Trabajo Especial de la Licenciatura en Ciencias de la Computación

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

Título

Detección de Diferentes Aspectos de Discurso de Odio en Redes Sociales

Autor

Lautaro Martinez

Directora: Laura Alonso Alemany

Resumen

En los últimos años las redes sociales fueron un punto de intersección entre usuarios y la libre expresión masiva. Esto con sus ventajas, presentó un problema para ciertas comunidades de la sociedad que se vieron avasalladas por comentarios o discusiones, las cuales denominamos discurso de odio, que hacen peligrar la calidad de sus vidas.

El discurso de odio se presenta como una gran problemática en la sociedad de nuestro siglo que tiene la capacidad de presentar ideas erróneas o racistas al resto de las personas influyendo como estas perciben las comunidades más vulnerables.

En los últimos años, la generación automática de contranarrativas fue una técnica de moderación de gran interés en la comunidad para evitar los problemas conlleva la censura, fuertemente utilizada por las redes en la actualidad y muy difícil de automatizar sin caer en problemas como la sobre censura.

En este trabajo, buscamos detectar de forma automática las diferentes y diversas unidades argumentativas que se presentan en el discurso de odio, empleando arquitecturas de aprendizaje profundo, con el objetivo de que sirvan de utilidad para las tareas de generación automática de contranarrativas. Además, realizamos un análisis a nivel cualitativo sobre estas arquitecturas y modelos lineales, y de lenguaje, siendo estos últimos de gran interés en los últimos años por llegar al estado del arte en diversas tareas del aprendizaje automático.

Abstract

In recent years, social networks have been a point of intersection between users and mass free expression. This, with its advantages, presented a problem for certain communities of society that were overwhelmed by comments or discussions, we call this hate speech, which endanger the quality of their lives.

Hate speech is presented as a major problem in the society of our century that has the ability to present erroneous or racist ideas to the rest of the people, influencing how they perceive the most vulnerable communities.

In recent years, the automatic generation of counter-narratives was a moderation technique of great interest in the community to avoid problems associated with censorship, heavily used by networks today and very difficult to automate without falling into problems such as over-censorship.

In this work, we seek to automatically detect the different and diverse argumentative units that are present in hate speech, using deep learning architectures, with the aim of making them useful for the automatic generation of counter-narratives. In addition, we carry out a qualitative analysis of these architectures, and linear and language models, the latter being of great interest in recent years for reaching the state of the art in various machine learning tasks.

Agradecimientos

En primera instancia, y para saldar cuentas, agradezco a mi gran amiga Mildre Cepeda que me acompañó y alentó no solo en el proceso de este trabajo, sino también en los años de cursada de la carrera.

Un enorme agradecimiento a mi directora Laura Alonso Alemany, quien a lo largo de este proceso tuvo la dedicación, paciencia y esfuerzo para guiarme en este trabajo y dejarme enormes enseñanzas que se escapan simplemente de esta tesis de licenciatura. Además de Damian Furman, quien colaboró activamente para el intercambio de ideas y experimentos a realizar.

También agradezco a mi familia y amigos que me acompañaron en estos años con constante apoyo y cariño alentándome en este camino.

Contenidos

Introducción

El desarrollo tecnológico a escala mundial cambia día a día la realidad de los individuos y comunidades reduciendo costos y en busca de una mejor calidad de vida. Producto de esto vivimos una globalización donde todas las distancias se han reducido enormemente, lo que ha provocado choques entre comunidades y realidades.

Las redes sociales son un punto clave de esto, con la posibilidad de comunicación con cualquier persona en el mundo, uno se encuentra avasallado con la cantidad de interacciones que puede tener, algo para lo que los seres humanos no están bien preparados.

Pero estas interacciones no siempre son positivas. Hemos notado a lo largo de los años como un gran grupo de personas han tomado lugar en las redes para expresar más que solo sus comunicados o ideas, han realizado denuncias y quejas con y sin fundamentos que agravan sus realidades. Esto es parte también de la globalización, la posibilidad de ver más allá de la ciudad que uno vive.

Como sociedad hemos luchado por la libertad de expresión, que en muchos casos por motivos legales han generado grandes problemas para las redes sociales mismas, cuando las publicaciones de los usuarios sobrepasan dicha libertad abusando de insultos, amenazas y racismo, algo que denominamos *discurso de odio*. A raíz de ello, la censura ha sido un camino sencillo por el cual se optó por muchos años.

Esta técnica ha probado tener muchos problemas a lo largo de los años, principalmente su carácter de limitar la expresión de las personas ha generado controversias en las mismas redes que la aplican. Además, censurar es una práctica tardía de no tratarse de un simple censurador de palabras ofensivas, lo que genera que el discurso de odio si bien limitado en su tiempo de vida llega a impactar a individuos o colectivos de forma muy negativa.

Otro punto a considerar, es el crecimiento de la comunicación, hablamos de volúmenes por unidad de tiempo que jamás se hubieran imaginado hace 100 años, que impacta aún más en la utilización de la censura *inteligente* que añadido el surgimiento de bots trolls que automatizan el discurso de odio se vuelve una práctica casi imposible de aplicar sin automatizar, lo que corre el riesgo de sobre censurar que podría concluir en problemas legales masivos por infringir una libertad básica.

Es por ello que se están planteando otros caminos a la censura, la cual no minimiza el problema que acarrea el discurso de odio sino que lo agrava para todas las partes involucradas.

Así es como las contranarrativas están siendo una nueva alternativa para que estos excesos sean apaciguados. Con ellas se deja en evidencia el discurso de odio y permanece a lo largo del tiempo para que las demás personas aprendan nuevas realidades.

Lo interesante de las contranarrativas es que se trata de una práctica de moderación, cuando la censura se trata de una práctica de ocultamiento.

Sin embargo, las contranarrativas tienen sus propios problemas, no es una técnica perfecta. El más claro de ellos es la redacción misma, algo que toma tiempo y dinero que muchas redes sociales no desean invertir. Además, la demora que tiene redactar la contranarrativa presenta los mismos que aquellos debido a censurar de forma tardía, el mensaje fue distribuido por usuarios que al momento de contarse con la contranarrativa fueron expuestos al discurso de odio.

Es así como se plantea, al igual que ocurrió con la censura, la posibilidad de automatizar esta tarea, área, joven y con mucho potencial, denominada en el procesamiento del lenguaje natural (PLN) como *Generación de contranarrativas*.

Es muy difícil automatizar la generación de contranarrativas adecuadas, porque requiere una comprensión del tópico que está mucho más allá de las capacidad de los generadores de lenguajes, que son muy superficiales.

Quizá un camino alternativo a la generación misma, que tiene un foco especial dentro del área, es la extracción de componentes argumentativos que ayuden y guíen a los redactores, ya sean personas o máquinas. Es por ello que en el presente trabajo exploramos diferentes alternativas para el análisis de la estructura argumentativa, y comparamos con otras aproximaciones existentes.

Las unidades argumentativas son partes claves que componen y dan razón a un texto argumentativo como son las justificaciones, conclusiones o el objetivo, individual o colectivo, al cual se ataca o reprime. Contar con la información de que sección de un texto conforma cada unidad, ayudaría al redactor a aclarar el panorama de lo que se está tratando de persuadir a los lectores. Función clave de los textos argumentativos.

Nuestro **objetivo** es **obtener modelos de alto desempeño para la extracción y clasificación de estas unidades argumentativas**. Queremos lograrlo manteniendo un coste computacional justo acorde a los beneficios que puede otorgar dicho coste mayor al modelo final. De esta forma, estos modelos podrán ser utilizados en tareas *downstream* para la generación de contranarrativas, o quizá para otro área de interés.

En busca de ello, este trabajo se estructura como sigue: en el siguiente capítulo ahondaremos por trabajos relacionados llevados a cabo por diferentes investigadores en el área de *Generación de contranarrativas* como son la creación de los corpus, arquitecturas utilizadas y la generación misma. Seguido de ello, una sección donde describimos a detalle la arquitectura propuesta para la extracción y clasificación de unidades argumentativas. Le sigue una descripción del dataset utilizado junto a ciertas transformaciones que llevamos a cabo para compatibilizar con la arquitectura previa. Luego, la descripción de los diferentes experimentos llevados a cabo seguido de las métricas utilizadas para evaluar el desempeño de los modelos para tomar las decisiones a seguir. Para así concluir con un análisis detallado de los experimentos además de un análisis de error para conocer las dificultades de los modelos seguido de las conclusiones generales de los resultados obtenidos y trabajo propuesto a futuro de interés para aquellas personas con afán de seguir el camino de esta propuesta.

Trabajo Relacionado

En este capítulo abordaremos diferentes aproximaciones al análisis automático de elementos argumentativos en los textos. Para conocer cómo se han llevado a cabo, sus motivaciones y sobre todo la evolución que este área ha sufrido a lo largo de los años acompañando del auge de las redes sociales.

Estructuramos este capítulo de la siguiente manera, una primera sección de trabajos sobre generación de corpus junto a sus anotaciones manuales seguido de un análisis realizado por investigadores de las tareas en común que suele tener esta tarea. Se sigue luego con una sección sobre arquitecturas presentes en el área, sobre todo de aprendizaje profundo, junto a trabajos propios para la predicción de unidades argumentativas concluyendo con trabajos relacionados a la obtención y generación automática de contranarrativas.

Finalizamos este capítulo con conclusiones obtenidas luego del análisis de estos trabajos, que nos permiten dar un primer paso relacionado a la arquitectura a utilizar.

Datasets

Comenzando por lo que fue la base del dataset utilizado en este trabajo que detallaremos en próximas secciones, (Basile et al., 2019) presenta la organización y trabajo para la creación de un dataset anotado de tweets sobre inmigrantes y mujeres para la detección automática de Discurso de odio, además de los resultados obtenidos por diferentes grupos participantes sobre dicha detección automática empleando desde técnicas tradicionales de Machine Learning hasta modelos de Deep Learning.

La creación del dataset se dividió en 3 etapas: La obtención de tweets publicados desde julio hasta septiembre de 2018, una clasificación binaria sobre presencia de discurso de odio para su posterior identificación de propiedades como su comunidad objetivo y agresividad.

Otro dataset basados en discurso de odio para su posterior anotación viene de la mano del trabajo de (Cotik et al., 2020) el cual busca entender la relación entre el discurso de odio en las redes y los artículos periodísticos más grandes de la Argentina, durante la época de la pandemia por COVID-19 en el año 2020.

Las etapas para la construcción del dataset involucraron:

1. Obtención de tweets: Se extrajeron las respuestas de tweets publicados por periódicos masivos argentinos del 10 de febrero al 9 de junio de 2020 con más de 9 respuestas y que hagan mención directa al artículo en su página web.
2. Clasificación binaria de presencia de discurso de odio en los tweets.
3. Detección de motivo discriminatorio (según INADI) si presenta Discurso de odio.
4. Identificación del daño generado y su capacidad de ser mensurable.

En el análisis se plantearon preguntas divididas en dos categorías:

- Descriptivas: permiten caracterizar elementos textuales que organizan el Discurso de odio en la pandemia.
- Exploratorias: permiten determinar factores que afectan el emerge del Discurso de odio en respuesta a artículos periodísticos.

Por otro lado, (Schaefer & Stede, 2020) presentan un corpus de tweets en alemán anotados con unidades argumentativas a nivel de palabra, permitiendo así conseguir una granularidad en el tweet más acertada debido a la posibilidad de la existencia de unidades como afirmación y evidencia al mismo tiempo.

El corpus generado consiste de 12.296 tweets obtenidos en el año 2019 relacionados con el clima y calentamiento global. Se realizó un filtrado extrayendo aquellos tweets no redactados en alemán y retweets.

Las anotaciones fueron llevadas a cabo por 2 anotadores expertos que etiquetaron las afirmaciones y evidencias en los tweets. Se hace notar que las primeras son independientes, mientras que las segundas dependen de una afirmación.

En el proceso de anotación no se diferenció si la evidencia ataca o apoya una afirmación, sólo si la evidencia se relaciona con una afirmación proveniente de un tweet, respuesta a tweet o ambos.

En esta tarea, el acuerdo entre los anotadores por la presencia de evidencia fue la más dificultosa debido a la subjetividad misma que implicó esta etapa.

Detección automática de elementos argumentativos

(Schaefer & Stede, 2021) realiza un análisis profundo a trabajos previos sobre de la minería de argumentos en twitter destacando las dificultades que esta plataforma presenta con respecto a los tweets redactados por usuarios como son el ruido, lenguaje irregular, preguntas retóricas o sarcasmo.

Para la minería de argumentos, se diferencian generalmente 4 pasos clave: La detección (general) de argumentos, detección de afirmaciones, detección de evidencia y la relación entre estas dos últimas.

Tanto la detección de argumentos, afirmación y evidencia se basan generalmente en clasificación supervisada. En el primero, modelos como SVM, LR y XGBoost obtienen los mejores desempeños en la tarea, aunque actualmente se están comenzando a utilizar word embeddings contextuales obtenidos de BERT.

A lo que se refiere a detección de afirmación, DT y XGBoost fueron modelos con mejor desempeño. Actualmente también se prueban nuevos modelos con embeddings basados en BERT y unigramas aumentados con features más sofisticadas como lo son features propias de Twitter.

La detección de evidencia es una de las tareas más importantes y desarrollada en el marco académico e investigación, pues determina la calidad de un tweet como unidad

argumentativa, debido que la gran mayoría de tweets no son argumentativos. Modelos como SVM, LR, XGBoost y CRF obtuvieron mejor desempeño en esta tarea.

La detección de relación entre afirmación y evidencia se basa en clasificar, generalmente, dicha evidencia como ataque o apoyo a la afirmación. Es una tarea difícil debido a la complejidad de contenido en los tweets lo cual dificulta relacionarlos entre ellos. Se suelen utilizar arquitecturas LSTM, las cuales de cualquier forma obtienen bajo desempeño.

Se propone además la construcción de un grafo, una etapa poco investigada donde cada nodo es un tweet argumentativo y las artistas representan la relación de ataque o apoyo entre sí.

Dicho grafo sería de interés para el modelado de una discusión en Twitter, lo cual podría beneficiar las estrategias de minería de argumentos en Twitter.

Diferentes aproximaciones a la detección

En cuanto a arquitecturas se refiere, (Huang et al., 2015) propone diversas variedades basadas en LSTM para el etiquetado de secuencias destacando sus ventajas.

Comenzando con una red LSTM tradicional que es un tipo de una red neuronal recurrente (RNN) la cual mantiene una memoria basada en el historial de ejecución, es decir, se cuenta con un estado oculto que se nutre a partir de las sucesivas secuencias de ejecución del modelo lo cual le permite realizar clasificaciones bajo condiciones de corta distancia. A diferencia de una RNN, la arquitectura LSTM presenta lo que es llamado *célula LSTM* que además le permite hallar y explotar dependencias a larga distancia.

Proponen luego la BiLSTM, una arquitectura que se compone de dos redes LSTM donde cada una comienza sus etiquetados partiendo de cada extremo de la sentencia. Vemos aquí una diferencia en como una sola LSTM recibe como entrada una secuencia infinita de palabras, mientras que la BiLSTM dicha secuencia es finita. Debido a las dos subredes, la BiLSTM cuenta con dos estados ocultos, los pasados y futuros, que permiten al modelo realizar predicciones bajo contextos de palabras pasadas y las que le siguen a la palabra a predecir lo que favorece cierto tipos de tareas.

Para las próximas variaciones de LSTM proponen una red CRF que se trata de un clasificador el cual, basándose en las predicciones de etiquetados pasados, realiza el etiquetado de la palabra actual. Así esta red no cuenta con estados ocultos, pero si una conexión de *feedback* entre las salidas para tener en memoria dichas predicciones previas.

Combinando la red LSTM con una red CRF, obtienen una nueva arquitectura LSTM-CRF que es capaz de explotar eficientemente las dependencias de contexto a corta y larga distancia y el etiquetado de palabras previas.
Finalmente se propone la arquitectura BiLSTM-CRF que combina una red BiLSTM junto a una red CRF con las mismas ventajas que el modelo LSTM-CRF además de poder utilizar eficientemente las dependencias a futuro gracias a la componente BiLSTM.

Concluyen notando que esta última arquitectura, BiLSTM-CRF presenta el mejor desempeño además de robustez e independencia frente a las *features* utilizadas en tareas clásicas como son *Part-Of-Speech tagging*, *Chunking* y detección de entidades nombradas.

Como un primer proyecto similar al propuesto en este trabajo (Chernodub et al., 2019) presentan TARGER, un framework neuronal para el minado y etiquetado de unidades argumentales.

El objetivo principal de este proyecto es disminuir la brecha entre los interesados inexpertos en la detección automática de unidades argumentales y modelos neuronales. Esto lo logra a partir del despliegue de una API REST donde el usuario puede enviar el texto a etiquetar, pudiendo elegir entre el modelo a utilizar ya que cuentan con modelos entrenados en corpus como son *Persuasive Essays*, *Web Discourse* e *IBM Debater*, recibiendo como respuesta el texto etiquetado.

Para disminuir aún más la brecha, cuentan con una página web muy intuitiva y simple para la comunicación con los modelos disponible en:
http://ltdemos.informatik.uni-hamburg.de/targer/.

TARGER utiliza un modelo BiLSTM-CNN-CRF para identificar las unidades argumentales del texto crudo enviado y clasificar dichas unidades, arrojando un alto desempeño en ambas subtareas.

Por otro lado, (Abkenar & Stede, 2021) extiende un modelo BiLSTM-CRF con la utilización de word embeddings contextuales para etiquetar unidades argumentales como justificación o afirmación en un texto.

Obtuvo así un mejor desempeño alcanzando el estado del arte en la tarea frente al enfoque tradicional con word embeddings pre-computados y sin contexto. Detallaremos más en profundidad los word embeddings en la próxima sección.

(Mensonides et al., 2019) presenta la utilización de un modelo Multitask para la extracción y clasificación de unidades argumentales utilizando técnicas de aprendizaje profundo.

