podría ser una de las causas de que un modelo se comporte mejor que el otro, ya que un mayor rango en la distribución de puntajes permite discriminar mejor entre palabras relevantes y no relevantes. Sin embargo, la diferencia en desempeño que observamos entre estos modelos con Word Attention es bastante pequeña. Esta diferencia en el rango de la distribución de los puntajes de atención se observa mejor en modelos con Context Attention, como muestra la Figura 4.15. El mejor modelo que utiliza Context Attention

word	true	prediction	attention
i	Premise	Premise	3.136107
disagree	O	O	2.688388
.	Premise	O	2.541772
me	Premise	Premise	2.498973
reason	Premise	Premise	2.481952
reasons	O	O	2.411316
however	O	Premise	2.396765
think	Premise	Premise	2.395903
do	Premise	Premise	2.379752
my	Premise	Premise	2.373171

(a) Palabras con mayor atención

word	true	prediction	attention
cattles	Premise	Premise	0.000683
prepares	Premise	Premise	0.000712
organising	O	O	0.003246
ibt	Premise	Premise	0.004226
dormitories	Premise	Premise	0.006606
aimed	Premise	Premise	0.008280
incoming	Premise	Premise	0.009228
dormitory	Premise	Premise	0.009890
indepedent	O	O	0.011803
snuffing	Premise	Premise	0.013088

(b) Palabras con menor atención

Figura 4.14: Puntajes de atención utilizados por el peor modelo con Word Attention y activación ReLU.

obtiene resultados cercanos al peor modelo con Word Attention, y podemos observar que tienen una distribución de puntajes similares en un rango entre 0 y 3. A su vez, el peor modelo obtiene resultados notablemente inferiores que el mejor modelo con Context Attention, y sus puntajes de atención se movieron en un rango aún menor.

Sin embargo, a la hora de ver los puntajes asignados como muestra la Figura 4.16a, vemos que las palabras que aparecen con mayor puntaje en modelos de Context Attention no son significativas para la tarea de minería de argumentaciones (con la excepción de “thus” y “since”), como sí lo eran en los modelos de Word Attention. Incluso las palabras con menor puntaje tampoco son significativas. De todas formas, teniendo en cuenta la observación anterior, podemos pensar que son palabras que recibieron poco puntaje por aleatoriedad, y no por el hecho de no ser significativas. El peor modelo, que obtuvo un F1 comparativamente muy bajo, asignó los puntajes de atención de una forma similar, i.e., asignó puntajes altos a palabras poco influyentes.

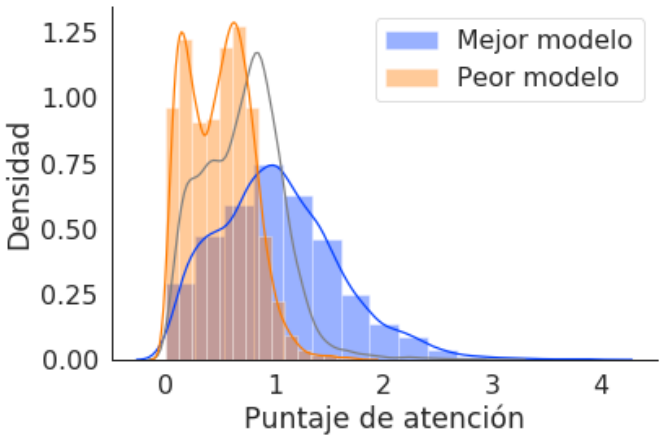


Figura 4.15: Distribución de los puntajes de atención para el mejor y peor modelo con Context Attention y activación ReLU, con modelo promedio en gris.

word	true	prediction	attention
teachers	Premise	Premise	4.022037
some	Premise	Premise	3.892876
.	Premise	Premise	3.856081
to	Premise	Premise	3.853708
rewarded	Premise	Premise	3.840741
thus	O	O	3.825558
activities	Premise	Premise	3.818546
since	O	Premise	3.766023
particularly	Premise	Premise	3.751031

(a) Palabras con mayor atención

word	true	prediction	attention
maintenance	O	O	0.000065
advance	O	O	0.008769
vs	O	O	0.008769
easing	O	O	0.009448
humans	O	O	0.009583
exploration	O	O	0.056948
releasing	O	O	0.063615
wisdom	O	O	0.080758
primary	O	O	0.086964
adaptability	O	O	0.089204

(b) Palabras con menor atención

Figura 4.16: Puntajes de atención utilizados por el mejor modelo con Context Attention y activación ReLU.

A partir de esto podemos concluir que el modelo de Context Attention no aplica puntajes de atención más grandes a las palabras significativas. Así mismo, tanto modelos medianamente exitosos como modelos con bajo desempeño, asignan puntajes altos a las mismas pa-

labras pero con distinta magnitud. Por lo tanto, es más importante a qué palabras se asigna más atención que el puntaje de atención en particular.

Activación Softmax

La distribución de puntajes de atención con activación softmax, para el modelo con Word Attention, puede observarse en la Figura 4.17. Como podemos ver, todos los puntajes de atención se centraron en valores muy pequeños, tanto para el mejor como para el peor modelo obtenido. Esto se debe a que la activación softmax, como se explicó en la sección 3.2.4, asigna probabilidades como puntaje. Si tenemos en cuenta la cantidad de valores que tiene la distribución (features para el caso de Word Attention, y palabras en el ensayo para el caso de Context Attention), es claro que dichas probabilidades se van a desvanecer y por lo tanto van a terminar "apagando" las señales en la red. Esto no sólo implica que la red no va a tener mayor información a la hora de aprender, como quisiéramos, sino que va a tener aún menos, afectando negativamente el desempeño de nuestro modelo.

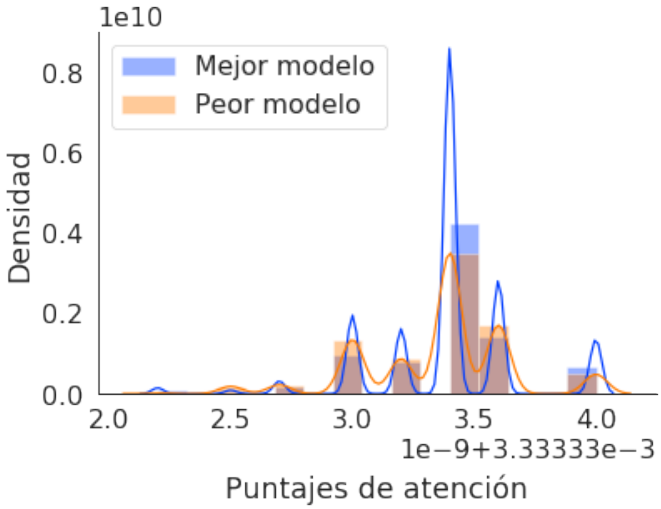


Figura 4.17: Distribución de los puntajes de atención para el mejor y peor modelo con Word Attention y activación softmax.

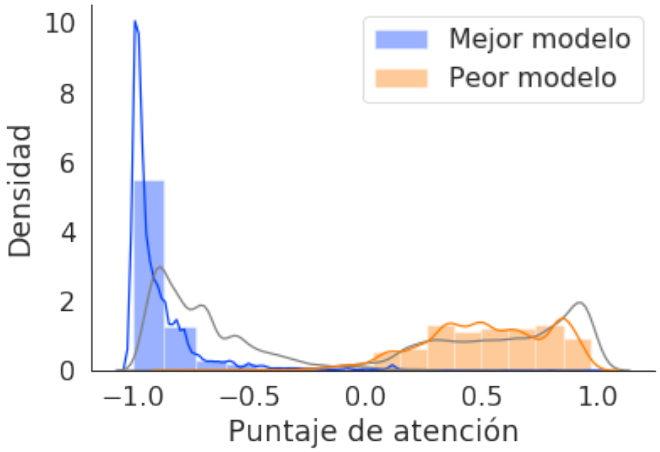


Figura 4.18: Distribución de los puntajes de atención para el mejor y peor modelo con Word Attention y activación tanh, con modelos promedio en gris.

Activación Tanh

Las distribuciones de los puntajes de atención del modelo de Word Attention obtenidos con la activación tanh, son los que se muestran en la Figura 4.18. Como podemos ver, los puntajes del mejor modelo se distribuyeron mayoritariamente en valores negativos, mientras que sucedió lo contrario con el peor modelo. Al igual que en la activación lineal, los puntajes se agrupan en el rango positivo o en el rango negativo. Una vez más observamos que, tal como en la activación sigmoide, el mejor modelo condensa los puntajes en un valor cercano a uno, mientras que el peor modelo los distribuye más uniformemente en el rango [0, 1].

Si bien esto supone una gran diferencia, a la hora de ver los puntajes asignados a las palabras en las Figuras 4.19a, 4.19b, 4.20a, y 4.20b, podemos ver que fueron asignados de forma similar. Tanto para el mejor como para el peor modelo, los puntajes con mayor valor absoluto fueron para las palabras más significativas, mientras que los de menor valor absoluto fueron para las palabras menos significativas. Podemos suponer que el hecho de que los modelos se hayan distribuido mayoritariamente en valores negativos y positivos respectivamente, no afecta

word	true	prediction	attention
i	Premise	Premise	-0.999462
we	Premise	Premise	-0.998115
you	Premise	Premise	-0.998019
do	Premise	Premise	-0.997698
my	Premise	Premise	-0.997696
not	Premise	Premise	-0.997410
they	Premise	Premise	-0.997213
me	Premise	Premise	-0.997038
important	Premise	Premise	-0.996664
think	Premise	Premise	-0.996570

(a) Palabras con mayor valor absoluto de atención

word	true	prediction	attention
cattles	Premise	Premise	0.986890
toefl	Premise	Premise	0.927075
indepedent	O	O	0.870540
polices	Premise	Premise	0.804988
snuffing	Premise	Premise	0.800522
ibt	Premise	Premise	0.799177
transportations	O	O	0.787154
adventurousness	Premise	Claim	0.780038
irak	Premise	Premise	0.739754
fatality	Claim	Premise	0.703703

(b) Palabras con menor valor absoluto de atención

Figura 4.19: Puntajes de atención utilizados por el mejor modelo con Word Attention y activación tanh.

word	true	prediction	attention
disagree	O	O	0.966429
i	Premise	Premise	0.958603
wrong	Premise	O	0.924844
however	O	O	0.919147
reason	Premise	Premise	0.917827
.	Premise	O	0.917043
reasons	O	O	0.916054
believe	O	O	0.908106
my	Premise	Premise	0.907233
me	Premise	Premise	0.907107

(a) Palabras con mayor atención

word	true	prediction	attention
cattles	Premise	Premise	-0.781241
indepedent	O	O	-0.689966
prepares	Premise	Premise	-0.645296
incoming	Premise	Premise	-0.620951
snuffing	Premise	Premise	-0.616128
organising	O	O	-0.600944
equips	O	O	-0.591688
equip	Claim	Premise	-0.520394
salesmanship	Premise	Premise	-0.515598
polices	Premise	Premise	-0.514733

(b) Palabras con menor atención

Figura 4.20: Puntajes de atención utilizados por el peor modelo con Word Attention y activación tanh.

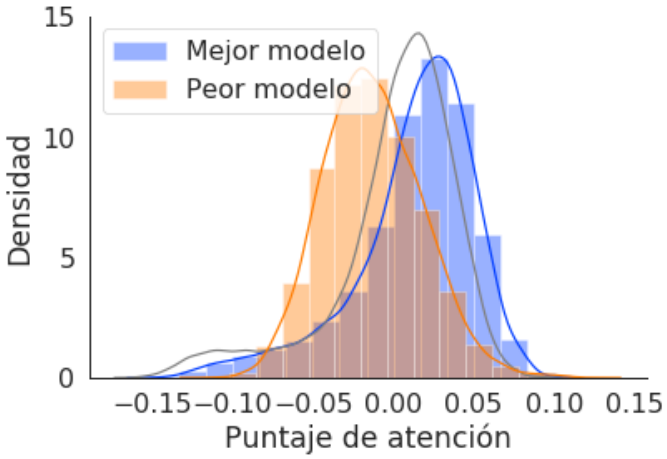


Figura 4.21: Distribución de los puntajes de atención para el mejor y peor modelo con Context Attention y activación tanh.

al momento de darle un peso a las palabras. Claramente la diferencia en el desempeño de los modelos fue debido a que, como en análisis anteriores, uno de los modelos obtuvo mayor densidad de palabras con un mayor peso, mientras que en el otro la densidad de los puntajes atención fue más uniforme, y por lo tanto no resultó tan informativa como la primera. Viendo los modelos promedio podemos ver que, por un lado, un modelo que se concentró en puntajes negativos, obtuvo peor desempeño, coincidiendo con el hecho de que la densidad de palabras con puntaje alto fue menor que el mejor modelo. Por el otro lado, un modelo que se centró en puntajes positivos, obtuvo mejor desempeño que el otro modelo, coincidiendo con el hecho de asignar un puntaje alto a una mayor cantidad de palabras.

Por último, analizaremos los modelos con Context Attention y activación tanh. Ninguno de estos modelos logró obtener un F1 mayor que 0.5. Si observamos la distribución de los puntajes de atención en la imagen 4.21, podemos notar que, si bien parecen tener una tendencia a centrarse en puntajes negativos o positivos, dichos puntajes nunca logran converger hacia alguno de los extremos, por lo que termina perjudicando el desempeño de los modelos.

4.6. Análisis de hiperparámetros

En esta sección, compararemos el desempeño de distintos clasificadores teniendo el cuenta los valores de sus hiperparámetros, con el objetivo de determinar si existen variaciones apreciables. Cada hiperparámetro impacta de una manera particular sobre el aprendedor, y los valores que mejores resultados obtienen pueden ser utilizados para analizar las características del problema y las arquitecturas necesarias para modelarlo adecuadamente.

Como se mencionó en la sección 4.4, los hiperparámetros para cada clasificador fueron seleccionados aleatoriamente, para lograr una mejor cobertura del espacio de combinaciones posibles.

Como los resultados obtenidos utilizando Context Attention no tuvieron un buen rendimiento en general, no tiene sentido analizar su comportamiento en base los distintos hiperparámetros, ya que los diversos resultados que se obtuvieron no dependieron de estos. Luego, sólo nos enfocaremos en los modelos de Word Attention y sin atención.

4.6.1. Tamaño de la capa recurrente

Como se explicó en el Capítulo 3, el tamaño de la capa recurrente debe ser lo suficientemente grande como para capturar todos los factores de variación, pero sin exceder la cantidad de parámetros entrenables con un conjunto de datos en particular. Es por esto que los valores elegidos para este hiperparámetro fueron 50, 100 y 150.

Como podemos ver, en el modelo con Word Attention los tres valores resultaron en desempeños similares, siendo 150 el tamaño de LSTM que obtuvo el mayor F1. En el modelo sin atención, el mejor F1 también se obtuvo con un tamaño de LSTM de 150, y decreció para el resto.