El modelo se compuso de la siguiente pila de tareas::

- Tarea de *Part-Of-Speech tagging*: utilizando un modelo BiGRU cuya entrada son los word embeddings de la capa anterior
- Tarea de *Chunking*: donde se computan estados ocultos explotando lo que el modelo aprende de la capa de Part-Of-Speech tagging
- Tarea de *Argument Components Detection*: tratado como un problema supervisado de segmentación de texto para delimitar cada unidad argumental a nivel de palabra.
- Tarea de *Argument Components Classification*: tratado como un problema de etiquetado de segmentos para determinar si se trata de una afirmación mayor, afirmación o premisa.

De esta forma se llevó a cabo una experimentación de dos versiones de este modelo, uno donde se entrenaban todas las tareas de la pila, y otro donde se omitían las tareas de *Part-Of-Speech tagging* y *Chunking*.

Resultados del modelo entrenando todas las tareas fue superior y similar al desempeño humano demostrando la eficacia del Multitask. Detallaremos y profundizaremos más de esta técnica en próximas secciones.

Finalmente, (Jahan & Oussalah, 2021) provee un análisis sistemático en el campo de Discurso de odio de los últimos 10 años con un foco en procesamiento del lenguaje natural y técnicas de aprendizaje profundo.

Para el análisis se utilizaron publicaciones del área de Ciencias de la Computación e Ingeniería recolectadas de *ACM Digital Library* y *Google Scholar*, obteniendo así 463 publicaciones de las cuales 96 siguen una metodología de aprendizaje profundo. Además 237 (51%) de dichas publicaciones utilizan el idioma inglés para el trabajo con discurso de odio, pero tan solo 5 (1%) el español.

Como vemos en la Figura 1, para la obtención de los corpus para entrenamiento de los modelos 47% proviene de Twitter, seguido de Facebook y Youtube donde el contenido generado por el usuario es basto y en gran medida presenta vocabulario y expresiones habituales de las personas en una comunidad.

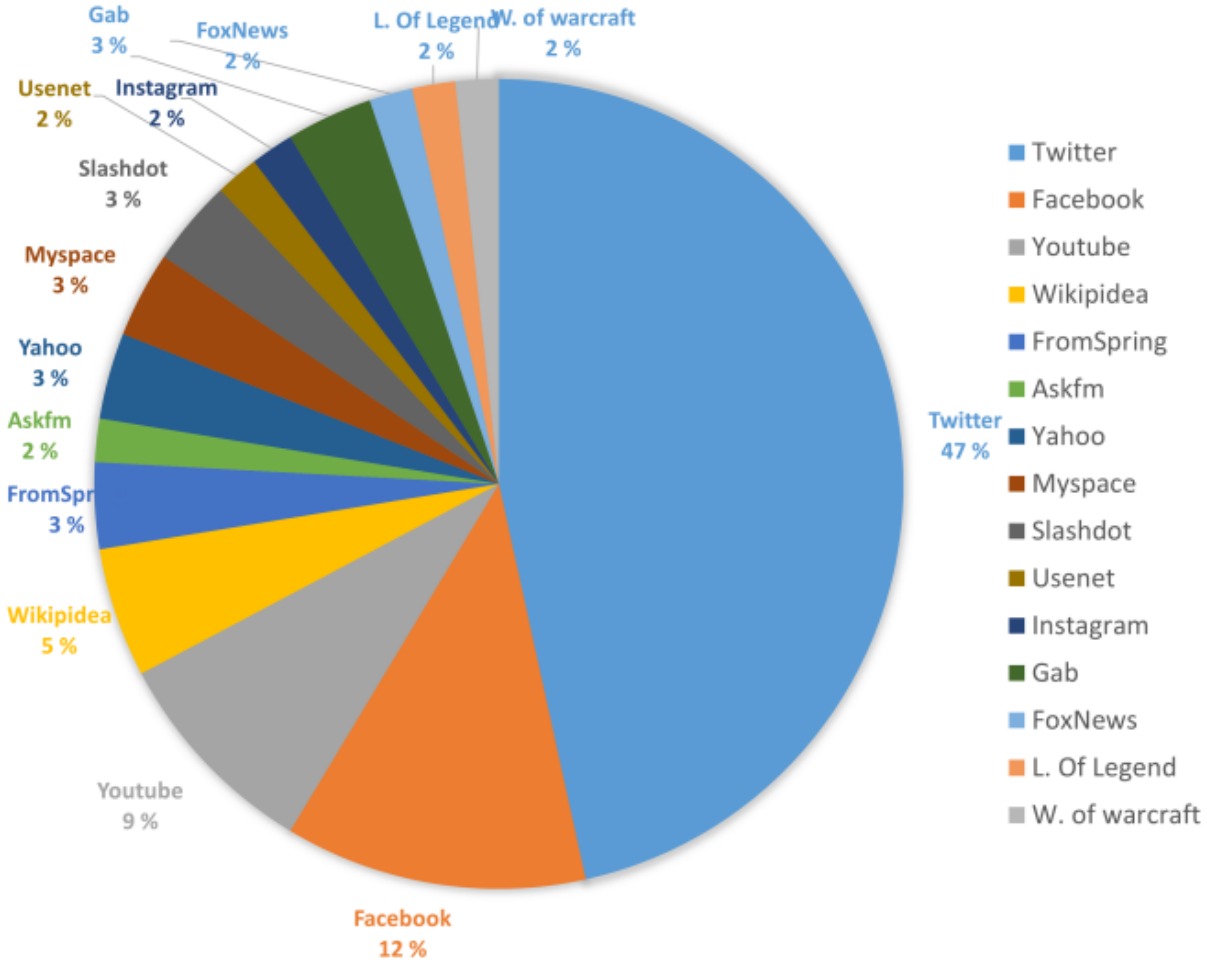


Figura 1. Aplicación de origen más frecuentadas para la generación de datasets con objetivo al discurso de odio. Cerca del 50% se comprende de ejemplos de twitter, seguido de Facebook y Youtube. Todas estas redes sociales masivas.

En general los dataset obtenidos fueron relativamente pequeños (alrededor del 41% cuentan con 1 a 5000 ejemplos) y presentando un bajo ratio de contenido de odio en 37% de los corpus. Falta de acuerdo entre los anotadores y definición rigurosa de las etiquetas utilizadas fueron problemáticas mencionadas en muchos trabajos.

Los algoritmos más utilizados para la detección del discurso de odio corresponde con un 29% a Support Vector Machine (SVM), un 22% a técnicas de aprendizaje profundo con una gran presencia del modelo de lenguaje BERT y redes LSTM con variaciones como observamos en la Figura 2, y un 20% algoritmos de regresión logística.

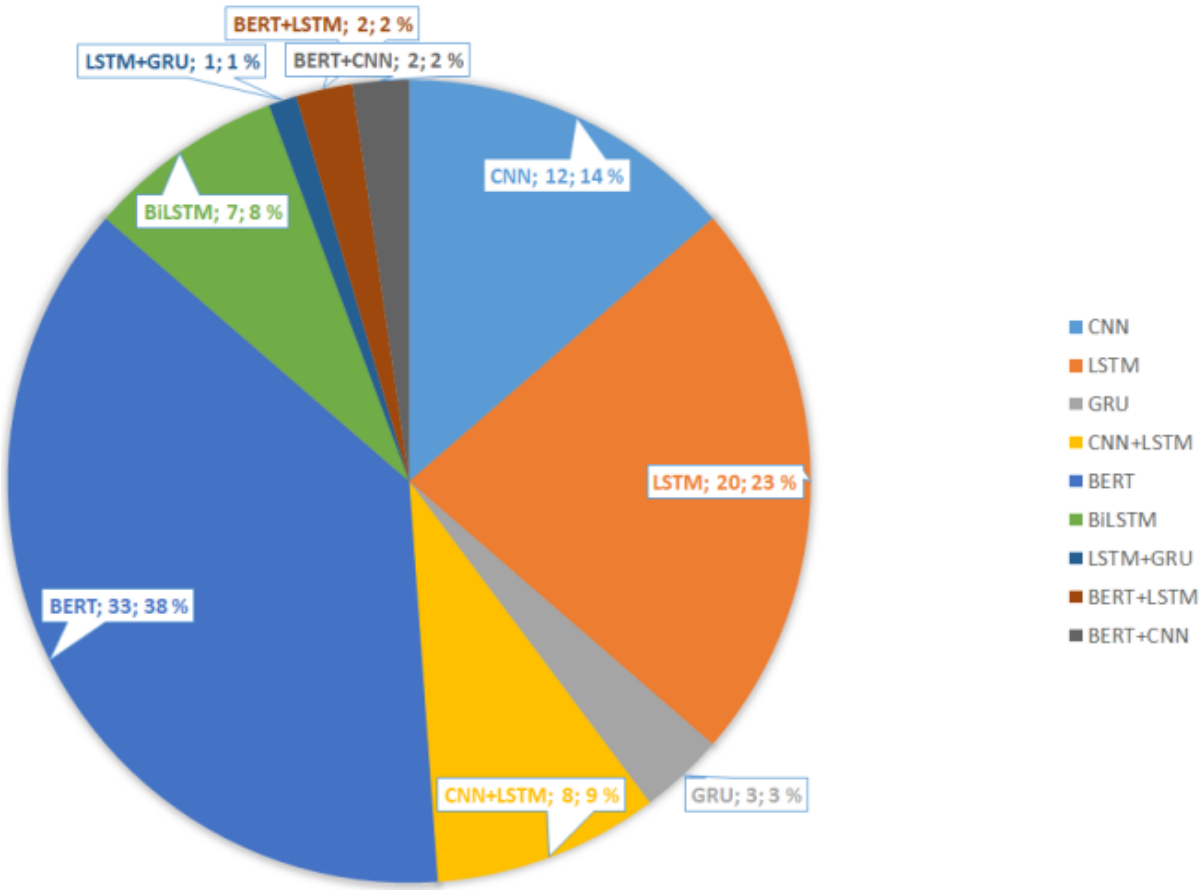


Figura 2. Distribución de porcentajes de arquitecturas utilizadas en el área de discurso de odio. Modelos de lenguaje están presentando los últimos años una gran presencia, pero donde la vasta mayoría se tratan de arquitecturas de aprendizaje profundo como son redes LSTM.

Se hace notar además, que en la utilización de técnicas de aprendizaje profundo, son modelos no pertenecientes a esta categoría los cuales se utilizan como baseline para comprar los primeros.

Figura 3 evidencia las diferentes fuentes de word embeddings utilizadas. Notar un uso de 34% proveniente de Word2Vec que genera word embeddings sin contexto pero justificado por velocidad a la hora de obtener el embedding. También observamos embeddings que provienen de FastText el cual ha acaparado gran parte de los proyectos por su metodología para obtener embeddings en escenarios de mucho ruido como lo es el generado por usuario. Además, se observa un 17% de casos donde se utilizan word embeddings

contextuales en los que generalmente se obtiene una mejora del 5%, según reportan los diversos proyectos.

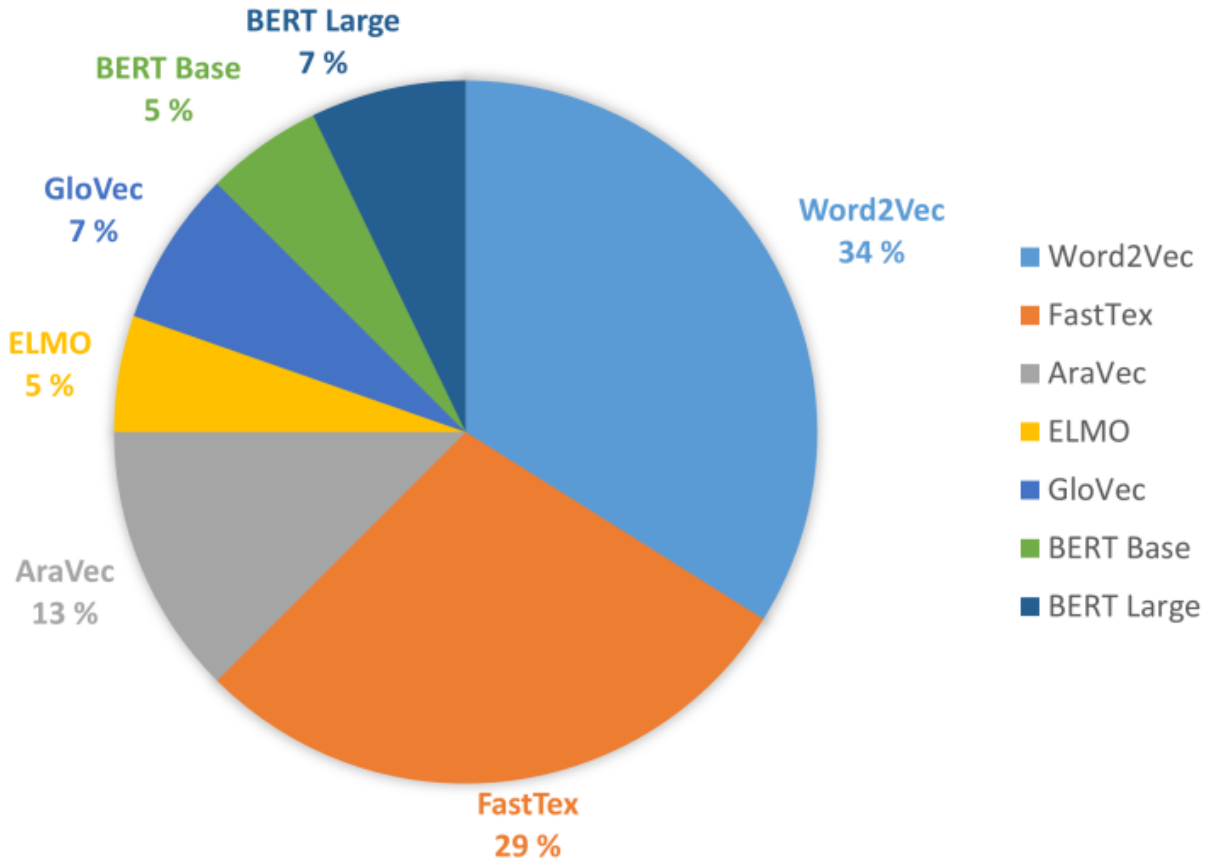


Figura 3. Embeddings más utilizados en el área de discurso de odio. Predomina Word2Vec por su optimización y eficiencia en tiempo de obtención del embedding, seguido de FastText por su eficiencia contemplando además la estructura interna de la palabra para obtener embeddings de palabras desconocidas frecuentadas en redes sociales donde el texto del usuario presenta mucho ruido. Embeddings ricos en contexto están teniendo también un gran volumen de casos de utilización en los últimos años.

En el estudio de las diferentes publicaciones se puede ver que inicialmente algoritmos SVG con features de TF-IDF eran ampliamente utilizados, las cuales fueron reemplazadas por modelos de aprendizaje profundo que a su vez fueron conectados entre sí para generar modelos con mejor desempeño aún como en el caso de CNN+LSTM o CNN+GRU. Es así como se observa una clara evolución al aprendizaje profundo para las tareas de discurso de odio, junto a la tímida pero presente utilización de modelos de lenguajes que prometen superar umbrales nunca antes vistos.

Generación de contranarrativas

En lo que a generación de contranarrativas (CN) se refiere, partiendo por la obtención de ejemplos para el entrenamiento de modelos, existen dos aspectos fundamentales a tener en cuenta: la cantidad y calidad. Donde esta última se define como conformidad a una guía de puntos a seguir provista y la diversidad de las CN mismas.

Existen varios enfoques de obtención, aquí 3 alternativas:

- *Crawling* (CRAWL): propone encontrar la CN en las respuestas del discurso de odio (suponiendo que existen).
- *Crowdsourcing* (CROWD): se construye una lista de discursos de odio y se instruyen a personas para redactar manualmente las CN.
- *Nichesourcing* (NICHE): similar a CROWD, pero donde las personas que redactan las CN son profesionales específicamente entrenados para combatir el discurso de odio online vía respuestas textuales.

CRAWL no es un enfoque prometedor, pues presupone la existencia de respuestas donde esté explícito una CN de calidad. Por otro lado, con CROWD se obtienen mucha cantidad de CN con la desventajas que incluyen argumentos simples y estereotipados, bajando así la calidad de los mismos. NICHE, por su lado, es el mejor enfoque en cuanto a calidad de CN, pero la dificultad que los expertos estén de acuerdo en una CN, más el tiempo que ésta toma en ser formulada convergen en una poca cantidad de CN totales.

Es así como (Tekiroğlu et al., 2020) proponen un enfoque alternativo para la obtención de CN, una arquitectura Autor-Revisor donde el Autor genera CN y el Revisor asegura que dicha CN satisfaga propiedades de calidad para luego ser entregadas a un experto para su validación o edición final.

Para el Autor se entrenaron 2 modelos, obtenidos a partir de finetunear modelos GPT-2 medianos (345 millones de parametros, 24 capas, 16 cabezas de atención y tamaño de estado ocultos de 1024), uno con CN obtenidas a partir de NICHE (GPT_{NICHE}) y otro con CN obtenidas a partir de CROWD (GPT_{CROWD}).

GPT_{NICHE} fue finetuneado con 5.366 pares de discurso de odio - CN obteniendo un mejor rendimiento que GPT_{CROWD} cual fue entrenado con 26.350 pares. Además, ambos modelos son capaces de generar múltiples CN para un mismo discurso de odio.

Para el Revisor se proponen 3 estrategias:

1. Revisor experto: Pasar la CN directo al experto para su validación o edición final.
2. Revisor no experto: Incorporar personas que filtren las CN bajo instrucciones claras manteniendo la calidad, para su próxima validación por un experto.
3. Revisor máquina: El filtrado lo realiza una arquitectura neuronal previa validación del experto.

Para la estrategia de Revisor máquina, se entrenaron 2 modelos basados en BERT large y ALBERT xx-large con 1.373 pares correctos de Discurso de odio - CN y 1.373 pares incorrectos, donde ALBERT obtuvo un mejor desempeño.

De las estrategias propuestas, se declina por esta última debido a que la primera supone un alto trabajo por parte del experto debiendo validar todas las CN generadas por GPT_{NICHE} y el segundo también exige un trabajo humano además de los desacuerdos que estos pueden tener acarreando como consecuencia una exigencia de tiempo mayor.

Para diversos modelos que trabajan con CN, ya sea para su generación u obtención, pueden ser beneficiados por una clasificación previa según su tipo. El problema es que un

alto porcentaje del trabajo investigador sobre discurso de odio y CN se desenvuelve en la lengua inglesa.

Es así como (Chung et al., 2021) realizan un trabajo pionero en la clasificación de CN en diversos lenguajes. Donde emplearon el modelo de lenguaje XLM-R, el cual fue pre entrenado con *CommonCrawl* para 100 lenguajes, fine tuneando con 4 configuraciones diferentes en sus dataset de entrenamiento y testeo:

- Monolingüe: Dataset de entrenamiento y testeo en el mismo idioma.
- Plurilingüe: Entrenado en dataset de varios idiomas (inglés, francés e italiano) y testeado en cada uno de ellos.
- *Zero-shot Cross-lingual*: Entrenado en un idioma y testeado en los otros dos no vistos por el modelo.
- *Zero-shot Translated*: Similar al Plurilingüe pero traduciendo los dos lenguajes distintos al inglés.

Dicho dataset de entrenamiento se trata del CONAN de 2019 a la cual fecha del paper era el único corpus con temática de discurso de odio plurilingüe obtenido por la estrategia de recolección NICHE.

Los tipos de CN propuestos en su trabajo fueron: Hecho, Pregunta, Denuncia, Humor o Hipocresía con los que se observó que, los modelos con configuración Plurilingüe obtuvieron mejores resultados de predicción en general que las configuraciones Monolingüe, excepto en Hecho y Humor. De hecho ambas configuraciones presentan dificultades para predecir las CN de tipo Humor pues ambas tienen un acierto de tan solo 50%, dejando en evidencia además la clasificación de este tipo de CN.

La configuración *Zero-shot Cross-lingual* obtuvo peor desempeño que la configuración Plurilingüe, pero los resultados dan prueba que la transferencia de lenguaje cruzada es una alternativa factible para clasificar CN en lenguajes sin un gran dataset de entrenamiento.

La última configuración, *Zero-shot Translated*, obtuvo mejores resultados que la configuración Plurilingüe además de conseguir superar el 50% de aciertos en la clase de Humor. Esta última configuración deja en evidencia que ciertas características de las contra narrativas se transfieren a través del lenguaje.

Finalizamos esta sección con un paper de (Reimers & Gurevych, 2017) importante para la comparación según métricas de los modelos resultantes, algo de interés para este trabajo.

Los nuevos enfoques no deterministas aplicados en las redes neuronales son indirectamente guiados por valores aleatorios, presentes desde la inicialización de los pesos de las neuronas, mezclado de datos de entrenamiento para las distintas épocas y hasta en la aplicación de máscaras de *dropout*.

Debido a esto, evaluar estos modelos mediante un único valor resultante de cierta métrica es poco representativo de la realidad del mismo modelo, es decir por ejemplo, un modelo puede no ser mejor que otro al cual se lo compara, solo verse favorecido por la semilla inicializadora de los valores aleatorios, o a la inversa, un modelo puede ser mejor que otro efectivamente, pero al momento de obtener el valor resultante de comparación, puede verse

afectado negativamente por la mala semilla inicialización de valores aleatorios. Además, es difícil replicar los resultados, lo cual es necesario para la validación del modelo mismo.

Es por ello que (Reimers & Gurevych, 2017) introduce como alternativa la comparación a través de la distribución de resultados obtenidos de los modelos a partir de múltiples iteraciones.

Además, se evalúan las redes LSTM en distintas tareas como *Part-Of-Speech tagging*, *Chunking*, detección de entidades nombradas (NER), detección de entidades y detección de eventos mediante la utilización de distintos hiperparametros de la arquitectura para conocer cuales presentan una mayor independencia entre ellos y los valores aleatorios, y así obtener resultados estables. Esto nos será de interés para el desarrollo del presente trabajo.

Conclusiones: análisis de los distintos abordajes al problema

Vemos que tanto para la detección como clasificación de componentes argumentativas utilizan un amplio espectro de modelos y abordajes, desde arquitecturas clásicas o aprendizaje profundo como son regresión logística, SVG, GRU o LSTM como modelos muy populares en la actualidad como son los modelos de lenguaje y sus diversas ramas a partir de entrenamiento y búsqueda de objetivos específicos.

Observamos que estos últimos modelos de lenguaje están comenzando cada vez más a tener una presencia en lo referido al discurso de odio. Tradicionalmente se utilizan técnicas de aprendizaje profundo que comprenden GRU o LSTM teniendo así un amplio espectro de trabajos y literatura de consulta a disposición. Estos modelos de lenguaje están presentando resultados SOTA en muchas tareas, aunque se debe evaluar su desempeño en las relacionadas a la temática de este trabajo.

En sí son modelos complejos y pesados que requieren un alto costo y recurso computacional. Por otro lado, las arquitecturas de aprendizaje profundo son más ligeras y simples, además de que existen muchas buenas implementaciones avaladas por papers que la utilizan.

Es por estas razones que decidimos explorar con detenimiento el desempeño de las redes BiLSTM, como aquella que utiliza el proyecto TARGER, que presenta una muy buena implementación configurable que describiremos en la siguiente sección.

Propuesta de Arquitectura

Por lo expuesto en la sección anterior, trabajamos con una arquitectura basada en redes neuronales recurrentes. Como línea base de comparación aplicamos también a la misma tarea un clasificador lineal simple, una regresión logística. Los resultados obtenidos también se compararon con un clasificador basado en modelos de lenguaje, descrito en (Furman et al., 2022).

Baseline: Regresión Logística

La regresión logística es un algoritmo de aprendizaje automático utilizado para problemas de clasificación binaria. Esencialmente se basan en una función logarítmica para modelar el resultado de probabilidad dado una entrada.

Parte de la idea de utilizar una función lineal para clasificación binaria pero, debido a la necesidad de tratarse de una probabilidad se aplica la función sigmoide para disminuir el rango de valores de salida entre 0 y 1.

Dado un vector x, pesos del modelo w y el bias b, se calcula la función lineal:

$$z = x \cdot w + b$$

Para luego reducir la imagen con la función sigmoide:

$$\sigma(z) = \frac{1}{1 + e^{-z}}$$

De esta forma, si se sobrepasa un determinado umbral corresponde clasificarla a una clase u otra.
Con esto vemos que se trata de un modelo muy sencillo y veloz frente a arquitecturas como las que describiremos próximamente.

Como entrada a este modelo, se emplearon dos alternativas: word embeddings que describiremos en la próxima sección, y embeddings a partir de un modelo *Bag Of Words* (BOW). Este último genera vectores a nivel de sentencia donde las columnas representan las palabras observadas en el entrenamiento y las filas las sentencias mismas. El valor que se almacena en cada celda corresponde al número de ocurrencia de la palabra de la columna en la sentencia de la fila. Por ejemplo, consideremos la siguiente sentencia:

Ana Maria Polo le gusta cerrar casos. Al desgraciado no le gusta perder.

Su representación vectorial mediante BOW podría ser *[1, 1, 1, 2, 2, 1, 1, 2, 1, 1, 2, 1*], dado por lo evidenciado en la Tabla 1.

Sentencia	Ana	Maria	Polo	le	gusta	cerrar	casos	.	Al	desgraciado	no	perder
A Ana Maria Polo le gusta cerrar casos. Al desgraciado no le gusta perder.	1	1	1	2	2	1	1	2	1	1	2	1

Tabla 1. Generación de embedding de sentencia por método BOW. Columnas corresponde al vocabulario encontrado en los ejemplos, en cuyas celdas se contabiliza la ocurrencia de cada una de ellas en la sentencia en cuestión.

.

En el caso del presente trabajo, debido a que debemos obtener representaciones vectoriales a nivel de palabra, pues es a ese nivel que deseamos clasificar con la regresión logística, compusimos sentencias a partir de las palabras presentes en el tweet considerando una ventana por delante y detrás de dos palabras.

Modelo: BiLSTM-CNN-CRF

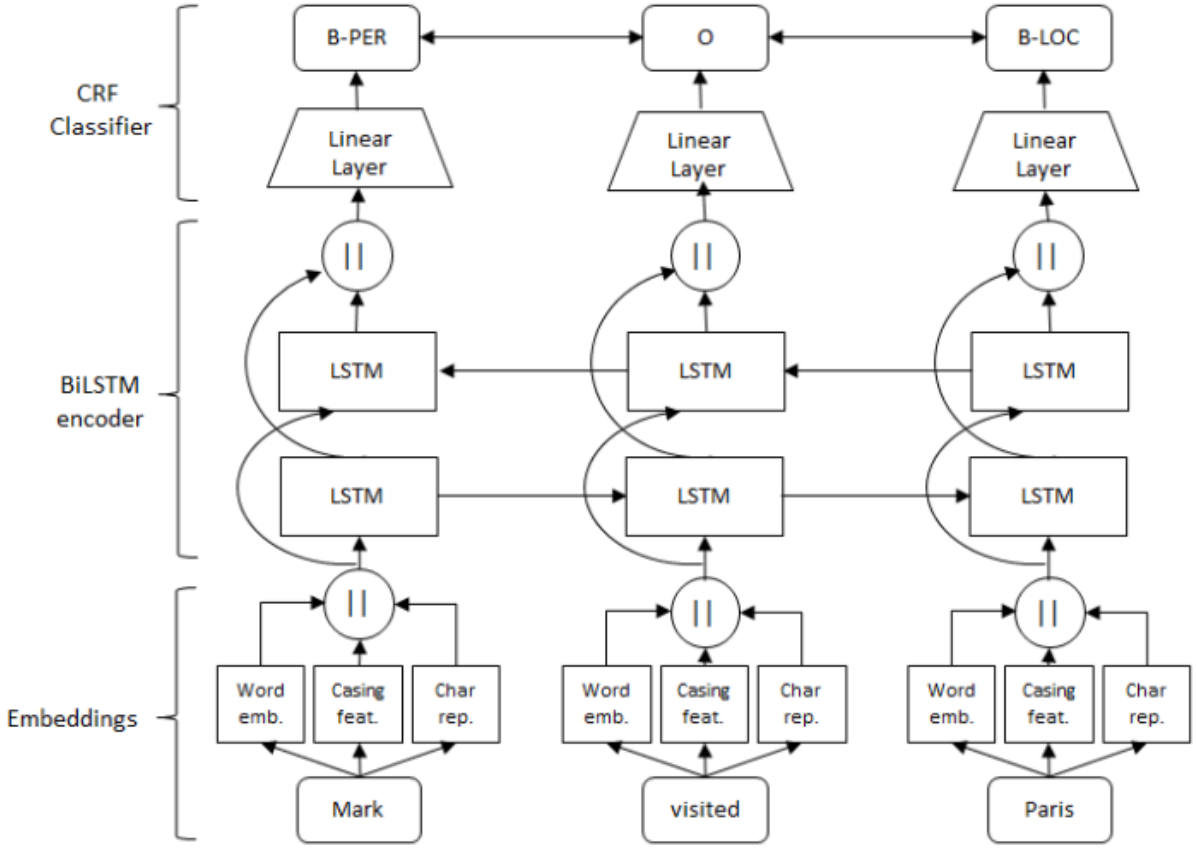


Figura 4. Arquitectura de red BiLSTM-CNN-CRF utilizada.

El modelo utilizado en este trabajo se basa en la arquitectura BiLSTM-CNN-CRF propuesta por Nils Reimers en (Reimers & Gurevych, 2017) disponible en su Github. Diseñada para tareas de etiquetado de secuencias como *Part-of-Speech tagging*, detección de entidades nombradas y *Chunking* llegando al estado del arte.

Definen a esta arquitectura como altamente configurable, algo que nos es atractivo para explorar con este trabajo diferentes seteos de los posibles hiperparametros. Además, la arquitectura trabaja a nivel de palabra, es decir, clasifica palabra por palabra como perteneciente a las clases posibles dado una sentencia

Esta arquitectura cuenta con varias capas, como vemos en su diagrama arquitectónico en la Figura 4. Estos son: una capa de *word embeddings* y *char embeddings*, una red BiLSTM y una red *"linear chain"* CRF. De cualquier forma, todas estas capas son sustituibles por otras, como por ejemplo un clasificador Softmax en vez de la *"linear chain"* CRF.

Capa de *embeddings* de palabras (*Word Embeddings*)

Esta capa es responsable, como su nombre lo indica, de obtener los *embeddings* de las palabras (*word embeddings*). Estos *embeddings* son representaciones en un espacio vectorial que capturan la semántica y sintaxis de estas para poder ser interpretadas por los modelos.

Existen diversas técnicas para optimizar la obtención de los *embeddings*, sobre todo en los últimos años donde se ha visto que la representación de las palabras influyen de forma significativa en el desempeño del modelo obtenido.

Los *embeddings* usualmente son de dimensionalidad baja (entre 50 y 600 dimensiones), densos que permiten la modelización de relaciones semánticas como operaciones en dicho espacio vectorial, un ejemplo clásico de esto es calcular con los vectores correspondientes Rey - Hombre + Mujer obteniendo un vector similar al de Reina (*hombre es a rey lo que mujer es a reina*).
Los *embeddings* utilizados en este trabajo son aquellos obtenidos por los siguientes modelos:

Word2Vec

(Mikolov et al., 2013) introdujo un método eficiente para aprender embeddings de palabras a partir de grandes corpus de datos.
Bajo la premisa:

"You should know a word by the company it keeps" - J.R. Firth 1957

El paper presenta dos tareas para aprender a representar palabras por un modelo.

Continuos Bag Of Words (CBOW)

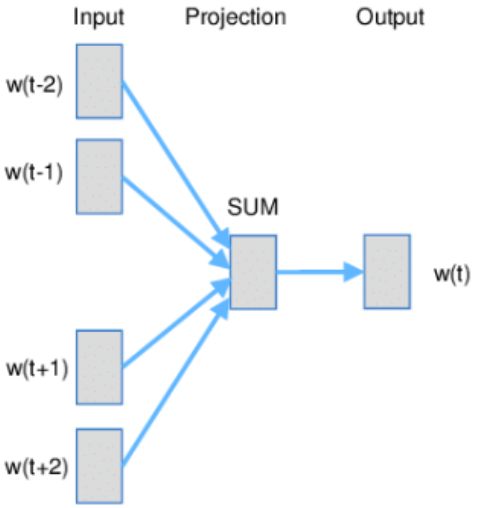


Figura 5. Esquema de predicción de método CBOW. Dada palabras vecinas como entrada, predecir la palabra central.

En esta tarea, como se observa en la Figura 5, se busca que el modelo pueda predecir una palabra por sus palabras vecinas en una ventana de tamaño N dada, tomando una sentencia (incompleta) como input.
Por ejemplo, a partir de la sentencia:

A quick brown fox jumps over a lazy dog

con una ventana de tamaño 2 podemos generar los siguientes ejemplos:

[((A, brown), quick), ((quick, fox), brown), ((brown, jumps), fox), …]

que son entrada al modelo para su entrenamiento y así ser capaz de realizar predicciones.

La arquitectura del modelo, como observamos en la Figura 6, se compone de una capa de embeddings para la cual a cada palabra del vocabulario le asigna un vector del tamaño deseado con valores aleatorias al inicializarse, una capa lambda que calcula el embedding promedio entre los suministrados como entrada y un clasificador softmax.

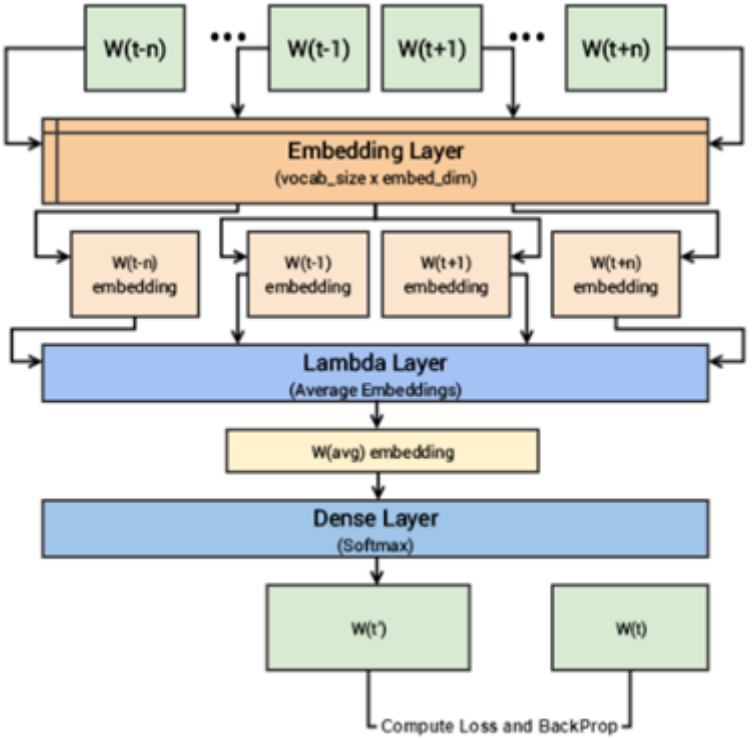


Figura 6. Arquitectura del modelo CBOW. Presenta una capa de embeddings aleatorios, una capa lambda para obtener un promedio de los embeddings de palabras vecinas, concluyendo con una capa de clasificación Softmax para predecir la palabra central.

Para entrenar en modelo en cuestión se siguen los siguientes pasos:

1. Se suministran las palabras vecinas a la capa de embeddings
2. Luego, estos embeddings aleatorios son pasados a la capa lambda para obtener el embedding promedio de ellas.
3. Finalmente, se suministra el resultado al clasificador Softmax que predice la próxima palabra en la secuencia. Comparamos esta palabra con la deseada y calculamos la pérdida para luego aplicamos un backpropagation para actualizar la capa de embeddings en el proceso de entrenamiento.