Las diferencias observadas son pequeñas, pero aún así son relevantes y permiten concluir que el tamaño de la capa recurrente recomendable es 150. Cabe destacar que valores más bajos parecen afectar más al

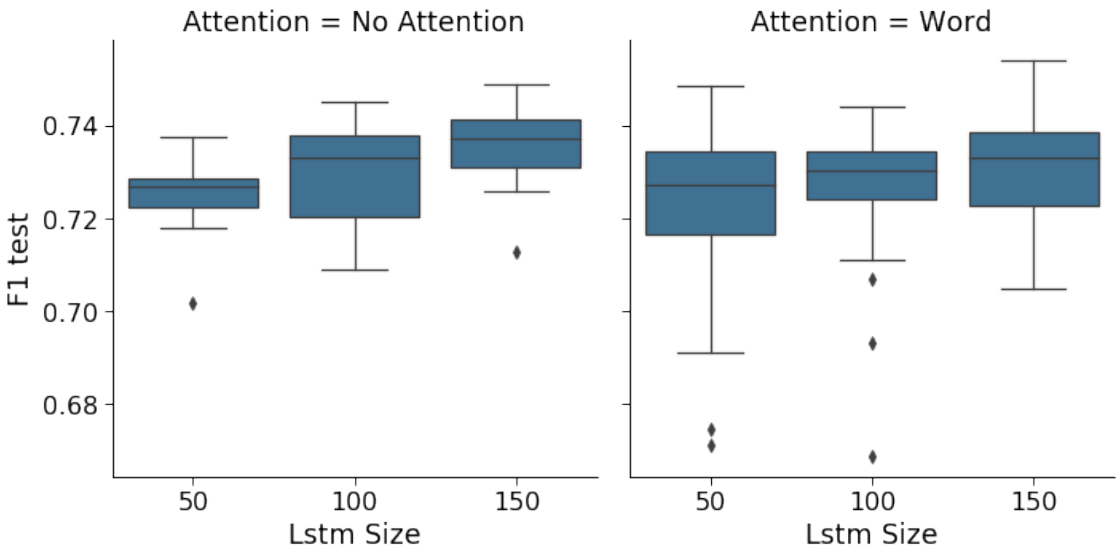


Figura 4.22: Puntajes de F1 obtenidos por modelos con distintos tamaños de la capa LSTM, con Word Attention y sin atención.

modelo sin atención que al modelo de Word Attention.

4.6.2. Dropout

El dropout es una técnica de regularización para reducir el *overfitting* en las redes neuronales previniendo que la red se coadapte a los datos de entrenamiento. La intuición detrás de este método es "apagar" aleatoriamente unidades (ocultas y visibles) de la red durante la etapa de entrenamiento, para evitar que las actualizaciones de algunas unidades se den de forma que arreglen los errores de las otras unidades (i.e., se coadapten).

Los valores usuales van desde 0.4 a 0.6, siendo 0.5 el valor más usado, ya que "apagar" una neurona con 0.5 de probabilidad obtiene la máxima varianza para la distribución, i.e., el máximo valor de regularización. Sin embargo, el valor de dropout termina siendo un hiperparámetro más para ajustar, por lo que puede tomar otros valores que no sean necesariamente los mencionados arriba.

Como podemos ver en la Figura 4.23, los valores de dropout utili-

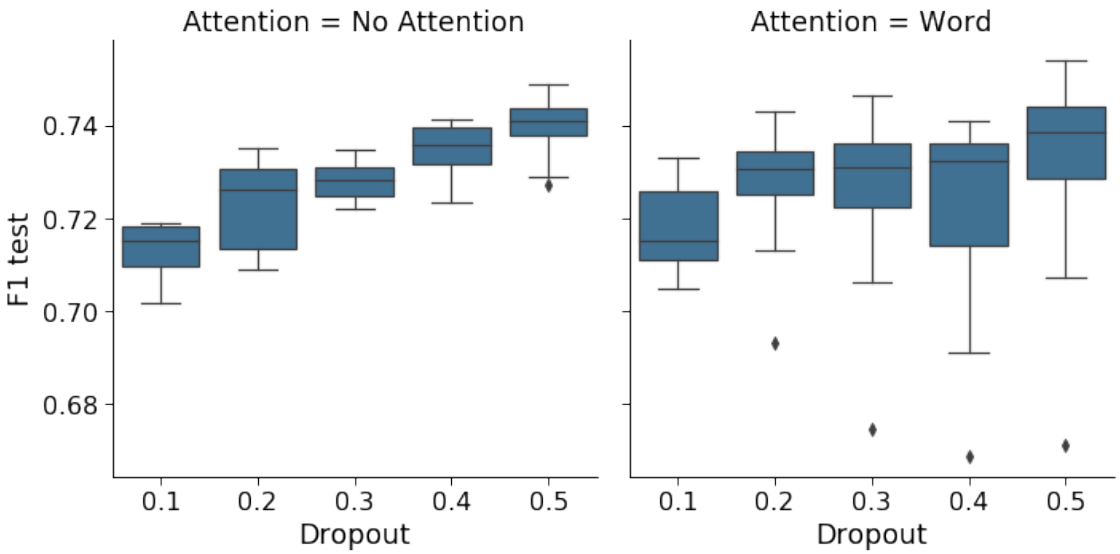


Figura 4.23: F1 por dropout, correspondientes a los modelos de Context Attention, sin atención, y Word Attention.

zados para los distintos modelos fueron de 0.1 a 0.5, siendo los valores entre 0.2 y 0.5 los que mejores resultados obtuvieron para el modelo de Word Attention. El mal desempeño obtenido por el dropout de 0.1 en este modelo, probablemente se deba a que el valor es demasiado chico, causando que no se apagan las suficientes neuronas para eliminar los "ruidos" que puedan haber en la red. Por otro lado, el modelo sin atención tuvo sus mejores resultados con un dropout de 0.5, mientras que su rendimiento decreció a medida que se disminuyó el valor del mismo. Podríamos suponer que el modelo sin atención es más sensible al dropout, viéndose beneficiado por los valores más grandes, mientras que el modelo de Word Attention parece ser menos sensible, si bien también obtuvo su mejor modelo con el mayor valor de dropout. Luego, para ambos modelos el mejor valor de dropout fue el de 0.5, i.e., el de máximo valor de regularización.

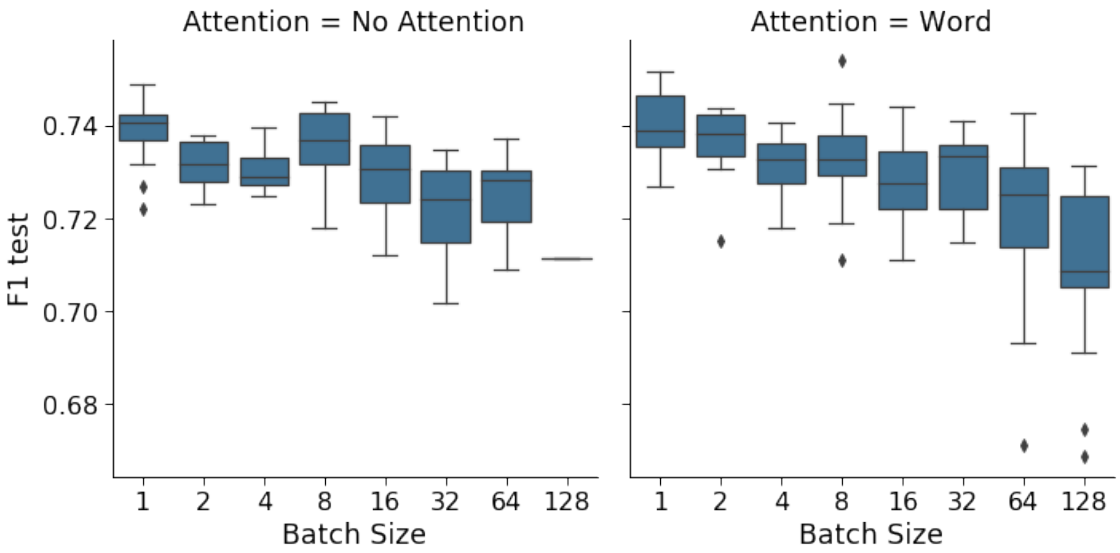


Figura 4.24: F1 obtenido por batch size, correspondientes a los modelos de Word Attention y sin atención.