Skip-gram

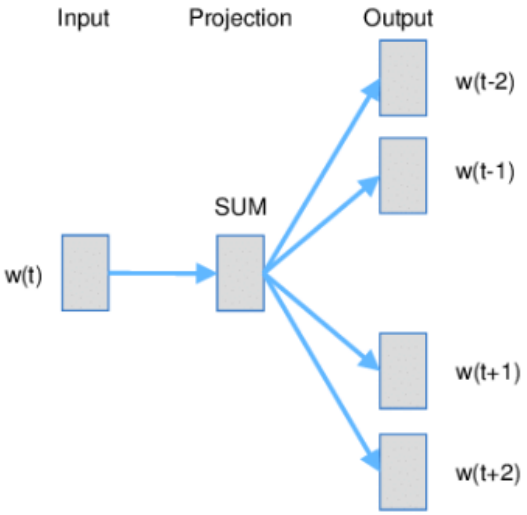


Figura 7. Esquema de predicción de método Skip-gram. Dada una palabra central, predecir las palabras vecinas a esta.

En esta tarea, un poco más compleja, se busca que el modelo pueda predecir las palabras vecinas bajo una ventana de tamaño N, dada una palabra como input. En la Figura 7 observamos el caso de una ventana con N = 2. Con la sentencia de ejemplo de CBOW y mismo tamaño de ventana, dada la palabra *fox*, esperar que el modelo pueda predecir las palabras *brown* y *jumps*.

Como el modelo debe predecir múltiples palabras dada una única palabra, la entrada del modelo son pares (***X***, ***Y***) donde ***X*** es la palabra de entrada e ***Y*** la etiqueta. Estos pares son creados a partir de ejemplos positivos y negativos.

Los ejemplos positivos tienen la forma de: *[(palabra, contexto), 1]* donde la palabra es aquella de entrada y contexto representa las palabras vecinas, con la etiqueta 1 que simboliza que se trata de un ejemplo positivo. Ejemplos negativos siguen la misma forma: *[(palabra, random), 0]* donde en vez de representar las palabras vecinas reales, se seleccionan un conjunto al azar indicando con la etiqueta 0 que se trata de un ejemplo negativo.

Su arquitectura, como se ilustra en la Figura 8, se compone por dos capas de embeddings separadas donde para cada palabra del vocabulario se le asigna un vector con valores aleatorios del tamaño deseado de los embeddings finales, luego una capa de *Merge* en la cual se calcula el producto punto de los embeddings que son la entrada a una capa densa donde se aplica la función sigmoide para obtener un valor de salida 0 o 1. De esta forma, la salida de esta última capa es comparada con la esperada para luego calcular el error y aplicar el backpropagation para actualizar las capas de embedding

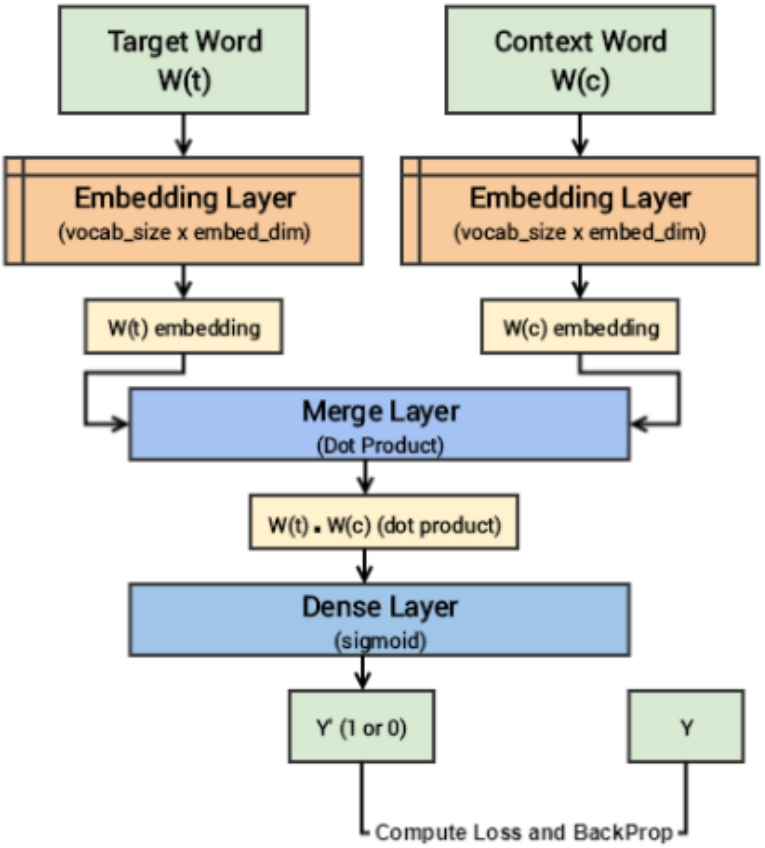


Figura 8. Arquitectura del modelo Skip-gram. Cuenta con dos capas independientes de embeddings aleatorios, seguido de una capa que calcula el producto punto entre la palabra central y vecina para culminar con la aplicación de función sigmoide para obtener un valor final de 0 y 1 para comparar en la etapa de entrenamiento.

De esta forma es como Word2Vec fue una revolución para el PLN, pero presenta un conjunto de falencias; la primera de ellas, viene de la mano de cómo ambos modelos propuestos parten de un vocabulario fijado, a partir del corpus utilizado. Esto acarrea la problemática de que el modelo no es capaz de obtener *embeddings* de palabras que no estuvieran en dicho vocabulario. A partir de estas palabras fuera del vocabulario, el modelo podría retornar un *embedding* por defecto, pero esto acarrea como consecuencia afectar al modelo en su aprendizaje.

El segundo problema es la morfología de las palabras, por ejemplo, para las palabras del inglés *eat* y *eaten* Word2Vec las aprende bajo sus propios contextos en los cual aparecen, cuando se podrían utilizar propiedades de la estructura misma de la palabra para un proceso más eficiente.

GloVe

(Pennington et al., 2014) argumentaban que lo presentado en Word2Vec no era óptimo, pues no explotaba la estadística global con respecto a las co-ocurrencias de las palabras.

Una diferencia clave entre los *embeddings* producidos por el método de Word2Vec y Global Vectors (GloVe), es que el primero tiene una naturaleza predictiva, mientras que la última se basa en contar.

El método de GloVe consiste en construir una matriz donde las filas son las palabras y las columnas el contexto donde el elemento X_{ij} es igual a la suma de veces que la palabra i ocurre en el contexto j. Esta matriz como es de esperarse termina siendo demasiado grande, es por ello que se emplean técnicas de factorización de matrices del álgebra lineal para reducir su dimensionalidad.

Notar que GloVe debido a esto requiere de un gran corpus para mejorar la calidad de sus *embeddings*, pero que además, sufre los mismo problemas de Word2Vec.

FastText

Para solventar este conflicto, (Bojanowski et al., 2017) propone un nuevo método de obtención de *embeddings* de palabras llamado FastText, donde el punto clave es la utilización de la estructura interna de las palabras para mejorar la representación vectorial obtenida del método de Skip-gram.

Para entrenar el modelo se realiza un Skip-gram con ejemplos negativos, dada una palabras se busca que sus palabras vecinas en la ventana de tamaño N sean más probables que palabras vecinas aleatorias, disminuyendo de esta forma también las probabilidades de que estas palabras aleatorias estén en el contexto de la palabra inicial. Dada la sentencia:

I am eating food now

con una ventana de tamaño 2 y centrándonos en la palabra eating, querríamos poder predecir las palabras *am* y food.

Para ello se insertan los símbolos ‘<’ y ‘>’ al inicio y final respectivamente de eating, obteniendo <*eating*>. De ella, se obtienen todos los 3-grama (los n-gramas pueden ser desde 3 a 6), es decir <*ea*, *eat*, *ati*, *tin*, *ing*, *ng*>, y se extraen los *embeddings* de estos.

Para obtener dichos embeddings, se utiliza una función de *hashing* para asignar a cada n-grama un valor entre 1 y B, donde B es el valor máximo lo cual permite controlar los requisitos de memoria (el paper original utiliza un B de 2 millones), además las colisiones que se pueden generar permiten controlar aún más el tamaño del vocabulario. Finalmente, se aprenden *embeddings* para estos n-gramas.

Con ellos, se suman junto al embedding de *eating* obteniendo una representación vectorial final. Luego se toman los embeddings directos de *am* y *food*, los ejemplos correctos a predecir, y de dos palabras aleatorias como pueden ser *parís* y *tierra*.

Se realiza un producto punto entre la palabra *eating* y sus contextos y se aplica la función sigmoid para obtener un valor entre 0 y 1. Basados en estos valores, se actualizan los *embeddings* de las palabras para acercar las del contexto correcto a la palabra *eating*, y alejar las del contexto incorrecto.

De esta forma, el modelo aprende *embeddings* de palabras del vocabulario. Además para aquellas palabras desconocidas retorna el *embedding* suma de los *embeddings* de los n-gramas de su estructura interna.
El paper asi concluye con las siguientes conclusiones:

- FastText es 1.5 veces más lento de entrenar qué Skip-gram debido a la carga del procesamiento de los n-gramas.
- En tareas de calcular similitud de palabras, FastText obtiene mejores resultados que CBOW y *Skip-gram* por la utilización de la estructura interna de las palabras.
- Para palabras desconocidas, utilizando los *embeddings* de FastText se obtiene mejor desempeño en diversas tareas de NLP ante la utilización de un *embedding* por defecto.
- FastText presenta peor desempeño que Word2Vec en tareas de analogía semántica de las palabras, por la fuerte componente sintáctica utilizada en la representación de sus embeddings.

Komninos

(Komninos & Manandhar, 2016) comparan diferentes word embeddings basados en la tarea de Word2Vec, skip-gram. Entre ellos, un modelo genérico de skip-gram, un modelo de skip-gram entrenado utilizando features de contexto de dependencia, y su modelo de embeddings que incorpora información a través de un grafo de dependencias (árbol sintáctico de la oración).

En su trabajo se comparan en tareas de hallar similitud entre palabras y clasificación de sentencias (estos son; tipo de pregunta, clasificación de sentimientos binarios y clasificación de relación semántica en sentencias).

El modelo de skip-gram con features de contexto de dependencia reemplaza las palabras que ocurren en una ventana por contextos de dependencia, esto es la palabra y su rol sintáctico. El entrenamiento de este modelo es similar al skip-gram genérico pero donde cada palabra es considerado un nodo del grafo de dependencia obtenido por un parser y los *embeddings* son optimizados para predecir su contexto sintáctico inmediato.

Su modelo de embeddings (Komninos), expande la ventana del modelo anterior teniendo en cuenta además las co-ocurrencias y los nodos del árbol sintáctico.
Cada palabra se considera un nodo en el grafo de dependencia y así los word embeddings son optimizados para maximizar la probabilidad dentro de una distancia de una o dos palabras

A raíz de todos estos métodos de obtención para la generación de *embeddings*, notamos un problema clave, una limitación: no son contextuales, por lo tanto no pueden distinguir diferentes sentidos o diferentes categorías sintácticas, sino que acumulan la información de todos los sentidos en la misma representación. Esta es una limitación que pueden superar los embeddings que se obtienen a través de modelos de lenguaje, que sí contemplan diferentes representaciones de una palabra según su contexto de uso, tal como se va a desarrollar en ELMo.

Capa de *embeddings* de caracteres (*Char Embeddings*)

Una capa de *embeddings* de caracteres es similar al de palabras. Esta capa tiene un rol importante en las situaciones donde no se cuenta con el embedding de la palabra, esto puede ocurrir por la utilización de un modelo de word embedding que nunca ha visto dicha palabra en su entrenamiento. Notar que en el texto generado por el usuario es común que se encuentren abreviaciones o modificaciones leves a palabras.

Frente a las palabras desconocidas como las mencionadas, un camino a tomar podría ser la asignación de un vector por defecto estrategia empleada por GloVe en su palabra especial *<unk>*, pero esto podría confundir y mal guiar al modelo en su aprendizaje. Otra alternativa más adecuada es la representación de palabras a nivel de caracteres con el objetivo de asimilar la palabra a otra conocida, bajo la premisa que esta palabra desconocida es quizá producto de un error ortográfico.

En esta arquitectura, el embedding a nivel de carácter se concatena con aquel de palabra, de tal forma de nutrir a la red BiLSTM con mayor información. Notar así que puede ser completamente inutilizada para conocer el impacto real y puro de los *word embeddings* específicos, estrategia que evaluaremos próximamente.

En este trabajo para generar el embedding a nivel de carácter se utilizó:

Red 1D-CNN

Esto es una red neuronal convolucional unidimensional que dada una palabra, genera una matriz **C** con dimensiones *d* x *l*, donde *d* es la dimensión del vector del carácter y *l* es la longitud de la palabra.

Luego, se genera un kernel **H** (también llamado filtro) con dimensiones *d* x *w*, con *w* menor a *l* en el cual los valores de **H** son inicializados de forma aleatoria y se van ajustando a medida que el modelo es entrenado. Notamos que esta matriz es *invariante de posición*, lo que permite capturar el significado de cierta combinación de caracteres independientemente de la posición en donde dicha combinación ocurra en la palabra, esto es debido a que **H** no modifica sus valores a lo largo de esta específica ejecución.

Con estas matrices se genera una nueva matriz que se obtiene a partir de calcular el producto *Hadamard* (producto punto a punto) entre **H** y su proyección en **C**. Obteniendo un escalar luego de sumar todos los valores de la matriz resultante.

Al deslizar **H** en **C**, se van obteniendo así un vector de escalares, del cual se extrae el mayor de todos. Este escalar es el primer valor del vector de representación de la palabra a nivel de carácter. Estos pasos se repiten hasta generar el vector de largo deseado con kernels H con *w* distintos. Notar así como este método es capaz de extraer *features* de interés a partir de la palabra dada.

Red LSTM

Aquí se utiliza una red LSTM, descrita en la próxima sección para la generación de char embeddings donde la secuencia de entrada son los caracteres que componen la palabra. Es decir, se trabaja a nivel de carácter en vez de palabra.

La 1D-CNN ha demostrado mejorar el desempeño en tareas de clasificación de texto, pero *embeddings* de caracteres obtenidos a partir de una BiLSTM han demostrado mejorar aún más en tareas como *Part-of-speech tagging* y detección de entidades nombradas.

Red BiLSTM

El modelo BiLSTM es un modelo recurrente para procesamiento de secuencias. Las redes neuronales recurrentes (RNN) tienen la propiedad de modelizar el contexto a través de un estado oculto lo cual beneficia al modelo para recordar ejemplos pasados.

A diferencia de una RNN tradicional, la arquitectura BiLSTM cuenta con *celdas LSTM*, como las que vemos en la Figura 9, que son más complejas en naturaleza y que permiten capturar contextos más distantes. Estas celdas cuentan con una compuerta de entrada, salida y olvido, de tal forma que permiten recordar valores vistos por el modelo previamente.

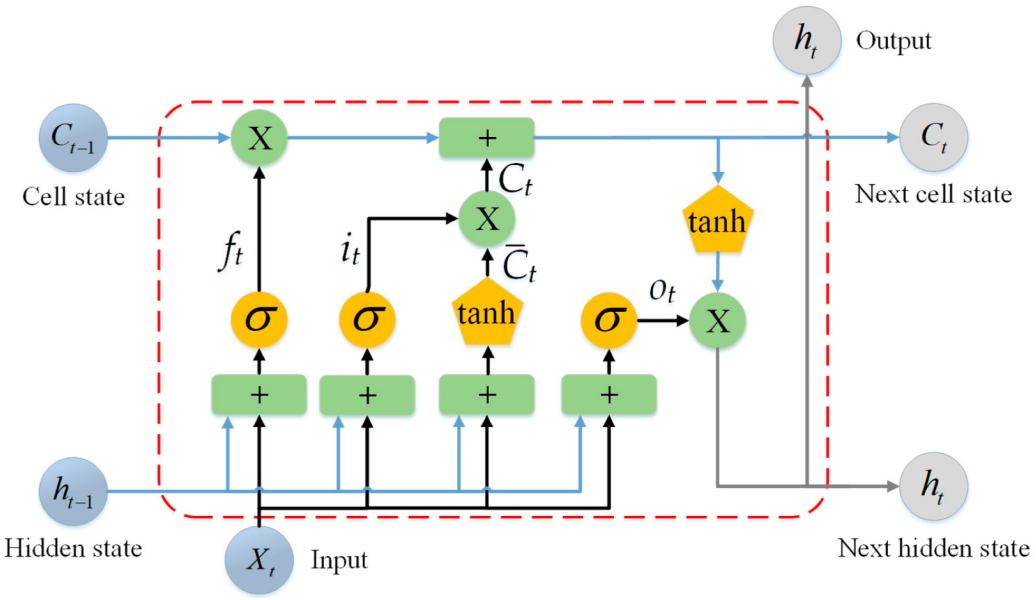


Figura 9. Arquitectura de una célula LSTM. Las entradas corresponden a la palabra actual a evaluar, y los estados previos de la célula, para así producir lo que serán las entradas en la próxima iteración como es el estado de la célula y el próximo estado oculto que coincide con la salida final de predicción de la palabra evaluada.

Dichas compuertas controlan la medida de entrada, salida y olvido respectivamente según:

$$i_t = \sigma(W_{xi} \cdot X_t + W_{hi} \cdot h_{t-1} + W_{ci} \cdot c_{t-1} + b_i)$$
$$f_t = \sigma\left(W_{xf} \cdot X_t + W_{hf} \cdot h_{t-1} + W_{cf} \cdot c_{t-1} + b_f\right)$$
$$c_t = f_t \cdot c_{t-|} + i_t \cdot tanh\left(W_{xc} \cdot x_t + W_{hc} \cdot h_{t-1} + b_c\right)$$
$$o_t = \sigma\left(W_{xo} \cdot x_t + W_{ho} \cdot h_{t-1} + W_{co} \cdot c_t + b_o\right)$$

$$h_t = o_t \cdot tanh(c_t)$$

donde *i*, *f*, *c* y *o* son los vectores de entrada, olvido, celda y salida respectivamente. *σ* es la función sigmoide, *t* es el tiempo o *timestamp*, *x* es el vector que representa cada palabra, *h* es el vector de estado oculto de la celda, *W_* son las matrices con los parámetros de cada capa y *b* son los vectores de bias correspondientes.

Observar como los cálculos de una celda dependen del cálculo de la celda anterior (h_{t-1}) y del vector de la palabra actual que la red está evaluando (x_t).

La función tangente hiperbólica es utilizada como una función de activación introduciendo una no linealidad luego de la multiplicación de la entrada por la matrices de peso decidiendo así si una neurona debe ser activada o no en base a si la información recibida es relevante o debería ser ignorada.

Clasificador CRF

La red CRF es un clasificador en base al contexto, es decir, según los etiquetados vecinos utilizando un modelo de probabilidad donde se representan las dependencias entre los etiquetados.

El modelo utilizado depende de la tarea. En el caso de tareas de procesamiento del lenguaje natural, *"linear chain"* CRF es el más popular donde las predicciones dependen exclusivamente de los vecinos inmediatos de la predicción actual, este mismo es el utilizado en la implementación de arquitectura que utilizaremos.

Dado **X** una secuencia de *l* palabras, esta red busca conocer la probabilidad de que se encuentre una secuencia de clasificación en dicha secuencia de palabras que llamaremos **y**. Así buscamos entonces la probabilidad:

$$\begin{aligned} P(y \mid X) &= \prod_{k=1}^{l} P(y_k \mid X_k) \\ &= \prod_{k=1}^{l} \frac{exp(U(X_k, y_k))}{Z(X_k)} \\ &= \frac{exp(\sum_{k=1}^{l} U(X_k, y_k))}{\prod_{k=1}^{l} Z(X_k)} \end{aligned}$$

Donde U(x, y) es llamado el valor unitario o emisor, que es el valor en el tiempo *k* de la clasificación *y* en la secuencia ***X***. Y donde Z(x) es conocido como una función de partición, que utilizamos para obtener una probabilidad al final del cálculo.

Superando cierto umbral establecido, se clasifica la secuencia como perteneciente a una clase u otra.

Modelo: ELMo-BiLSTM-CNN-CRF

Esta arquitectura propuesta también por Nils Reimers agrega una capa ELMo previa a la arquitectura BiLSTM-CNN-CRF previamente descrita mejorando así el desempeño en diferentes tareas de etiquetado de secuencias al ser nutrido por *word embeddings* ricos en contexto.

Capa ELMo

Un problema que presentan típicamente los *embeddings* de palabras es la falta de contexto en su representación. Por ejemplo:

- *This apple is delicious.*
- *Apple is a very profitable company.*

En la primer oración, esperamos que el *embedding* de *apple* (manzana), se encuentre en el cluster de frutas en su representación vectorial. Pero en la segunda sentencia, queríamos que la representación vectorial de la empresa Apple se encuentre más cercana al cluster de empresas.

Notado por (Peters et al., 2017), siguiendo por la creación de ELMo como una solución a esto en (Peters et al., 2018).

ELMo calcula embeddings de palabras *on the fly*, solucionando así el problema que acarrean los embeddings estáticos.
Esta red utiliza una arquitectura BiLSTM entrenada en el modelado del lenguaje, tarea donde se busca predecir la siguiente palabra en una sentencia.

Para la obtención de los embeddings de palabra, ELMo utiliza los estados ocultos de la arquitectura (y *embeddings* iniciales), produciendo así 3 vectores de 1024 dimensiones.

A partir de estos ellos, (Reimers & Gurevych, 2019) propone un estudio de diversas formas de integrar estos vectores para producir un único vector e introducirlos a la arquitectura BiLSTM-CNN-CRF original que ellos implementaron.

La propuesta original en (Peters et al., 2018) era la multiplicación de estos vectores por pesos dependientes de la tarea, para luego realizar una suma vectorial obteniendo así el *embedding* final. Sin embargo, (Reimers & Gurevych, 2019) demuestra que estos *embeddings* no son fehacientes para ciertas tareas, e incluso decrece el desempeño del modelo final. Además nota que la utilización del vector de la última capa del modelo BiLSTM decrece el desempeño del modelo para la obtención de los *embeddings* sin mejorar de forma significativas a estos.

Así es como se propone en la arquitectura la utilización de las dos primeras capas, obteniendo el vector final a partir de la media de los vectores de dichas capas multiplicadas por un peso correspondiente.

Reduciendo el modelo ELMo de tres capas a solo dos genera una aceleración en el entrenamiento del modelo de hasta un 50%.

Notar así que la utilización de *embeddings* de ELMo son computacionalmente más costosos en recursos como memoria y tiempo que aquellos obtenidos de forma estática de métodos tradicionales como Word2Vec o GloVe.

Modelo de Lenguaje vs. Arquitectura BiLSTM

Un modelo de lenguaje es un modelo *entrenado* en la tarea de predecir la siguiente palabra dada una sucesión de ellas como entrada.

Formalmente, dada una secuencia de palabras *x(1)*, *x(2)*, …, *x(t)*, computa la distribución de probabilidad de la siguiente palabra *x(t+1)*.

Un ejemplo claro de este tipo de modelos está comúnmente presente en la aplicación del teclado de los *smartphones*, que a partir del texto escrito por el usuario aparecen opciones de palabras para completar la oración.

Los modelos de lenguaje presentan una ventaja clara sobre otros tipos de modelos y es la capacidad de poder ser *entrenados*. Para entrenar un modelo existen diversas técnicas, entre ellas utilizando N-gramas, esto es desplazar una ventana de N palabras sobre el corpus de entrenamiento para generar así *(virtualmente)* un dataset sobre el cual el modelo puede computar predicciones y ajustar sus parámetros.

El ajuste de parámetros se realiza en cada iteración de entrenamiento, dada un N-grama del corpus, remover la *(N-1)-ésima* palabra e insertarla como entrada al modelo para que este realice la predicción. Dicha predicción del modelo es un vector con todas las palabras conocidas por este con una probabilidad, comúnmente el modelo retorna aquella palabra con mayor probabilidad pero no en el caso de entrenamiento, aquí nos interesa el vector completo.

Dado que se conoce la palabra que debía predecir, se genera un vector con las mismas palabras que conoce el modelo de tal forma que todos los valores son cero, excepto la palabra correcta a predecir la cual vale 1. Se realiza una resta vectorial del vector con predicción correcta y el vector de predicción del modelo, generando así un vector de error el cual se utiliza para realizar el ajuste de los parámetros.

Al contarse con una secuencia finita de palabras, se puede explotar el desempeño de un modelo de lenguaje utilizando un modelo de lenguaje bidireccional. Esto permite al modelo predecir la palabra siguiente, o la palabra anterior. BERT de (Devlin et al., 2019) es un ejemplo de modelo bidireccional.

Una vez contado con el modelo base pre entrenado, este puede ser *finetuneado* para una tarea específica. Generalmente los modelos de lenguaje están entrenados sobre un corpus genérico como son páginas web, entradas de Wikipedia, etc., este entrenamiento puede, como describimos previamente, profundizarse utilizando un corpus o ejemplos específicos a la finalidad del objetivo para así mejorar sus predicciones en dicha tarea.

Esta es una de las claves del porque podemos esperar de un modelo de lenguaje un mayor desempeño que, en este caso, una arquitectura BiLSTM-CNN-CRF frente a un dataset pequeño. El hecho que un modelo de lenguaje fuera pre entrenado crea un baseline claro sobre el cual *finetunear* con el dataset específico, lo cual se espera que mejore el rendimiento del mismo.

La arquitectura LSTM era el estado del arte entre 2013-2015, pero se vio desplazada por la aparición de los modelos basados en Transformers. Por ejemplo, Bert ha demostrado tener un mejor desempeño frente a una arquitectura BiLSTM incluso en casos donde se integra la capa CNN para los embeddings de caracteres, lo que es capaz de extraer features de calidad que enriquece a la arquitectura y que junto a la capa final CRF favorece aún más su desempeño al incorporar más información para la tarea de etiquetado.

De cualquier forma, en el paper de (Eger et al., 2017), se realizaron diversos experimentos sobre la arquitectura BiLSTM-CNN-CRF con diversos hiperparametros concluyendo que la utilización de la capa CRF al momento de clasificar no mejora los resultados de forma notoria.

(Ezen-Can, 2020) realiza una comparación entre BERT y una red LSTM utilizando un corpus pequeño llegando a la conclusión que LSTM presenta un desempeño significativamente mayor que BERT, debido a que este último sobreajusta (*overfits*) sobre datasets pequeños.

Dataset

El conjunto de ejemplos para la realización de este trabajo viene de la mano de (Furman et al., 2022).

Una problemática actual que enfrentan los modelos en el área de discurso de odio, y sobre todo la generación de contranarrativas, es la escasez de ejemplos en los cual basarse para su entrenamiento como observamos en los trabajos relacionados. Dicha escasez tiene muchas fuentes como son; el coste del anotado de ejemplos, falta de expertos en el área para realizar las anotaciones o incluso la poca iniciativa socio-económica que tiene el área frente a otras investigaciones en computación.

Frente a esto, (Furman et al., 2022) proponen una alternativa para mitigar el problema para la generación de contranarrativas, la cual se basa en agregar información valiosa en los etiquetados manuales, mejorando así la *calidad* de las anotaciones y permitiendo que los modelos puedan guiarse mejor bajo esta condición de falta de ejemplos suficientes para su redacción.

Los ejemplos se tratan de tweets en inglés y español obtenidos de la red social Twitter donde los primeros fueron el resultado de realizar una filtración del dataset original del HateEval 2019, (Basile et al., 2019).

Este filtrado excluye tweets que consistian en gran parte por lenguaje ofensivo (i.e. insultos, exhortaciones, etc.) considerando esto como no argumentativo. También se excluyeron todos los tweets en contra de las mujeres viendo que muchos caian en esta categoría, generando un desbalance de ejemplos, y luego se removieron aquellos que su objetivo era un único individuo, dejando así aquellos tweets en contra de colectivos que en gran parte constituyen minorías en la sociedad.

Sobre estos ejemplos se realizaron las siguientes anotaciones:

- Si es no argumentativo, en cuyo caso no se prosigue con las siguientes anotaciones.
- Identificar las componentes argumentativas que se tratan de la justificación y conclusión, y diferenciar si se tratan de hecho (cuando una de estas componentes es tratada como un hecho verídico), político (cuando se manda a tomar acción) o valor (cuando se trate de una opinión).
- Identificar el colectivo objetivo del tweet, y la
- propiedad (explícita) que se le asigna al objetivo del ejemplo.
- Identificar también el elemento que transfiere sentido de una justificación a una conclusión. Dicho elemento es el que llamamos pivote que en ciertas ocasiones puede ser implícito, en cuyo caso se lo anota.
- Realizar la redacción de una contra narrativa para el tweet.

Los anotadores consistieron en total de tres personas, bajo las cuales se midió el acuerdo por categoría utilizando el método de κ de (Cohen, 1960) luego de anotar 80 ejemplos del corpus inglés. Esto arrojó un κ cercano a 0.7, lo que implica una alta reproducibilidad, para las componentes de *Colectivo*, *Justificación*, *Conclusión*, *Tipo de Conclusión* y deserción entre ejemplos argumentativos y no argumentativos. Por otro lado, las categorías de *Pivote*

y *Propiedad* con un κ cercano a 0.6 presenta un menor acuerdo entre los anotadores siendo estas categorías más difíciles de identificar en los ejemplos, y finalmente el tipo de justificación tuvo el κ más bajo con valor de 0.05, básicamente no hubo casi acuerdo entre los anotadores. Esto se justifica del hecho que las justificación del tipo Valor son difícilmente diferenciables de aquellas del tipo de Hecho.

Hipótesis sobre el comportamiento esperado de los modelos a partir de las propiedades estadísticas del dataset

Encontramos una descripción detallada del dataset en el *Anexo I: Análisis del dataset*, este es más bien pequeño y muchas de las anotaciones son escasas y con gran variabilidad, lo cual nos hace presumir que los predictivos que podamos inferir de él no tendrán un muy buen rendimiento. Sin embargo, hemos observado algunas propiedades de los datos que nos permiten esperar rendimientos mejores en algunos de los aspectos argumentativos a predecir.

Conociendo la arquitectura descrita junto al análisis del dataset, podemos hipotetizar que la gran diferencia en cantidad de ejemplos entre argumentativos y los que no lo son causará un bajo desempeño.

En cuanto a las diferentes categorías, vemos cerca de un 45% de los ejemplos totales contienen un colectivo por lo que la arquitectura al tener un balance entre ellos podrá aprender y generalizar de forma correcta. Eso si, por el mismo tipo de arquitectura, el modelo aprende una componente a la vez por lo que en casos como la presencia de *Colectivo* que viene siempre acompañada de una *Propiedad*, no esperamos que se cumpla, es decir, hipotetizamos que el modelo predecirá el colectivo en muchos tweets incluso aquellos no argumentativos o sin presencia de *Propiedad*. Además, esta categoría en gran mayoría se comprende por una única palabra por lo que el modelo puede explotar este aspecto, quizá la capa “linear chain” CRF pueda aprovechar el etiquetado vecino para aprender a generalizar a partir de los demás ejemplos con mayor cantidad de colectivos.

En cuanto a *Propiedad* y *Pivote*, a diferencia del *Colectivo*, las palabras repetidas que se anotaron con ellas presentan una baja cantidad, sumado a no tener otro tipo de información o forma de brindarse al modelo nativo, la capacidad resultante de los mismos para generalizar las secuencias de palabras que componen estas categorías es, hipotetizamos, muy baja.

La categoría de *Justificación* es particular, al igual que ocurre con la *Conclusión*, se encuentran en más del 72% del total de ejemplos, abarcando entre ambas casi la totalidad del tweet. Vimos en el análisis que la *Justificación* tiende a encontrarse al principio mientras que la *Conclusión* al final, algo esperado por cómo las personas aprenden a expresar sus ideas de forma textual, aunque no es del todo menor cantidad los casos que ocurren a la inversa. Es por ello que pensamos que los modelos al no poder explotar palabras claves que distinguen estas categorías de forma clara, aprovecharán la posición y largo del tweet, por como también vimos los tweets no argumentativos contienen menor cantidad de palabras que los argumentativos, para acertar en sus predicciones. Además puede que aquellos modelos que utilicen la capa CRF vean un mejor desempeño por la misma junto al

hecho de que estas categorías, sobre todo *Justificación*, suele componerse de una media alta de palabras; inclusive llegar a ver casos donde el modelo sobrepasa la cantidad de palabras a anotar.

Para la predicción de *Tipo de Justificación* y *Conclusión*, vimos una clase mayoritaria clara en ambas, de *Hecho* para la primera y de *Política* para la segunda así que podemos hipotetizar que los modelos tendrán una tendencia ante ellas sobre los demás tipos.

Es importante destacar que el tipo, ya sea de *Justificación* o *Conclusión*, acompaña en lo posible a dicha categoría, es por ello que su presencia en los tweets es menor al 72% mencionado previamente, pero mayor al 60% por lo que vimos en el análisis. Esto nos da pie a entender que la tarea de clasificación de tipos de cierta forma esconde la tarea de predicción de *Justificación* o *Conclusión* lo que resulta en modelos que exploten las características que mencionamos eran de interés para estas categorías, debido a esto no esperamos un mejor desempeño que estas mismas tareas de predicción pero si uno menor por este trabajo adicional y no binario de clasificacion.

Esperamos que exista un único tipo de componente por componente misma, siguiendo así las pautas de los anotadores en los ejemplos, es decir, que el modelo no mezcle tipos por unidad de componente, ya sea *Justificación* o *Conclusión*, como podría ser mezclar palabras como tipo de *Hecho* y otras de tipo de *Valor* cuando ambos conjuntos pertenecen a la misma componente del tweet.

Formato de anotación: Brat standoff y CoNLL

Un trabajo intermedio que se debió realizar se trata de convertir las anotaciones realizadas por los anotadores en un servidor Brat a un formato basado en CoNLL 2009. Este último formato se compone por líneas donde cada una representa una única palabra con una serie de valores separados por tabulador (*TAB*), una línea vacía representa el final de la sentencia.

El formato de anotaciones hechas en Brat, denominado Brat-Standoff, se compone de dos archivos en total por tweet, uno de texto plano (.txt) y el de anotaciones (.ann). El .txt contiene el tweet normal, como si un copia y pega de Twitter se tratara.

El .ann, por su lado, está formado por líneas donde cada una se compone por una anotación particular, esta puede tratarse de la anotación de una entidad (en el caso del dataset esto se trata de *Colectivo*, *Propiedad*, *Justificación*, etc.), atributo (*Tipo de Justificación* o *Tipo de Conclusión*), relación, evento o disparador de evento, junto al rango de caracteres que esta anotación abarca en el .txt y el fragmento específico del .txt del que se trata.

Se encontraron diferentes scripts en Github implementados en Python para la conversión del formato Brat-Standoff a CoNLL, pero debido a que el formato objetivo *se basa* en CoNLL debimos optar por una implementación manual para satisfacer nuestros requerimientos y el de la arquitectura.

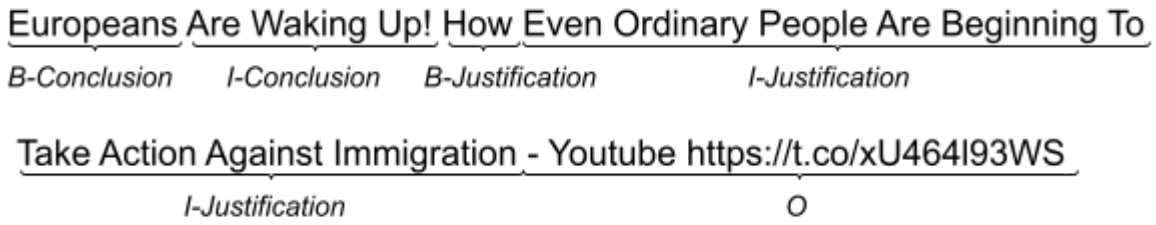
Esta implementación además de utilizar una codificación BIO o IO para las categorías, posteriormente detallada, también realiza como es de esperarse la separación de las palabras que componen el tweet. En este aspecto se tomaron consideraciones especiales para realizar tal separación, es específico las palabras se dividieron no solo por espacio, sino también por signos de puntuación (*“.”*, *“,”*, *“‘”*), de exclamación (*“!”*) e interrogación (*“?”*).