4.6.3. Batch size

Como hemos mencionado anteriormente, el tamaño del lote de ejemplos recibido por la red es uno de los factores que determina la rapidez de entrenamiento. Se requiere un balance entre el tiempo que tarda cada actualización, y la cantidad de actualizaciones necesarias para que el modelo converja. En este trabajo, los valores aleatorios utilizados para este hiperparámetro fueron 1, 2, 4, 8, 16, 32, 64 y 128.

En la Figura 4.24 podemos observar que para el modelo con Word Attention los batches de menor tamaño fueron los de mejor desempeño, siendo los de mayor tamaño los que obtuvieron un menor rendimiento. Para el modelo sin atención el comportamiento fue similar al de Word Attention, aunque el batch de tamaño 1 obtuvo un notable mejor rendimiento que el resto de los tamaños, si bien requiere de una mayor cantidad de actualizaciones por época.

4.7. Análisis de error

A continuación, haremos un análisis sobre cómo se distribuyeron las distintas etiquetas de las palabras en los ensayos, y cuáles fueron los principales errores y diferencias entre los clasificadores en base a ellas. Para esto, graficamos las matrices de confusión para los mejores modelos de cada enfoque en el corpus de dev y test, presentados en la Figura 4.1 de la sección 4.5.1. Finalmente mencionaremos otras observaciones realizadas durante el entrenamiento de los modelos.

4.7.1. Matrices de confusión

Comenzaremos nuestro análisis sobre las matrices de confusión para el corpus de dev.

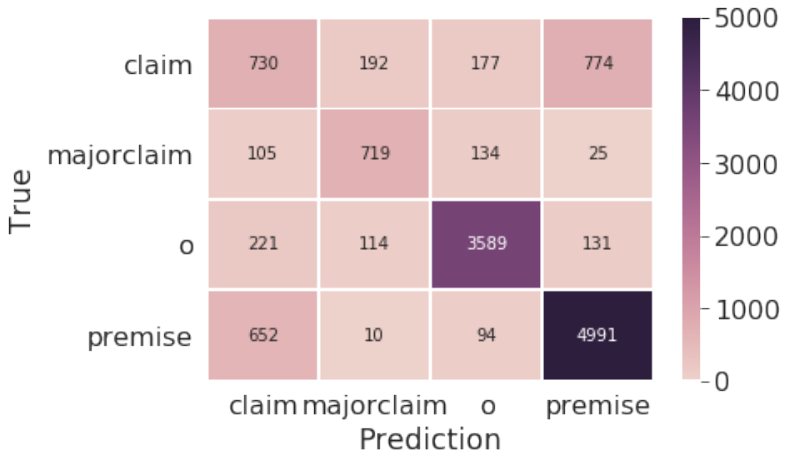


Figura 4.25: Matriz de confusión para el modelo sin atención con un F1 de 0.717 en el corpus de dev.

Como podemos ver en la Figura 4.25, el modelo sin atención tiende a confundir las etiquetas de Claim con las de Premise y viceversa, aunque esta última es en menor proporción. Más en detalle, podemos ver que el clasificador etiqueta erróneamente Claims como Premises una cantidad mayor de veces que las que etiqueta correctamente a las propias Claims. Luego, que el clasificador haya etiquetado como Pre-

mises a un 41 % de las Claims totales, nos indica que el mismo tiene una gran dificultad para reconocer Claims, confundiéndolos frecuentemente con Premises. Por otro lado, la cantidad de veces que etiqueta erróneamente Premises como Claims se da sobre el 11 % de las Premises totales. Aunque este porcentaje no resulta tan elevado como en el caso anterior, sí resulta llamativo cuando vemos que el resto de los porcentajes en las etiquetas erróneas para la clase Premise fue 0.001 % con MajorClaims, y 0.01 % para O.

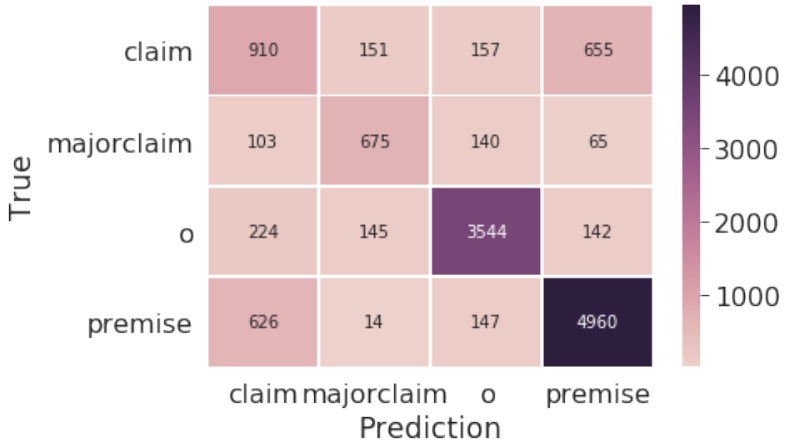


Figura 4.26: Matriz de confusión para el modelo de Word Attention con un F1 de 0.728 en el corpus de dev.

Si observamos la matriz de confusión para Word Attention en la Figura 4.26, podemos observar una mejora en la clase Claim con respecto al modelo sin atención (Figura 4.25), debido a una disminución de etiquetas erróneas de Premise. El porcentaje en este caso disminuye al 34 %, aunque sigue siendo un porcentaje alto de error. La cantidad de etiquetas erróneas de Claim para Premise se mantiene similar con una leve mejora, aunque empeora respecto a las etiquetas erróneas de MajorClaim. Sin embargo, la clase que empeoró levemente en comparación al modelo sin atención, fue la clase MajorClaim. Esto se debió a un aumento en el número de errores con la etiqueta Premise. Por lo tanto, el modelo de Word Attention comparte, en menor medida, la dificultad para diferenciar la clase Claim de la clase Premise que tiene

el modelo sin atención. Por otro lado, aumenta esta misma dificultad con respecto a la clase MajorClaim, así como también disminuye muy levemente su rendimiento respecto a la clase Premise.

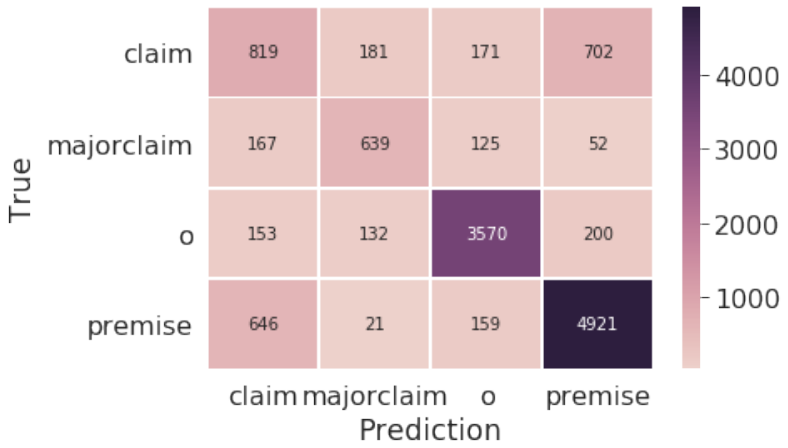


Figura 4.27: Matriz de confusión para el modelo de Context Attention con un F1 de 0.708 en el corpus de dev.