Codificación: BIO e IO

Para codificar las anotaciones de las palabras como pertenecientes a una categoría, utilizamos en este trabajo tanto un formato BIO como formato IO. Detallamos el tipo de codificación en cada experimento realizado.

Ambos formatos tienen como objetivo clasificar sentencias, es decir, un conjunto de palabras seguidas como pertenecientes a la misma categoría.

En el formato BIO, la palabra cabecera que comienza una sentencia como perteneciente a la categoría *X* se anota, en la forma que corresponda, como *B-X*, las palabras sucesivas en la misma sentencia como *I-X*.
Aquellas palabras en el ejemplo que no pertenecen a ninguna categoría, se las anota con una *O*. En el próximo ejemplo vemos cómo se aplica el formato BIO con las categorías *Conclusion* y *Justification* en la anotación de un tweet del dataset utilizado:



El formato IO es similar al anterior, solo que omite diferenciar la palabra cabecera de la sentencia anotando indistintamente con *I-X*.
Este formato produce así una disminución en el espacio de categorías, se obtiene una anotación binaria de la palabra; pertenece a la categoría *I-X* o a la categoría *O*, pero se pierde información sobre las palabras con las que suele comenzar la misma en una secuencia.

Para ejemplificar al lector, dejamos de ejemplo en el *Anexo Ⅱ: Ejemplo de formato de los tweets*, lo que compone un tweet del dataset en el formato Brat standoff y su posterior conversión a un formato basado en CoNLL con codificación IO que sirve como entrada para la arquitectura con las anotaciones de interés, además en el formato basado en CoNLL se añade como cabecera la componente de cada columna.

Experimentos

Como detallamos en la sección anterior, la arquitectura utilizada presenta un amplio abanico de configuraciones de hiper parámetros posibles, además de la posibilidad incluso de extender la arquitectura misma.

Detallamos en esta sección los diferentes experimentos llevados a cabo junto a los parámetros específicos de la arquitectura para cada uno de ellos. Revisar la posterior selección para un análisis de resultados.

Experimento N° 1: *BIO*

Con el objetivo de conocer qué configuraciones de hiper parámetros se adecuan en sentido de maximizar la métrica macro-F1, próximamente descrita junto a otras de interés, se realizó el entrenamiento de varios modelos alternando los siguientes parámetros:

- *Red recurrente*: se exploraron la utilización de una sola red BiLSTM de tamaño 100 ([100]) o una concatenación de dos redes de tamaño 100 cada una ([100, 100]).
- *Capa de clasificación*: se exploraron la utilización de una *"linear chain"* CRF detallada en la arquitecura, utilizar la función Softmax, común en tareas de clasificación que asigna a cada clase una probabilidad dado el vector de salida de la red, y la utilización de la función lineal (*None*) también conocida como función de no-activación, pues retorna el mismo valor que toma como entrada.
- *Word Embeddings*: en esta primera instancia de experimento, exploramos con la utilización de los word embeddings de Komninos presente en la arquitectura de base, y los embeddings de un modelo FastText entrenado con entradas de la Wikipedia.
- *Char Embeddings*: se exploraron los char embeddings resultantes de una red 1D-CNN, una red LSTM y la no utilización de ellos (*None*), para valorar su utilidad de forma crítica ante no contar con ellos.

Los siguientes parámetros se mantienen estáticos en todos los modelos en esta primer instancia de experimentos:

- *Dropout*: se utilizó un dropout variacional de 0.25 de dropout en salida y 0.25 de dropout recurrente.
- Optimizador: Adam.
- Mini-Batch size: número de sentencias para entrenamiento en mini baches de 32
- Epochs de entrenamiento: 25.
- *Formato de entrada: BIO.*

Debido a la gran cantidad de modelos resultantes a entrenar, redujimos las categorías a predecir a tan solo el *Colectivo* y *Justificación-Conclusión-No Argumentativo* (*JCnA*). En posteriores experimentos separaremos estas clases por sus componentes.

Experimento con Pesos de CRF

A raíz de la examinación exhaustiva de las predicciones de los modelos del experimento anterior, concluimos que nos era de interés explorar el seteo de pesos iniciales de la capa

de la *"linear chain"* CRF, pues notamos ejemplos donde ésta no fue capaz de seguir el formato *BIO* en sus predicciones.

Para ello, exploramos diversos modelos para la clasificación de la propiedad *Colectivo* y *JCnA* donde seteamos los pesos iniciales de la capa *"linear chain"* CRF para cada clase de la propiedad a predecir (e.g. *O*, *B-collective*, *I-collective* para el caso de la propiedad *Colectivo*) de las siguientes formas:

- Pesos iniciales donde en todas las clases es igual a 0.
- Pesos iniciales donde en todas las clases es igual a 1.
- Pesos iniciales donde una clase presenta un peso inicial de 2 y los demás 1.

De esta forma obtuvimos 5 configuraciones para la propiedad *Colectivo* y 9 para el caso de *JCnA*, que se duplican considerando los modelos que utilizan la capa ELMo.

Los siguientes parámetros se mantuvieron estáticos en todos los modelos:

- *Word Embeddings*: se utilizó únicamente los embeddings de Komninos por su buen desempeño en la generalidad de los modelos entrenados en el experimento anterior.
- *Char Embeddings*: se utilizaron a partir de los obtenidos por el método 1D-CNN pues presentaban buen desempeño con estas configuraciones revisando la tabla de métricas.
- *Red recurrente*: una única red LSTM de tamaño 100.
- *Dropout:* se utilizó un dropout variacional de 0.25 de dropout en salida y 0.25 de dropout recurrente.
- *Optimizador:* Adam.
- *Mini-Batch size:* número de sentencias para entrenamiento en mini baches de 32
- *Epochs de entrenamiento:* 25.
- *Formato de entrada: BIO.*

Los resultados se presentan en la Tabla 2, pero observamos que ningún modelo fue capaz de sobrepasar el F1 de la clase minoritaria de forma satisfacible frente al seteo aleatorio de los pesos. Observamos además en las predicciones, que los modelos no fueron capaces de preservar el formato *BIO* motivo por el cual planteamos este experimento inicialmente. Es por ello que descartamos setear de forma manual los pesos de la capa CRF en los próximos entrenamientos dejando así que se seteen de forma aleatoria.

Vector de peso inicial	**Colectivo**			**JCnA**		
	Pr	Rec	F1	Pr	Rec	F1
Aleatorio	0,40	0,45	0,42	0,62	0,53	0,57
[0, 0, 0]	0,38	0,22	0,28	-	-	-
[1, 1, 1]	0,40	0,26	0,32	-	-	-
[1, 1, 2]	0,32	0,26	0,29	-	-	-
[1, 2, 1]	0,29	0,17	0,22	-	-	-
[2, 1, 1]	0,38	0,22	0,28	-	-	-
[0, 0, 0, 0, 0, 0, 0]	-	-	-	0,61	0,43	0,51
[1, 1, 1, 1, 1, 1, 1]	-	-	-	0,66	0,59	0,62

[1, 1, 1, 1, 1, 1, 2]	-	-	-	0,64	0,57	0,61
[1, 1, 1, 1, 1, 2, 1]	-	-	-	0,57	0,58	0,57
[1, 1, 1, 1, 2, 1, 1]	-	-	-	0,82	0,48	0,61
[1, 1, 1, 2, 1, 1, 1]	-	-	-	0,62	0,63	0,62
[1, 1, 2, 1, 1, 1, 1]	-	-	-	0,55	0,59	0,57
[1, 2, 1, 1, 1, 1, 1]	-	-	-	0,56	0,55	0,56
[2, 1, 1, 1, 1, 1, 1]	-	-	-	0,73	0,57	0,64

Tabla 2. Resultados al aplicar diferentes inicializaciones de pesos a la capa "linear chain" CRF. En el caso del Colectivo no se observa una mejoría. En JCnA vemos un par de casos donde ciertas inicializaciones superan a la inicialización aleatoria, pero no por mucho como para motivarnos a setear individualmente los pesos de las categorías en los experimentos.

Experimento FastText vs *Twitter* FastText

El corpus utilizado, como se mencionó en la descripción del Dataset, fue generado a partir de tweets publicados en la red social Twitter. Debido a la naturaleza del mismo, las expresiones gramaticales y sintácticas difieren en una gran medida de canales de comunicación genéricos como pueden ser; cartas, charlas/reuniones o publicaciones.

Es debido a ello que se presenta la utilización de word embeddings obtenidos a partir de un modelo FastText pre-entrenado en Twitter; donde el objetivo del mismo es poder adimensionalizar las palabras y su semántica en el contexto de uso en la aplicación.

El modelo en cuestión fue obtenido a través del entrenamiento con 400 millones de tweets (lo equivalente al 1% de los tweets anglosajones a lo largo de un año). Presentando resultados *en el estado del arte* en dos tareas de *Part-Of-Speech tagging* utilizando datasets de *Ritter* y *Gimpel*.

Para evaluar su utilización en pos del modelo genérico de FastText entrenado con ejemplos de Wikipedia que utilizamos en los ejemplos anteriores, entrenamos 4 modelos por categoría a predecir utilizando:

- *Red recurrente*: una única red BiLSTM de tamaño 100
- Capa de clasificación: *"linear chain"* CRF
- *Word Embeddings*: obtenidos a través del modelo FastText entrenado en Wikipedia, y aquellos obtenidos a través del modelo entrenado con ejemplos de Twitter (*Twitter FastText*). Además de aquellos obtenidos por el modelo ELMo utilizando estos embeddings como entrada inicial de la red.
- *Char Embeddings*: *None*, pues queremos conocer el impacto puro que presenta los *word embeddings* por si solos de los modelos FastText

conservando además los parámetros estáticos igual al experimento anterior.

A raíz de esto, se obtuvieron los resultados tabulados en Tabla 3 donde vemos que los embeddings de Twitter FastText obtuvieron una mayor F1 de clase minoritaria en tres de las cuatro tareas. De cualquier forma, solo vemos una gran diferencia en las categorías de

Colectivo y Propiedad, mientras que en JCnA y Pivote la diferencia entre uno u otro es de tan solo aproximadamente 0.03 puntos.

Embeddings	JCnA			Colectivo			Propiedad			Pivote		
	Pr	Rec	F1	Pr	Rec	F1	Pr	Rec	F1	Pr	Rec	F1
FastText	0,72	0,45	0,55	0,50	0,33	0,40	0,57	0,08	0,15	0,28	0,12	**0,17**
Twitter FastText	0,65	0,66	**0,65**	0,60	0,36	**0,45**	0,49	0,16	0,24	0,21	0,11	0,14
ELMo + FastText	0,63	0,66	0,64	0,50	0,30	0,38	0,46	0,17	0,25	0,23	0,12	0,16
ELMo + Twitter FastText	0,64	0,54	0,59	0,52	0,35	0,42	0,50	0,21	**0,30**	0,17	0,11	0,13

Tabla 3. Resultados de utilizar embeddings obtenidos de FasText generico entrando en Wikipedia y FastText entrenado con tweets de Twitter. En JCnA, Colectivo y Propiedad este último obtuvo el mejor desempeño por cerca de 0.05 puntos. En el pivote, FastText generico obtuvo mejor desempeño pero aquel entrenado con tweets obtuvo tan solo 0.01 puntos menos, lo cual nos motiva a utilizar FastText entrenado en twitter a partir de este momento. Por tratar mejor estas palabras con mucho ruido, algo muy natural en redes sociales.

Bajo estos resultados, consideramos entonces que FastText entrenado con ejemplos de Twitter brinda un mejor rendimiento que FastText genérico. Es así que optamos por utilizar este modelo en los próximos experimentos.

Experimento N° 2: *IO*

Al concluir el experimento anterior para conocer el desarrollo de la arquitectura bajo diversas configuraciones de hiperparametros, se prosiguió a entrenar modelos para todas las clases de propiedades (*Colectivo*, *Propiedad*, *Pivote*, *Justificacion*, *Conclusion* y *No Argumentativo*) bajo los siguientes hiperparametros en busca de mejorar el desempeño obtenido anteriormente donde además, migramos a un formato *IO* en el etiquetado de las clases de entrada a la red por las conclusiones observadas:

- *Red recurrente:* una sola red BiLSTM de tamaño 100.
- *Capa de clasificación*: utilización de una *"linear chain"* CRF y Softmax.
- *Word Embeddings*: en esta segunda instancia de pruebas, se utilizaron embeddings de Komninos y, embeddings de FastText a partir de un modelo pre-entrenado de Twitter. También se incluyeron sus contrapartes utilizando estos embeddings para el entrenamiento del modelo ELMo, para obtener los embeddings ricos en contexto.
- *Char Embeddings:* se exploraron los char embeddings de 1D-CNN, una red LSTM y la no utilización de ellos (None).

Se mantuvieron los mismos parámetros estáticos igual que en el experimento anterior:

- *Dropout:* se utilizó un dropout variacional de 0.25 de dropout en salida y 0.25 de dropout recurrente.
- *Optimizador:* Adam.
- *Mini-Batch size:* número de sentencias para entrenamiento en mini baches de 32
- *Epochs de entrenamiento:* 25.
- *Formato de entrada: IO.*

Experimento Tipos de Justificación y Conclusión

A raíz del experimento anterior, se decidió entrenar modelos para la predicción de los tipos de justificación y conclusión. Recordando al lector, las justificaciones y conclusiones fueron, en lo posible, anotadas como de *Hecho* si se lo trataba como verídico, de *Valor* si se trataba de una opinión subjetiva, o de *Política* si se buscaba que se genere una llamada a la acción.

Notar además, cómo a diferencia de la mayoría de las categorías anteriores (exceptuando JCnA del primer experimento) a predecir, esta no se trata de una clasificación binaria; es por ello que hipotetizamos un desempeño bajo del modelo final.

Los parámetros de la arquitectura se componen en su mayoría estáticos por lo visto en otros experimento que maximizan sistemáticamente el macro-F1 de los modelos anteriores, modificando únicamente la componente de *Word Embeddings* donde se utilizó aquellos de Komninos y Twitter FastText, y la capa de clasificación entre la *"linear chain"* CRF y Softmax. Los demás:

- *Red recurrente:* una sola red BiLSTM de tamaño 100.
- *Char Embeddings: None*.
- *Dropout:* se utilizó un dropout variacional de 0.25 de dropout en salida y 0.25 de dropout recurrente.
- *Optimizador:* Adam.
- *Mini-Batch size:* número de sentencias para entrenamiento en mini baches de 32
- *Epochs de entrenamiento:* 25.
- *Formato de entrada: IO.*

Experimento N° 3: Variaciones en *Dropout*

Concluimos del experimento *IO* anterior observando que muchos modelos terminaban lexicalizados por la relación entre las palabras y clases que se encontraban en los datos de entrenamiento. En base a esto, avanzamos en la evaluación de los modelos aplicando una técnica de *smoothing*, el *dropout*.

Dropout es una técnica donde en base a probabilidades se reduce a 0 el paso de información de una neurona a otra (unidades ocultas para el caso de la red LSTM). De esta forma la red no puede basarse en la presencia de ejemplos concretos en su entrenamiento, siendo así una técnica para evitar el *overfitting*. En otras palabras, de forma aleatoria se desconectan neuronas temporalmente que simulan una pérdida de memoria en la red, como si nunca hubiera visto el ejemplo en concreto. Esto nos permitiria ayudar a los modelos a no basarse en el alto valor de la información mutua puntual (IMP) entre las palabras y la clase al momento de realizar las predicciones.

Hasta el momento, siempre se utilizó un dropout variacional por defecto de la arquitectura que según los desarrolladores de la misma afirman que presenta mejores resultados de cara a los valores aleatorios con el cual se suele inicializar la arquitectura (e.g. pesos de la misma).

El *dropout* variacional se basa en utilizar la misma probabilidad de dropout en todos los nodos. Así, por defecto el dropout variacional de la arquitectura setea un dropout de 0.25 en

todas las salidas y 0.25 entre todas las unidades recurrentes. También existe el *naive dropout* donde todas las probabilidades de *dropout* son seteadas de forma aleatoria en la red.

En este experimento, buscamos comprobar si modificando el *dropout* utilizado, obtenemos mejores resultados en las predicciones, lo que significaria un mejor desempeño de los modelos. Para ello examinamos las siguientes configuraciones de dropout en la arquitectura:

- No utilizar *dropout*.
- Utilizar un *naive dropout* inicializado en 0.5.
- Utilizar un *naive dropout* inicializado en 0.75.
- Utilizar un *dropout* variacional con 0.5 de dropout en salida y 0.5 de dropout entre nodos recurrentes.
- Utilizar un *dropout* variacional con 0.5 de dropout en salida y 0.7 de dropout entre nodos recurrentes.

Se entrenaron con estas variaciones de *dropout* la mejor arquitectura, según macro-F1, con y sin capa de ELMo para cada una de las categorías.

Las siguientes configuraciones se mantuvieron estáticas en todos los entrenamientos:

- *Optimizador:* Adam.
- *Mini-Batch size:* número de sentencias para entrenamiento en mini baches de 32
- *Epochs de entrenamiento:* 25.
- *Formato de entrada: IO.*

Los resultados de este experimento no fueron del todo alentadores para modificar el *dropout* utilizado de base, inclusive vimos casos donde el *overfitting* de los modelos era mucho mayor, pero que en ciertos casos generaba un leve mejor desempeño. Un análisis detallado de esto se encuentra en el *Anexo Ⅲ: Exploración de valores de dropout*. Debido a estos resultados, seguiremos aplicando el *dropout* por defecto que implementa la arquitectura.

Experimento N° 4: Multitask

Previa a la utilización de la técnica en Cascada que se decidió probar en el análisis de resultados del experimento anterior, exploramos una técnica alternativa, nativa de la implementación de la arquitectura utilizada, el Multitask.

Esta técnica mejora la generalización de los modelos, lo que se logra a partir del entrenamiento en paralelo de diferentes tareas explotando similitudes y diferencias entre las mismas bajo una representación de la información en común.

Para este experimento, examinaremos los siguientes combinaciones de Multitask:

- *Propiedad* y *Colectivo*, (Furman et al., 2022) muestra que cuando el modelo conoce de la presencia de Propiedad, no se generan errores de predicción de *Colectivo*, en los modelos basados en BERT, con tweets donde la *Propiedad* no está presente. Nos es de interés evaluar cómo la red desempeña estas tareas fuertemente relacionadas y dependientes entre sí.

- *Justificación* y *Conclusión*, esperaríamos que estas predicciones sean más acertadas y no se solapen como podia suceder cuando el modelo las aprendía de forma individual.
- *No Argumentativo*, *Justificación* y *Conclusión*, similar a lo anterior hipotetizamos que no se deberían producir errores de clasificación de *Justificación* o *Conclusión* en tweets predichos como *No Argumentativos*.
- *Justificación*, *Conclusión* y *Pivote*, este último es el nexo entre las dos primeras siendo la componente que menos acuerdo obtuvo entre los anotadores y peor desempeño en general hasta el momento.
- *No Argumentativo*, *Justificación*, *Conclusión*, *Colectivo*, *Propiedad* y *Pivote*, esto es lo más semejante a la tarea completa del anotador manual de un tweet.

Para el entrenamiento de esta nueva arquitectura utilizamos los mismos parámetros estáticos como en los anteriores experimentos.

A destacar utilizamos los *Word Embeddings* de Komninos pues al momento arrojaron, sistemáticamente, los mejores resultados en seis de las ocho tareas de clasificación, así como también la utilización de embeddings calculados *on the fly* ricos en contexto con el modelo ELMo inicializados con dichos *embeddings*.

Además, nuestra arquitectura nos permite utilizar la capa de clasificación que deseamos para cada tarea, es por ello que para la de predicción de *Colectivo*, *Justificación*, *No Argumentativo* y *Propiedad* utilizamos la *"linear chain"* CRF, mientras que para la predicción de *Conclusión* y *Pivote* utilizamos la capa Softmax, pues con estas configuraciones obtuvimos los mejores resultados, como se pueden ver en el análisis del experimento *IO*.

A raíz de ello, se observaron como ciertas tareas influyen entre ellas, pero en ningún caso observamos una mejoría a los resultados de los experimentos previos, es por ello que dejamos un análisis detallado en el *Anexo Ⅲ: Predicción multiobjetivo (Multitask)*, sección 3: Predicción multiobjetivo.

La falta de ejemplos, y complejidad mismas de estas tareas, no permiten a los modelos entrenados explotar esta representación intermedia que comparten, observando cómo en ciertos casos, como entre *Propiedad* y *Colectivo*, el primero presenta una mínima mejora, pero el segundo termina con una gran caída en desempeño. Es por ello que descartamos esta arquitectura, hasta contar con mayor cantidad de ejemplos.

Experimento N° 5: Cascada

Vemos que tanto el Multitask como el entrenamiento de modelos para la predicción de categorías de forma individual presentan una alta complejidad en sus salidas que concluyen en un bajo desempeño final del modelo resultante.

Ante esto se nos presentan 3 caminos claros que podemos tomar; el primero de ellos comprende la obtención de más ejemplos de entrenamiento para que los modelos tengan mayores probabilidades de aprender y generalizar sus predicciones de forma correcta pero por el momento no es viable por el costo que esta tarea conlleva debido a la necesidad de anotadores expertos manuales, además del tiempo que se dispone, la segunda opción es la

búsqueda y utilización de una arquitectura más *adecuada* para este tipo de tareas, algo que de momento desconocemos, y finalmente nuestra ultima opcion es modificar ligeramente la arquitectura actual para realizar la técnica de entrenamiento *en Cascada*.

La técnica en Cascada consiste en la concatenación de clasificadores para explotar las representaciones simbólicas que estos manejan de tal forma de que al momento de realizar una predicción en una capa, contar con más información (las predicciones) obtenida de las capas anteriores.

Como mencionamos en el análisis de resultados de las categorías de tipos, (Furman et al., 2022) demuestra que cuando su modelo de lenguaje es conocedor de la presencia de *Justificación* o *Conclusión* en el tweet, su clasificación de tipos mejora en cerca de 0.2 puntos la métrica macro-F1, es por ello que bajo esta técnica intentaremos reproducir este comportamiento con la arquitectura utilizada.

Para realizarlo se debió implementar manualmente, debido a que no se encontraba de forma nativa la utilización en Cascada en la arquitectura base, lo que consistió en apilar dos BiLSTM que venimos utilizando de tal forma que la salida de la primer red sea la entrada de la segunda junto a los embeddings que toma la primera como su propia entrada.
De esta forma, la segunda red será nutrida no solo con los word embeddings, si no también con las predicciones que realizó la arquitectura anterior para basarse así en sus propias predicciones.

Esta implementación fue a diseño por lo que no podemos generalizar el apilamiento de otras tareas, únicamente nos es posible apilar en cascada las tareas que deseamos explorar cómo son *Justificación* y *Tipo de Justificación* o *Conclusión* y *Tipo de Conclusión*.

Experimentamos tanto con la arquitectura base de BiLSTM como con la que incluye el modelo ELMo al inicio, ELMo-BiLSTM. Como se evidencia en el análisis de resultados del experimento IO, utilizamos *word embeddings* de Komninos pues tuvieron mejor resultado frente a Twitter FastText, esto es importante pues la primer tarea de predicción presentará un alto impacto en el desempeño de la segunda.

Con esta implementación se obtuvieron los siguientes resultados tabulados en la Tabla 4.

Embedding	**Tipo de Justificación**			**Tipo de Conclusión**		
	Pr	Rec	macro-F1	Pr	Rec	macro-F1
Komninos	0,51	0,37	0,37	0,53	0,30	0,29
ELMo	0,58	0,36	**0,38**	0,40	0,34	**0,34**

Tabla 4. Resultados aplicando arquitectura en Cascada. Para el Tipo de Justificación se observa un ligero decremento de desempeño de 0.01 punto comparado al obtenido en el experimento n° 2: IO, mientras que para el Tipo de Conclusión este decremento llega a ser de 0.1 puntos. Se sufre un alto impacto del desempeño final del modelo debido a la dependencia de predicción correcta de Justificación o Conclusión, según el caso, lo que luego complejiza al modelo al clasificar esto según su tipo.

En ella observamos un degradé en la macro-F1 comparando con la obtenida en el experimento *IO* de 0.01 punto para el *Tipo de Justificación* y de 0.1 para el *Tipo de Conclusión*.

Como se mencionó, utilizando Cascada la segunda tarea de predicción que se trata del *Tipo* del componente se ve fuertemente impactada por el desempeño de la primera. En la predicción de *Conclusión*, por ejemplo, debido a que se obtuvo un F1 bajo de 0.58, muchas de las predicciones realizadas fueron erróneas lo que perjudicaron a la segunda arquitectura para clasificar el tipo dichas conclusiones.

Concluimos así notando que la técnica en Cascada no es fehaciente con el desempeño de nuestros mejores modelos.

Métricas de evaluación

Se optó por la utilización de la métrica macro-F1 como métrica principal para valorar el desempeño de clasificación de los modelos entrenados, aunque también se exploraron el detalle de precisión y cobertura.

En determinados experimentos, nos fue de mayor interpretabilidad la métrica F1 de la clase minoritaria presente en el conjunto de clases que señalamos oportunamente en cada momento. Además, en los experimentos referidos a la predicción de *Tipos*, exploramos la matriz de confusión para extraer hipótesis y realizar evaluaciones sobre los desaciertos del modelo.

F1 y macro-F1

La métrica F1 es la media armónica entre precisión y sensibilidad (recall) que detallaremos a continuación, donde el valor máximo es 1, y el valor mínimo 0 solo si dicha precisión o sensibilidad es 0.

Esta métrica es popular para evaluar el desempeño de clasificación de modelos. Se calcula bajo la proxima fórmula:

$$F1 \;=\; 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall}$$

En modelos de clasificación multiclase, como son nuestros modelos resultantes utilizando por ejemplo el formato *BIO*, se opta por calcular un promedio entre las F1 individuales de cada clase de la siguiente forma:

$$macro\ F1 \;= \frac{1}{n} \cdot \sum_{i=1}^{n} F1_i$$, con *n* número de clases

Notar que de esta forma que este macro-F1 definido no tiene en consideración el desbalance entre clases, por lo cual puede verse afectada por presencia de clases predominantes. Esto en general no nos será de mucho problema pues en conjunto evaluaremos, en lo posible, la métrica F1 de la clase minoritaria de interés

Precisión

La métrica de precisión nos permite medir la ***calidad*** del modelo en tareas de clasificación. Para calcularla se utiliza la siguiente fórmula:

$$Precisión \;= \frac{True\ Positive}{True\ Positive + False\ Positive}$$

Viendo el dominador, entendemos que se trata del total de positivos que predijo el modelo calculando así el porcentaje de positivos reales predichos por el mismo.

Esta métrica es de interés cuando el costo de **falsos positivos** es alta, como por ejemplo un testeo de enfermedad que arroja un falso positivo genera un gasto económico importante para el tratamiento de la misma.

Sensibilidad (*Recall*)

La métrica de sensibilidad o exhaustividad, por su parte, nos permite medir la **cantidad** de predicciones correctas que realiza el modelo. Su fórmula:

$$Recall = \frac{True\ Positive}{True\ Positive + False\ Negative}$$

Viendo otra vez el denominador, notamos que se trata de el total real de positivos, calculando así el porcentaje de predicciones correctas positivas que realiza el modelo.

Similarmente a la Precisión, se utiliza cuando el costo de **falsos negativos** es alto, como por ejemplo, cuando el testeo de enfermedad arroja un falso negativo, se descarta la enfermedad y el paciente puede sufrir altas consecuencias en su estado de salud.

Información Mutua Puntual (IMP)

Esta métrica fue altamente utilizada en el análisis de error de los modelos, pues nos permite dar valor a la relación entre una palabra y una categoría. Nos permite poder ordenar la estrecha relación que tienen las diferentes palabras en una misma categoría.

En el caso de este trabajo, podemos por ejemplo calcular el valor entre una palabra y la clase de la propiedad a predecir que deseamos conocer su relación, para saber que tanta dependencia existe entre ellas. Para su cálculo utilizamos la siguiente fórmula:

$$IMP(palabra,\ categoría) = log(\frac{N°\ de\ ocurrencias\ de\ palabra\ en\ categoría}{N°\ de\ ocurrencias\ de\ palabra \cdot N°\ de\ ocurrencias\ de\ categoría})$$

Se utiliza el logaritmo natural pues asumimos que la distribución de las palabras es exponencial. Así esta métrica puede tomar valores tanto negativos como positivos.

Análisis de Resultados

En la Figura 10 se pueden ver los resultados obtenidos para cada categoría por las diferentes aproximaciones exploradas en este trabajo.

Se muestran los resultados obtenidos en el conjunto de tweets de testeo para la configuración de hiper parámetros de cada aproximación que obtuvo los mejores resultados en el conjunto de desarrollo.

Estos resultados se tratan de la F1 de clase minoritaria (macro-F1 para el caso de los *Tipos de Justificación* y *Tipo de Conclusión*), observando también la variación obtenida en 3 ejecuciones del mismo experimento con subconjuntos diferentes del dataset de entrenamiento y subconjuntos diferentes del dataset de testeo, constituidos con el 95% de los ejemplos de test, seleccionados aleatoriamente.

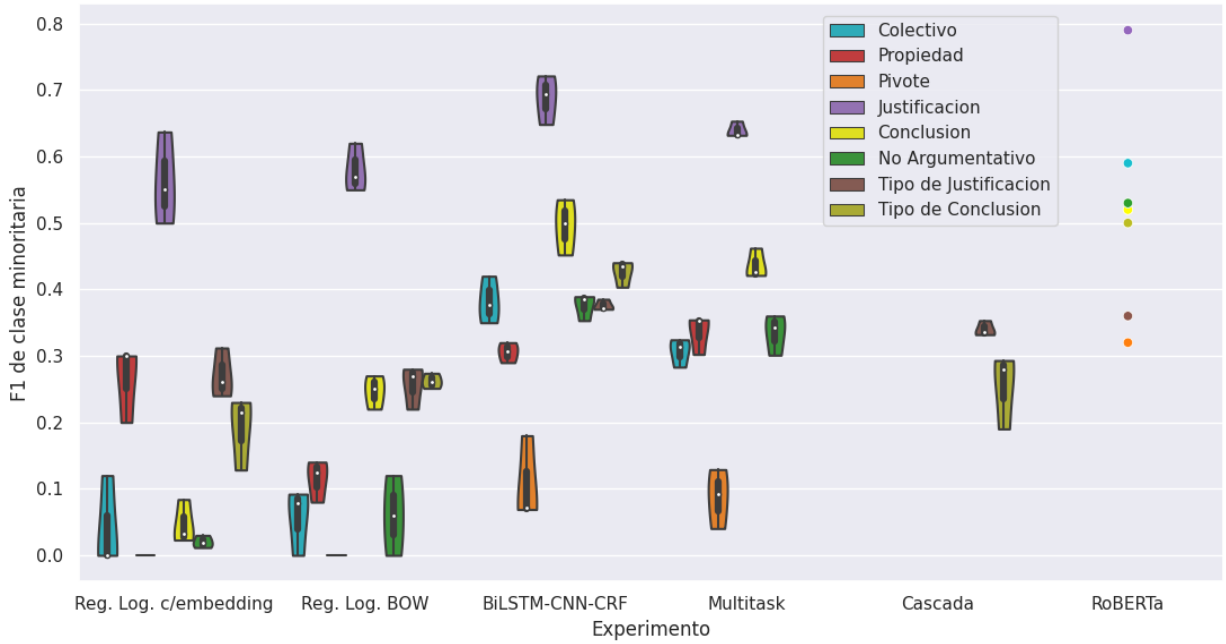


Figura 10. Gráfico de violines para cada arquitectura explorada bajo 3 exploraciones de la partición de testeo. Regresiones logísticas presentan resultados competentes pero con mucha variación. BiLSTM-CNN-CRF, nuestra arquitectura por defecto, presentó mejor desempeño en todas las tareas comparado con demás arquitecturas exceptuando el modelo de lenguaje RoBERTa de la cual solo contamos con un único valor de F1. Multitask presenta un peor desempeño pero con menor variación comparado con la BiLSTM. En Cascada para el Tipo de Conclusión la variación es mucho mayor además de presentar peores desempeños en ambas categorías de clasificación.

El experimento BiLSTM-CNN-CRF refiere a los mejores modelos de arquitecturas que utilizamos para la predicción individual de categorías. RoBERTa fue evaluado en (Furman et al., 2022), del cual solo contamos con un único resultado del desempeño, imposibilitando la generación de violines. Además, en la figura, la categoría de *Propiedad* se encuentra detrás del de *Conclusión*, obstruyendo una visión clara del punto.

Como observamos en la imagen, la tarea de predicción de *Justificación* obtuvo mejor desempeño en todos los modelos llegando inclusive los modelos de regresión logística a

acercarse a los modelos neuronales. Como leve detrimento a este buen desempeño, podemos observar que los valores obtenidos para esta categoría tienen una considerable dispersión, lo cual indica resultados inestables y baja reproducibilidad.

Además observamos que las **regresiones logísticas** presentan mayor desviación es sus desempeños, aunque vemos es menor en el caso del que utiliza BOW, exceptuando la categoría de *No Argumentativo,* la cual presenta la mayor desviación de todos los violines.

Para dimensionar de forma más adecuada la dispersión de valores en las regresiones logísticas, se decidió contar con los violines resultantes de 10 ejecuciones de los experimentos de regresión logística acompañados de su alta velocidad, tal como se muestra en la Figura 11. Para el resto de aproximaciones se mantuvo el número de ejecuciones en 3 para limitar el coste computacional de los experimentos.

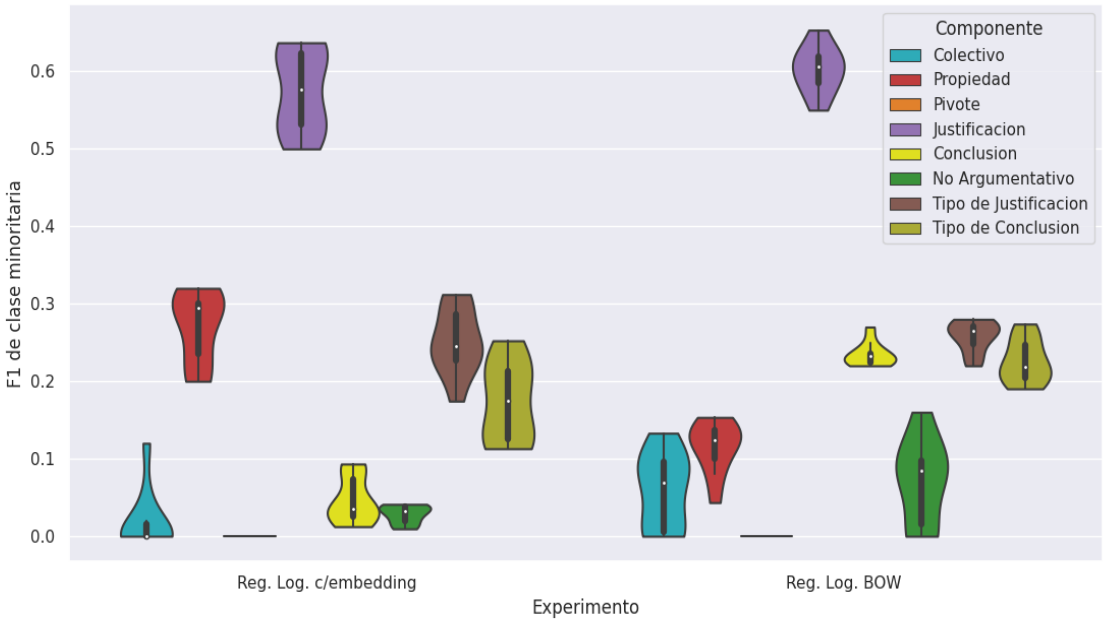


Figura 11. Gráfico de violines para las regresiones logísticas exploradas bajo 10 exploraciones del conjunto de testeo. Ciertas categorías como Justificación, Conclusión, Tipo de Justificación y Tipo de Conclusión, las categorías que más palabras abarcan en un tweet, presentan una menor variación utilizando embeddings a partir de BOW. Además, todas las categorías, excepto Propiedad, obtuvieron mejor desempeño con esta última regresión,

En esta figura observamos una dispersión de valores comparable a los resultados obtenidos con tres ejecuciones. Podemos ver aún más la diferencia y variabilidad de desempeño entre una regresión y la otra frente a la categoría *No Argumentativo*. Notamos como el *Colectivo* presenta una ligera mejor F1 más consistente utilizando BOW que embeddings. La *Conclusión* se beneficia altamente de BOW al igual que el *Tipo de Conclusión* que presentan un mejor desempeño y menor varianza.