Finalmente, observando la matriz de confusión para Context Attention en la Figura 4.27, podemos observar que su comportamiento respecto a la clase Claim es mejor que la del modelo sin atención, pero peor que el de Word Attention. Así mismo, el rendimiento con la clase MajorClaim disminuyó tal como sucedió en el modelo de Word Attention, sólo que en este caso el clasificador confundió como Claims a una mayor cantidad de MajorClaims que el resto de los modelos. Así mismo disminuyó levemente su rendimiento con la clase Premise, en comparación a los otros modelos.

A continuación pasaremos al análisis de las matrices de confusión para los corpus de test, y cómo la diferencia entre ambos corpus afecta los resultados.

En la Figura 4.28 podemos observar la matriz de confusión para el modelo sin atención. Lo principal que salta a la vista es la notable mejora que obtuvo la clase Claim con respecto al corpus de dev (Figura 4.25), disminuyendo el porcentaje de error de etiquetar como Premises a los Claims del 41 % al al 32 %. En este caso, el rendimiento de la

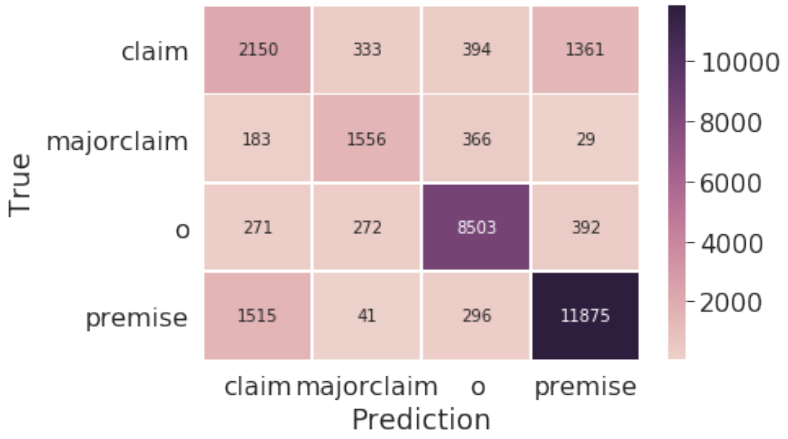


Figura 4.28: Matriz de confusión para el modelo sin atención con un F1 de 0.748 en el corpus de test.

clase pudo alcanzar al que obtuvieron los modelos con atención en el corpus de dev. Con respecto al resto de las clases, sólo se obtuvo una leve mejora en la etiqueta O, manteniendo un rendimiento similar para la clase Premise y MajorClaim.

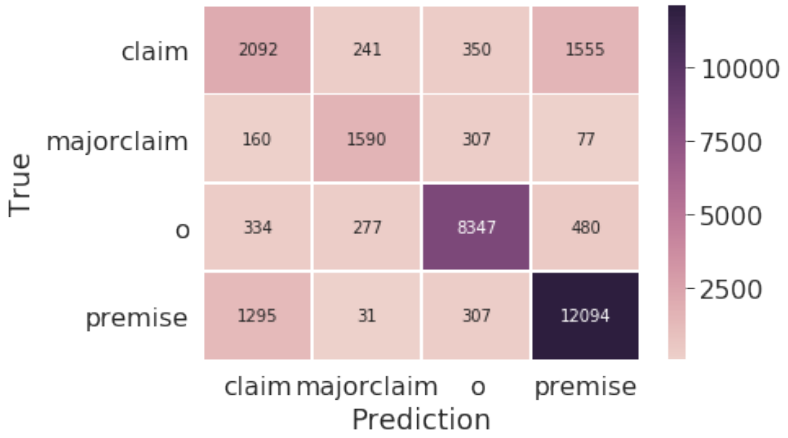


Figura 4.29: Matriz de confusión para el modelo de Word Attention con un F1 de 0.754 en el corpus de test.

La matriz de confusión para el modelo de Word Attention puede observarse en la Figura 4.29. En este caso, el modelo se mantuvo con un rendimiento similar respecto a la clase Claim, obteniendo una pe-

queña mejora. Aún así, esta mejora no fue suficiente para alcanzar el rendimiento que obtuvo el modelo sin atención. Sin embargo, la clase MajorClaim obtuvo una buena mejora, logrando en este caso superar al modelo sin atención. El resto de las clases mejoró en menor proporción, sin alejarse demasiado del rendimiento obtenido con el corpus de dev.

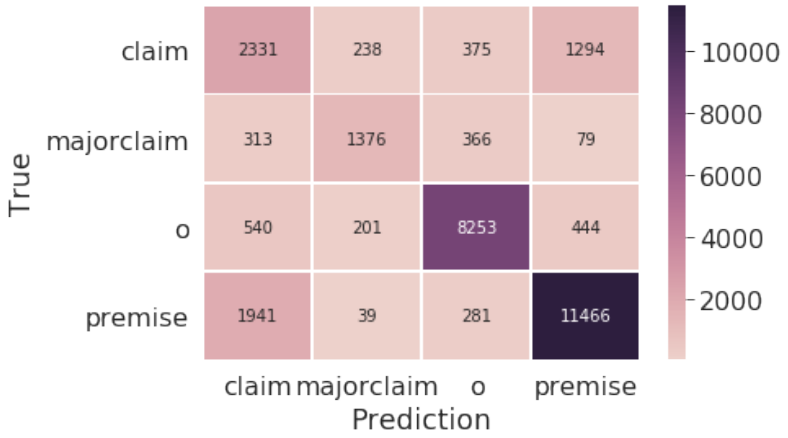


Figura 4.30: Matriz de confusión para el modelo de Context Attention con un F1 de 0.729 en el corpus de test.

En 4.30 podemos observar la matriz de confusión para el modelo de Context Attention. En este caso, el modelo obtuvo una buena mejora respecto a la clase Claim, no sólo superando su rendimiento en dev, sino que superando la de los modelos de Word Attention y sin atención. Sin embargo, el rendimiento para el resto de las clases disminuyó levemente con respecto al corpus de dev.

Como conclusión del análisis de las matrices de confusión, podemos decir que, tanto el modelo sin atención como los modelos con atención, no tienen mayores dificultades en distinguir los componentes argumentativos de la clase O, sino en clasificarlos.

Todos los clasificadores tuvieron claras dificultades con la clase Claim, aunque los modelos con atención en menor medida. Los modelos con atención tuvieron mayor dificultad con la clase MajorClaim que los modelos sin atención, aunque la diferencia disminuyó en el

corpus de test. Para la clase Premise, los resultados fueron similares para los tres modelos, aunque Word Attention tuvo mejor desempeño. Así mismo, los modelos sin atención fueron levemente mejores con respecto a la etiqueta O en ambos corpus.

Finalmente, se encontraron diferencias no triviales entre el desempeño en los corpus de dev y de test. Esto puede ser evidencia de que este tipo de arquitectura no generaliza correctamente, independientemente del tipo de atención utilizado.

4.7.2. Overfitting

Durante el entrenamiento de los distintos modelos, pudimos observar que todos los modelos terminan haciendo overfitting sobre el corpus de train. Es por esto que se estableció el parámetro de paciencia mencionado en 4.4, que interrumpe el entrenamiento si no hay mejoras de desempeño en el corpus de dev. Sin embargo, esto no soluciona el problema completamente, ya que la red termina haciendo overfitting sobre el corpus de train de todas formas, aunque con la diferencia que lo hace en un menor porcentaje. Este overfitting podría deberse a que el clasificador no tiene información suficiente como para generalizar a ejemplos no vistos anteriormente.

Capítulo 5

Conclusiones y trabajo futuro

En este capítulo vamos a analizar las conclusiones que obtuvimos respecto a las diferentes hipótesis planteadas en el capítulo 4, así como también se listarán las distintas posibilidades sobre trabajo futuro que fueron surgiendo en base al nuestro.

5.1. Conclusiones

En este trabajo se simplificó una arquitectura del estado del arte para agregarle un mecanismo de atención, pudiendo diferenciar el rendimiento de la misma en la tarea de minería de argumentación, ya sea con o sin dicho mecanismo. Teniendo en cuenta nuestras hipótesis y los resultados obtenidos en los experimentos, enunciaremos nuestras conclusiones.

5.1.1. Primera hipótesis

Nuestra primera hipótesis planteaba que agregar un mecanismo de atención mejoraría el desempeño de la red. A partir de los experimentos realizados, se llegó a la conclusión de que Word Attention efectivamente se focaliza sobre palabras determinantes, mientras que las de poca relevancia son ignoradas. Esto facilita la identificación de los argumentos y ayuda al modelo a converger a un mejor clasificador.

Por otro lado, al contrario de lo que intuimos en el capítulo de arquitectura, los modelos que utilizan Context Attention lograron obtener resultados similares a los modelos sin atención en algunos casos, pero en otros nunca convergieron. Si bien dichos resultados indicarían que este método no es recomendable para esta tarea, cabe aclarar que el tamaño del contexto utilizado en este trabajo, es decir, todo el ensayo, puede haber sido la razón por la que el rendimiento no fue bueno. Nuestra explicación es que posiblemente la red no logró captar la noción de contexto a través de los distintos features de cada timestep, debido a que este contexto puede ser demasiado grande como para que la red pueda caracterizar correctamente una palabra dentro de él.

Esto sugiere un mejor análisis sobre los distintos tamaños del contexto a utilizar, con el objetivo de poder dar una conclusión más precisa sobre el rendimiento general de este enfoque de atención.

Así mismo, este mecanismo ayuda a la red a generalizar en una tarea que no sólo es semánticamente compleja, sino donde generalmente se cuenta con pocos ejemplos de entrenamiento. Con esto se demostró que nuestra suposición de que el mecanismo de atención podría ser redundante para la red debido a que el comportamiento "intuitivo" del mismo era parecido al de la LSTM, era errónea.

5.1.2. Segunda hipótesis

La segunda hipótesis planteada hacía referencia al impacto de las distintas funciones de activación sobre el mecanismo de atención. Se compararon las funciones propuestas, tanto en el desempeño de la red como en las distribuciones de los puntajes de atención. Se concluyó que las funciones de activación tienen un alto impacto en el desempeño de la atención (y consecuentemente en el de la red), y que las mejores funciones de activación para el mecanismo fueron aquellas que lograron polarizar los valores de la atención, como la función lineal, ReLU y sigmoide; mientras que fueron las funciones tanh y softmax las que menos pudieron lograrlo, obteniendo un rendimiento inferior. En con-

traste con nuestra hipótesis inicial, la función lineal, que teóricamente no agrega expresividad a la red, obtuvo una mejor distribución de resultados que los modelos sin atención. Por otro lado, la función tanh, que es muy similar a la función sigmoide, obtuvo los peores resultados (dentro de los modelos con Word Attention). Por lo tanto, al contrario de lo que supusimos, el desempeño de la función de activación no depende de la linearidad o no-linearidad de la misma.

Más allá de los resultados que indican que la función de activación es relevante para el modelo, se desconoce cuáles son las propiedades teóricas que hacen que una función sea más adecuada que otra. Para explicar los resultados empíricos obtenidos, es necesario un desarrollo más profundo sobre las características de cada modelo.

5.1.3. Hiperparámetros

Adicionalmente, se evaluó el impacto de los hiperparámetros sobre la red y cómo terminaron afectando al desempeño de la misma, con la conclusión de que los hiperparámetros son relevantes sólo para los modelos con Word Attention o sin atención. Con respecto a Context Attention, observamos que hubieron muy pocas combinaciones de funciones de activación e hiperparámetros que lograron que este enfoque obtenga mejor rendimiento que la red sin atención. Esto evidenció que es el enfoque en sí el que no logra aportar a la red, y no una cuestión de elección de funciones de activación e hiperparámetros. Sin embargo, como mencionamos anteriormente, queda por investigar si el método con un contexto más reducido puede generar mejores representaciones para las palabras.

5.1.4. Resumen

En resumen, nuestra conclusión es que utilizar atención mejorará el desempeño de los clasificadores neuronales en la tarea de minería de argumentaciones, dependiendo de un conjunto de decisiones previas. Éstas abarcan desde el enfoque del mecanismo de atención utilizado,

hasta la elección de las funciones de activación dentro del mismo. La elección de los hiperparámetros de la red también tendrá un impacto sobre el rendimiento, aunque será menos relevante que el causado por el uso de algún mecanismo de atención.

5.2. Trabajo futuro

Teniendo en cuenta los resultados obtenidos en este trabajo, quedan abiertas distintas alternativas para seguir profundizando sobre esta investigación. Trataremos dichas alternativas bajo los conceptos de mejora del modelo y análisis de resultados, y mejoras de dataset y proceso de anotación. El primero hace referencia a las posibles modificaciones que podrían aplicarse sobre el modelo con atención para seguir mejorando sobre la tarea de minería de argumentaciones, así como también la forma en la que podrían hacerse análisis más complejos de los resultados obtenidos y obtener conclusiones más robustas. El segundo concepto trata sobre qué cambios podrían hacerse sobre el dataset, así como también las diferentes alternativas que podrían aplicarse en el proceso de anotación, y qué nuevos resultados podríamos esperar con ellos.

5.2.1. Mejoras del modelo y del análisis de resultados

La principal idea que nos surge luego de los resultados obtenidos, es la de analizar al enfoque de Context Attention utilizando distintos tamaños de contexto. Como se detalló en la sección 4.4, el contexto utilizado en nuestro trabajo fue el ensayo completo, lo que debido al largo del mismo puede haber sido causa del mal desempeño de los modelos. Luego, como se mencionó anteriormente en la sección 5.1, es necesario un mejor estudio de este enfoque variando el largo del contexto que recibe el mismo. Los distintos contextos que podrían utilizarse, yendo de mayor a menor, son a nivel párrafo, a nivel oración, e incluso utilizando ventanas de palabras de largo fijo o, en base a algún

criterio, de largo variable. Así mismo, también podría mantenerse el contexto a nivel ensayo, y observar el comportamiento del mecanismo de atención ante ensayos más largos que los presentes en nuestro dataset. Luego, de ser el largo de los ensayos lo que realmente afectó a nuestros modelos de Context Attention, deberíamos obtener mejores resultados para los contextos menores, y peores resultados para los mayores.

Otra idea que nos surge es la de usar conjuntamente ambos enfoques de atención, y analizar si éstos logran complementarse. Como mencionamos en la sección 3.2, cuando utilizamos el enfoque de Word Attention, todas las palabras obtienen un puntaje de atención único. Esto significa que, por ejemplo, la palabra "We" obtendrá el mismo puntaje de atención en todos los contextos en los que aparezca, sea indicativa de un componente argumentativo o no. Luego, indirectamente estamos introduciendo el sesgo de que las palabras siempre son igual de importantes, lo que claramente perjudicará al clasificador en los diferentes contextos, disminuyendo su rendimiento. Sin embargo, nuestra intuición sobre el comportamiento de Context Attention nos decía que este enfoque debería ser capaz diferenciar la importancia de las palabras bajo los distintos contextos. Si nuestra suposición de que el bajo rendimiento de Context Attention fue debido al largo de los ensayos es cierta, y llegaran a obtenerse mejores resultados con los análisis planteados en el párrafo anterior, podríamos suponer que nuestra intuición es cierta, y que los enfoques con Context Attention dan un aporte a la red, al igual que Word Attention. Luego, si ambos enfoques logran complementarse disminuyendo el sesgo mencionado anteriormente, tendremos la robustez observada en los modelos de Word Attention sumado al concepto de contexto introducido por el enfoque de Context Attention.

Otra alternativa a nuestro trabajo es la de aplicar el mecanismo de atención luego de BiLSTM, en vez de aplicarla antes que ésta. Si bien no es trivial obtener una buena intuición de como puede funcionar esto, en base a trabajos previos, como el de Koreeda et al. [2016],

podemos suponer que también pueden obtenerse buenos resultados de esta forma. Aún así, debería hacerse un análisis más detenido de cuáles son los posibles beneficios de aplicar la atención de esta forma, y cuales son las razones detrás de los diferentes resultados que se obtengan.

Otro importante aspecto a evaluar es el comportamiento del mecanismo de atención en la red original, siendo utilizada junto a la capa de CNN y CRF. De esta forma podría evaluarse si el mecanismo de atención sigue aportando a la red o si el mismo pierde relevancia dentro de una arquitectura más compleja, así como también si la afecta negativamente.

También, inspirados por modelos más recientes, como *transformers* o modelos de lenguaje no recurrentes como Bert [Devlin et al., 2018], se pueden analizar distintas formas de aplicar atención y compararlas con la utilizada en este trabajo. Una posibilidad, sería aplicar la capa de atención varias veces para capturar distintos niveles semánticos, como por ejemplo, aplicar atención a nivel palabra, luego a nivel oración, y finalmente a nivel párrafo. Otra opción sería aplicar atención dentro de la capa oculta de la LSTM (*inner attention*), tomando la idea de Wang et al. [2016], o utilizar atención conjunta sobre el texto y sobre el título de los ensayos, tomando una idea similar a la de Koreeda et al. [2016].

Así mismo, nuestra arquitectura también ofrece posibilidades de mayor estudio. Un aspecto sería el análisis más exhaustivo de cada posible función de activación, obteniendo conclusiones más generalizables sobre el comportamiento de cada modelo. Ya que las mismas tienen alto impacto en la red, merecen un mayor detenimiento a la hora de analizarlas y elegirlas, logrando entender cuál es la verdadera causa de por qué afecta a la red de la forma en que lo hace.

De la misma manera, en este trabajo sólo hemos realizado una prueba de concepto que arroja resultados prometedores. Un análisis más completo, seleccionando manualmente la mejor combinación de hiperparámetros junto con un proceso de optimización más eficiente, podrían producir tanto mejores resultados como una perspectiva

más clara sobre los mecanismos de atención vistos. Por ejemplo, una opción sería utilizar *cross-validation* para encontrar cuáles son los hiperparámetros más adecuados para los distintos tipos de atención que se evaluaron. Adicionalmente, se deberían realizar test de hipótesis para estimar el nivel de significancia de las distintas distribuciones de resultados obtenidas.

Otro aspecto a analizar es si el mecanismo de atención efectivamente ayuda a los modelos a converger más rápido que los modelos sin atención. Para esto, deberíamos comparar la distribución de los tiempos de convergencia de los modelos, utilizando los mismos valores de batch size y tamaño de LSTM.

5.2.2. Mejoras en el dataset y el proceso de anotación

Una posible alternativa al dataset de ensayos persuasivos utilizado en nuestro trabajo, es la de utilizar datasets con distintos dominios argumentativos, como por ejemplo, un dataset con dominio de textos legales. Con esto podremos evaluar cómo la red responde ante distintos tipos de argumentos, y en base a los resultados que se obtengan, analizar si el mecanismo de atención podría tener una mayor dificultad para aportar al rendimiento de la red en los distintos dominios argumentativos. Luego se podría determinar si el buen funcionamiento del mecanismo de atención es independiente del dominio argumentativo o no.

Por otro lado, el uso de datasets considerablemente más grandes que el nuestro podría evidenciar si el mecanismo de atención sigue siendo un aporte relevante al modelo, o si pierde relevancia. Esto se evidenciaría por un comportamiento similar a una LSTM sin atención, concordando con la idea que planteamos anteriormente en la sección 3.1.3. Así mismo, también podrían utilizarse datasets menores al nuestro, para analizar cuál es el límite en el que la atención deja de aportar una mejora en la generalización del modelo ante la escasez de datos.

Otra posibilidad, es la de unificar las etiquetas de Claim y Major-

Claim, tratando a todas sus palabras como Claim. Esto puede tener sentido, ya que un MajorClaim es un Claim en sí. Luego podríamos analizar si los clasificadores mejoran respecto a esta clase, que tal como vimos en la sección 4.7.1 fue la de peor rendimiento. Por otro lado podríamos analizar si con esto Context Attention obtiene una mejora respecto a Word Attention y la red sin atención, ya que sus modelos obtuvieron una mayor dificultad con la clase MajorClaim, y confunden estas etiquetas entre sí en mayor porcentaje.

Siguiendo lo mencionado en la sección 1.3 con respecto a cómo los diferentes esquemas de etiquetado y las discrepancias que se generan entre los anotadores podrían introducir un bias a la red, podría analizarse el impacto por el uso de distintos esquemas de etiquetado y distintos anotadores sobre nuestro dataset. Con esto se podría evaluar si efectivamente los distintos esquemas de anotación ocultan o no al modelo las causas latentes de la argumentación, así como también podría evaluarse si las discrepancias de los anotadores hacen que los modelos se sobreajusten más a un estilo particular de corpus-anotación o no. Más aún, que se confirmen dichas suposiciones implicaría que gran parte de los resultados presentados en los distintos trabajos de investigación sobre la minería de argumentos contienen un bias, indicando que los sistemas propuestos podrían no ser tan buenos como se los presentan.

Finalmente, como se mencionó en la sección 1.4, los puntajes asignados por el mecanismo de atención podrían utilizarse para ayudar en el proceso de etiquetado manual, por ejemplo, resaltando al anotador las palabras con alto puntaje de atención. Incluso el mecanismo de atención podría sugerir correcciones en los esquemas de anotación, por ejemplo, si llegaran a haber contradicciones o reglas vagas en el mismo. Luego, podría evaluarse el impacto producido por estas mejoras en el proceso de etiquetado sobre el desempeño de la red, y analizar si con esto se logra una reducción en el bias mencionado en el párrafo anterior.

Bibliografía

T A Govier. A practical study of argument. 01 2010.

Yamen Ajjour, Wei-Fan Chen, Johannes Kiesel, Henning Wachsmuth, and Benno Stein. Unit segmentation of argumentative texts. In Proceedings of the 4th Workshop on Argument Mining, pages 118–128, Copenhagen, Denmark, September 2017. Association for Computational Linguistics. doi: 10.18653/v1/W17-5115. URL https://www.aclweb.org/anthology/W17-5115.

Khalid Al-Khatib, Henning Wachsmuth, Matthias Hagen, Jonas Köhler, and Benno Stein. Cross-domain mining of argumentative text through distant supervision. In Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pages 1395–1404, San Diego, California, June 2016. Association for Computational Linguistics. doi: 10.18653/v1/N16-1165. URL https://www.aclweb.org/anthology/N16-1165.

Daniel Andor, Christopher Alberti, David Weiss, Aliaksei Severyn, Alessandro Presta, Kuzman Ganchev, Slav Petrov, and Michael Collins. Globally normalized transition-based neural networks. pages 2442–2452, 03 2016. doi: 10.18653/v1/P16-1231.

Dzmitry Bahdanau, Kyunghyun Cho, and Y Bengio. Neural machine translation by jointly learning to align and translate. ArXiv, 1409, 09 2014.

Philippe Besnard, Alejandro Garcia, Anthony Hunter, Sanjay Modgil, Henry Prakken, Guillermo Simari, and Francesca Toni. Introduction to structured argumentation. Argument & Computation, 5, 02 2014. doi: 10.1080/19462166.2013.869764.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805, 2018.

Steffen Eger, Johannes Daxenberger, and Iryna Gurevych. Neural end-to-end learning for computational argumentation mining. CoRR, abs/1704.06104, 2017. URL http://arxiv.org/abs/1704.06104.

Juyeon Kang and Patrick Saint-Dizier. A discourse grammar for processing arguments in context. In COMMA, volume 266 of Frontiers in Artificial Intelligence and Applications, pages 43–50. IOS Press, 2014.

Yuta Koreeda, Toshihiko Yanase, Kohsuke Yanai, Misa Sato, and Yoshiki Niwa. Neural attention model for classification of sentences that support promoting/suppressing relationship. In Proceedings of the Third Workshop on Argument Mining (ArgMining2016), pages 76–81, Berlin, Germany, August 2016. Association for Computational Linguistics. doi: 10.18653/v1/W16-2809. URL https://www.aclweb.org/anthology/W16-2809.

Xuezhe Ma and Eduard Hovy. End-to-end sequence labeling via bi-directional LSTM-CNNs-CRF. In Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 1064–1074, Berlin, Germany, August 2016. Association for Computational Linguistics. doi: 10.18653/v1/P16-1101. URL https://www.aclweb.org/anthology/P16-1101.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. Distributed representations of words and phrases and their compositionality. In Proceedings of the 26th International

Conference on Neural Information Processing Systems - Volume 2, NIPS'13, pages 3111–3119, USA, 2013. Curran Associates Inc. URL http://dl.acm.org/citation.cfm?id=2999792.2999959.

Huy Nguyen and Diane Litman. Extracting argument and domain words for identifying argument components in texts. In Proceedings of the 2nd Workshop on Argumentation Mining, pages 22–28, Denver, CO, June 2015. Association for Computational Linguistics. doi: 10.3115/v1/W15-0503. URL https://www.aclweb.org/anthology/W15-0503.

Huy Ngoc Nguyen and Diane J. Litman. Context-aware argumentative relation mining. In ACL, 2016.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. Glove: Global vectors for word representation. In Empirical Methods in Natural Language Processing (EMNLP), pages 1532–1543, 2014. URL http://www.aclweb.org/anthology/D14-1162.

Isaac Persing and Vincent Ng. Modeling argument strength in student essays. 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 543–552, Beijing, China, July 2015. Association for Computational Linguistics. doi: 10.3115/v1/P15-1053. URL https://www.aclweb.org/anthology/P15-1053.

Isaac Persing and Vincent Ng. End-to-end argumentation mining in student essays. pages 1384–1394, 01 2016. doi: 10.18653/v1/N16-1164.

Nils Reimers and Iryna Gurevych. Reporting Score Distributions Makes a Difference: Performance Study of LSTM-networks for Sequence Tagging. In Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 338–

348, Copenhagen, Denmark, 09 2017. URL http://aclweb.org/anthology/D17-1035.

Tim Rocktäschel, Edward Grefenstette, Karl Moritz Hermann, Tomáš Kočiský, and Phil Blunsom. Reasoning about entailment with neural attention. 09 2015.

Christian Stab and Iryna Gurevych. Annotating argument components and relations in persuasive essays. In COLING, pages 1501–1510. ACL, 2014a.

Christian Stab and Iryna Gurevych. Annotating argument components and relations in persuasive essays. 08 2014b.

Christian Stab and Iryna Gurevych. Parsing argumentation structures in persuasive essays. Comput. Linguist., 43(3):619–659, September 2017. ISSN 0891-2017. doi: 10.1162/COLI_a_00295. URL https://doi.org/10.1162/COLI_a_00295.

Christian Stab, Tristan Miller, and Iryna Gurevych. Cross-topic argument mining from heterogeneous sources using attention-based neural networks. CoRR, abs/1802.05758, 2018a. URL http://arxiv.org/abs/1802.05758.

Christian Stab, Tristan Miller, Benjamin Schiller, Pranav Rai, and Iryna Gurevych. Cross-topic argument mining from heterogeneous sources. pages 3664–3674, 01 2018b. doi: 10.18653/v1/D18-1402.

Sainbayar Sukhbaatar, Arthur Szlam, Jason Weston, and Rob Fergus. End-to-end memory networks. Arxiv, pages 1–11, 03 2015.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. Sequence to sequence learning with neural networks. Advances in Neural Information Processing Systems, 4, 09 2014.

Ola Svenson. Process descriptions of decision making. Organizational Behavior and Human Performance, 23:86–112, 02 1979. doi: 10.1016/0030-5073(79)90048-5.

Stephen E. Toulmin. The Uses of Argument. Cambridge University Press, 2003.

F.H. van Eemeren, R. Grootendorst, F.S. Henkemans, E.C. Krabbe, J.A. Blair, R.H. Johnson, C. Plantin, and C.A. Willard. Fundamentals of Argumentation Theory: A Handbook of Historical Backgrounds and Contemporary Developments. L. Erlbaum, 1996. ISBN 9780805818611. URL https://books.google.cz/books?id=bEBqvAEACAAJ.

Douglas Walton. Argumentation theory: A very short introduction. In Argumentation in Artificial Intelligence, pages 1–22. Springer, 2009.

Bingning Wang, Kang Liu, and Jun Zhao. Inner attention based recurrent neural networks for answer selection. In Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 1288–1297. Association for Computational Linguistics, 2016. doi: 10.18653/v1/P16-1122. URL http://aclweb.org/anthology/P16-1122.

Ursula Wingate. 'argument!' helping students understand what essay writing is about. Journal of English for Academic Purposes, 11: 145–154, 06 2012. doi: 10.1016/j.jeap.2011.11.001.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, ukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, and Jeffrey Dean. Google's neural machine translation system: Bridging the gap between human and machine translation. 09 2016.

Adam Wyner, Raquel Mochales, Marie-Francine Moens, and David Milward. Approaches to text mining arguments from legal cases. pages 60–79, 01 2010. doi: 10.1007/978-3-642-12837-0_4.

Fan Zhang, Rebecca Hwa, Diane Litman, and Homa B. Hashemi. Argrewrite: A web-based revision assistant for argumentative writings. pages 37–41, 01 2016. doi: 10.18653/v1/N16-3008.