La categoría de *Pivote* obtuvo un nulo desempeño en todas las ejecuciones de los experimentos con regresiones logísticas, debido a marcar todas las palabras como *O*. Esto es esperable de un clasificador lineal, siempre más sensible al desbalance de clases, pero también nos demuestra la poca cantidad de ejemplos sumado a la alta complejidad para encontrar un patrón por parte de estos modelos dentro de los pocos existentes.

La *Propiedad* sí ve un decremento de performance utilizando BOW, pero los valores de diferentes ejecuciones parecen estar más pegados entre sí, al igual que ocurre con el *Tipo de Justificación*.

La estrategia **multiobjetivo** también parece disminuir la variación, exceptuando el caso *No Argumentativo* lo que puede deberse por la cantidad de ejemplos de esta categoría existen en el corpus. Una búsqueda de balance entre categoría sería muy beneficioso para los modelos.

La arquitectura utilizada sobrepasa en todas las categorías a los modelos lineales, pero tan solo en las categorías de *No Argumentativo*, *Conclusión* y *Tipo de Justificación* al modelo RoBERTa.

Utilizar la técnica en **cascada** resulta en peor desempeño que utilizando la arquitectura para la predicción individual de categorías, presentando además para el caso de *Tipo de Conclusión* una alta variación de desempeño debido a la dependencia de la primera tarea de predicción de *Conclusión*.

En general, como mencionamos previamente, tan solo la categoría *Justificación* y además la *Conclusión* rebalsaron una métrica F1 mayor a 0.5 en los experimentos que se realizaron en este trabajo, mientras que para el caso de RoBERTa se incluyen además las categorías de *Colectivo*, *No Argumentativo*, *Tipo de Conclusión* y *Propiedad* aunque no contamos con la variación de estos resultados como mencionamos.

Las categorías más complejas como son *Pivote* y *Propiedad*, ven un desempeño casi duplicado cuando sus predicciones provienen de un modelo de lenguaje ante las arquitecturas de aprendizaje profundo. Este modelo presenta así un mejor desempeño en las categorías complejas relacionadas a la semántica, tarea para la cual fueron entrenados primeramente lo que las hace de gran interés para explorar nuevos caminos.

Se pueden observar resultados con más detalle de cada aproximación, con sus respectivas métricas de precisión y *recall*, y también con las métricas estándar de *accuracy* y F1 *macro*, *micro* y *average*, en el *Anexo Ⅲ: Resultados ampliados*. Notando únicamente que los resultados se obtuvieron de la partición de ejemplos de desarrollo, y no del de testeo.

En la próxima sección detallaremos más en profundidad el experimento *BIO* evaluando el desempeño con una vista un poco más detallada junto a un análisis de error que nos motivaron a evaluar los pesos de inicialización de la *"linear chain"* CRF, junto al paso del formato de codificación de clases a *IO*.

Le sigue a esto justamente, un detalle del experimento *IO* evaluando los modelos obtenidos a un nivel genérico para luego ondear a las clases individuales para un análisis de error que dan lugar a los últimos experimentos detallados en la sección anterior. Recordamos que tanto el análisis detallado de la exploración de diversas técnicas y valores de *dropout* como la aplicación de la técnica *Multitask* se encuentran en el *Anexo Ⅲ*. Notamos además, que no realizamos un análisis en detalle de la aplicación de la técnica en Cascada por su mal rendimiento y conclusiones vistas en la sección de su experimento mismo.

BIO

Como mencionamos en la sección de Experimentos, debido a que la cantidad de modelos a entrenar es considerado, únicamente se entrenaron modelos para la predicción de las categorías *Colectivo* y *JCnA*.

Hipotetizamos que esto nos dará un buen panorama acerca de las distintas configuraciones de parámetros que maximizan la métrica macro-F1 para que en próximos experimentos, ya con una menor cantidad de elección de parámetros a determinar, poder ejecutar el entrenamiento de modelos con mayor eficiencia en recursos computacionales y tiempo.

Finalizado el entrenamiento de todos los modelos con las diversas configuraciones, estos fueron tabulados en la próxima Tabla 5 que contiene los datos resumidos:

Parámetros			**Colectivo**			**JCnA**		
Tamaño	Embedding	Clasificador	Pr	Rec	macro-F1	Pr	Rec	macro-F1
100	FastText	Lineal	0,11	0,11	0,11	0,33	0,22	0,09
		CRF	0,48	0,43	0,45	0,57	0,49	0,52
		Softmax	0,36	0,31	0,33	0,54	0,49	0,51
	Komninos	Lineal	0,33	0,33	0,33	0,15	0,12	0,04
		CRF	0,58	0,58	0,58	0,53	0,50	0,50
		Softmax	0,67	0,61	**0,63**	0,55	0,51	0,51
	ELMo	Lineal	0,33	0,35	0,17	0,32	0,20	0,12
		CRF	0,67	0,56	0,60	0,57	0,54	**0,54**
		Softmax	0,67	0,59	0,62	0,54	0,53	0,53
100, 100	FastText	Lineal	0,34	0,35	0,03	0,12	0,17	0,11
		CRF	0,61	0,62	0,61	0,54	0,49	0,51
		Softmax	0,61	0,62	0,62	0,56	0,49	0,52
	Komninos	Lineal	0,33	0,33	0,33	0,06	0,15	0,03
		CRF	0,55	0,57	0,56	0,58	0,49	0,52
		Softmax	0,57	0,58	0,58	0,52	0,48	0,48
	ELMo	Lineal	0,66	0,34	0,34	0,09	0,15	0,07
		CRF	0,61	0,60	0,61	0,61	0,50	**0,54**
		Softmax	0,63	0,57	0,60	0,54	0,53	0,53

Tabla 5. Resultados del experimento n°1: BIO. Observamos que un clasificador sistemáticamente otorga el peor desempeño. Para la categoría de Colectivo observamos un macro-F1 consistentemente mayor a 0.55 exceptuando el caso FastText + Softmax. Para JCnA, un valor de macro-F1 mayor a 0.5 es frecuente, algo positivo notando que se trata de tres categorías complejas y no comparables. No se observa una gran mejoría al utilizar dos redes BiLSTM apiladas al pagar el alto costo computacional que esto supone.

A partir de estos resultados, podemos comenzar a realizar el descarte de configuraciones de hiperparametros. En primera instancia, vemos que la utilización de una capa de clasificación lineal arroja valores inferiores al resto de forma sistemática. Así también, se decidió descartar la utilización de dos redes LSTM de tamaño 100 pues vemos que no aportan mejoras significativas sobre arquitecturas de una única red y que además presentan un costo computacional mucho mayor y propensión al sobreajuste.

Examinando las predicciones de distintos modelos en la partición de desarrollo del dataset, vemos como estos identifican una serie de palabras seguidas como perteneciente a la clase *B-* pertinente, como vemos en los siguientes ejemplos donde el primero corresponde a la predicción de la categoría *Colectivo*, y el segundo a la categoría *JCnA*:

> *Under Trump, Muslim refugees from Burma (Rohingya) are numero uno https://t.co/aJRhNxJNll via @wordpressdotcom Why the hell we importing trouble for USA? Islamists never assimilate for the simple reason their religion does nor PERMIT assimilation into other societies..Period!*

> *A #multibillion-dollarcriminalnetwork is being fueled by inadequeate refugee policies, experts say… https://t.co/yzbKcYGlQX https://t.co/rV0b5bVUEL*

Específicamente, en el primer ejemplo, vemos como en la anotación manual de colectivo encapsula las palabras *Muslim refugees*, mientras que el modelo predice a *Muslim* y *refugees* como dos colectivos independientes. Si bien esta predicción no es del todo incorrecta desde el punto de vista de los colectivos, pues *Muslim* y *refugees* son colectivos, semánticamente vemos que estas dos palabras refieren a un único colectivo, *Muslim refugees*.

Ante este comportamiento, hipotetizamos que los modelos se han apegado demasiado a la forma léxica de las palabras. Esto tiene una fuente clara de procedencia, los datos de entrenamiento.

En los datos donde el modelo aprende, podemos calcular el IMP de la palabra *Muslim* y *refugees* con las clases *B-collective* e *I-collective*; para *Muslim* y *B-collective* se obtuvo -3.22, para *Muslim* e *I-collective* el resultado es indefinido pues nunca sucede en los datos, para *refugees* y *B-collective* un valor de -3.03 y finalmente entre *refugees* e *I-collective*, -3.54.
De esta forma, se puede evidenciar que las palabras *Muslim* y *refugees* están mucho más relacionadas con la clase *B-collective* en los datos de entrenamiento lo cual efectivamente puede influir en las predicciones de los modelos.

Pära el caso de *refugees* e *I-collective*, vemos que su valor IMP es mucho menor a la media IMP de las palabras de la clase *I-collective* con un valor de -2.84, lo cual fortalece aún más

nuestra hipótesis viendo que dada la baja relación entre esta palabra y clase, el modelo no ha podido de igual forma predecir estas dos palabras como un único colectivo.

De cualquier forma, una componente influyente en estas predicciones es la capa CRF, que como mencionamos en la descripción de la arquitectura, realiza clasificaciones tomando en consideración la clasificación de palabras vecinas. Vemos así interés realizar una experimentación tratando de aumentar la influencia que tiene esta capa sobre la arquitectura con el objetivo de que los modelos puedan realizar predicciones como la de *Muslim refugees* vista como un único colectivo, es decir, aumentar la consideración que tiene la capa CRF con las palabras vecinas.

En el segundo ejemplo dado previamente, sucede algo similar entre las palabras *A, #multibillion-dollarcriminalnetwork* y las clases *B-justification* e *I-justification.*
La palabra *#multibillion-dollarcriminalnetwork* presenta un IMP de -2.97 e indefinido para las clases *B-justification* e *I-justification* respectivamente, mientras que la palabra *A* presenta valores de IMP de -3.27 y -4.64.

Vemos entonces similarmente a lo anterior, que ambas palabras están más relacionadas con la clase *B-justification*, sobretodo en el caso de *#multibillion-dollarcriminalnetwork* que presenta un IMP mucho mayor a la media IMP de las palabras de la clase con un valor de -3.27. Esto nos motiva aún más a experimentar con la capa CRF.

Colectivo

Para la predicción de la primer categoría, el modelo que maximiza la métrica emplea los word embeddings de Komninos, char embeddings obtenidos a través de una LSTM, capa de clasificación Softmax y un tamaño de arquitectura LSTM de 100 con una métrica macro-F1 de *0.6296*.

Examinando más detenidamente las predicciones por este modelo, observamos como mencionamos previamente que se encuentra fuertemente lexicalizado, ejemplos de ello es la palabra *MS-13*, nombre de una mafia oriunda de Los Ángeles, la cual fue mal predicha por el modelo en todas las ocurrencias. En los siguientes ejemplos vemos estos casos:

*#BozoBeto Says, He's The Best 4 TX He's For:*Ending #ICE*Open Borders*Legalizing DRUGS*Ending Immigration Laws4 MS-13 Gang Members BRUTALLY Hacked AN Informant To Death W/A Machete In TX! #BuildThatWall#VetoBeto#KeepTexasRed #ChoosCruz*

One-Third of 214 Arrested MS-13 Gangsters Were 'Unaccompanied Alien Children' - Breitbart #EndDACA#NoAmnesty#BuilTheWall#ProtectAmerica#EnforceUSLaws #KeepAmericansSafe

Así entonces, el 100% de las predicciones difieren de la anotación manual, representando un 4% de los errores totales de predicción del modelo.

Es importante destacar, que en muchas situaciones, la predicción de un modelo puede ser considerada como un error si nos apegamos a la definición de error como la discrepancia entre las predicciones del modelo y la anotación manual.

De cualquier forma, dejamos ejemplos donde se produce un error, pero que la predicción del modelo no es del todo errónea, pues la discrepancia con la anotación gold surge de que en varias ocasiones todo un tweet es etiquetado como no argumentativo produciendo así que no se destaque la componente de colectivo. Esto es información la cual nuestros modelos no tienen en consideración, por lo menos en esta instancia de experimentación.

New Olympic sport! Illegal migrant stowaways jump lorry. #IllegalImmigration

Illegal Alien Awaiting Deportation For Child Rape, Now Caught Producing Child Porn https://t.co/t7FLU2CjEy

JCnA

Para la predicción de esta categoría, el modelo que maximiza el macro-F1 con un valor de *0.5449,* utiliza embeddings de ELMo (con word embeddings iniciales de Komninos), sin char embeddings, con capa de clasificación CRF y un tamaño de arquitectura LSTM de 100.

Examinando las predicciones del modelo observamos que ciertas ocasiones flaquea al momento de clasificar las conclusiones que se encuentran al final de un tweet que comienza con una justificación, extendiendo esta última clase hasta el final. Podemos verlo en los siguientes ejemplos:

@DailyMailUK Turkey is a big place with a big population with respect we just do not need more migrants a country who want paid for this

@ScotExpress Please be warned SNP the scots people did not give u a mandate for mass immigration in our name dont under estimate this

@bjdzyak @charlaynek @sabine_durden @IngrahamAngle Wrong! illegal immigrants make 5% of the population so of Course More Americans Commit Crimes because its more Citizens! Yet 34% Of Federal Prisoners are illegals these does not include State Prison

Si bien este comportamiento no es sistemático en las predicciones, nos hace plantear la necesidad de separar la clasificación de *JCnA* en sus componentes, es decir, entrenar modelos para la predicción de *Justificación*, *Conclusión* y *No Argumentativo* como si de *Colectivo* se tratara. Esta tarea se posterga para la próxima experimentación.

Siguiendo con la inspección, notamos que una palabra es sistemáticamente el inicio de la componente de conclusión en las predicciones. Este el caso del hashtag *#BuildThatWall* que ocurre en casi 23% de los tweets, y siendo clasificado un 30% de las ocurrencias como *B-conclusion* siendo así la palabra con mayor cantidad de predicciones como *B-conclusion*. Esto representa un 60% de error de clasificación de *#BuildThatWall*.

Hipotetizamos que esto puede deberse, similar al caso de la predicción de *Colectivo*, al sobreajuste del modelo debido a la fuerte relación que puede existir entre *#BuildThatWall* y la clase *B-conclusion*.

Evaluando el IMP de la palabra con la clase, su valor es de -3.27, siendo mucho menor al IMP medio de la clase *B-conclusion* que presenta un valor de -3.05. Esto descarta nuestra hipótesis, pero de cualquier forma el modelo termina realizando un sobre ajuste en esta palabra. Dejamos ejemplos de estos errores de clasificación

> *@mattfharris @ChiefBordeleau @billcarolltalk @CFRAOttawa @ctvottawa That was not bought in store! Black market merchandise. Yet Libs want to crack down on legal owners? Typical delusion. #Rifle #NRA #BuildThatWall*

> *What the White House has come up with has been shot down by Crime loving Democrats #BuildThatWall #MondayMotivation #BuildThatDamnWallNow*

> *@KurtSchlinchter LEGAL is. Not illegal. #BuildThatWall*

Para el caso de tweets no argumentativos, podemos ver ejemplos donde el modelo a clasificado únicamente un conjunto de palabras de un tweet como no argumentativa, en vez de todo el ejemplo:

> *#BuildThatWall and then extend it to Wall, J, & K Streets. #DrainTheSwamp*

What the White House comes up with has been shot down by Crime loving Democrats #MondayMotivation #BuildThatDamnWallNow

De cualquier forma vemos pocas ocurrencias de predicciones de tweets no argumentativos. Pensamos que esto puede tratarse de una falta de ejemplos de los datos de entrenamiento.

Examinando las predicciones, sólo un 13% de ellas corresponden a las clases no argumentativas como son *B-NoArgumentative* e *I-NoArgumentative*. De los cuales 17% son hashtags y 4% son menciones a otros usuarios. Así mismo, un 6% de ellos pertenecen a la clase minoritaria *B-NoArgumentative*.

Vemos en los datos de entrenamiento, que tan solo 17% de los ejemplos son palabras no argumentativas conformado por 10% de ellos por hashtags y 4% de menciones, y de los cuales un 4% pertenecen a la clase minoritaria.

Concluimos así que efectivamente debido a los pocos ejemplos no argumentativos, el modelo no ha sido capaz de realizar predicciones satisfactorias de la categoría *No Argumentativo*, realizando incluso predicciones de esta clase en medio de tweets argumentativos lo cual produce el incremento de ocurrencias de clasificaciones *B-NoArgumentative*, esto nos motiva a observar el comportamiento de los modelos una vez disminuido el espacio de clases con un formato *IO* y separando las clases que componen *JCnA*, obteniendo así un clasificador binario.

IO

Bajo un conjunto de configuraciones de parámetros de la arquitectura reducida y formato de datos de entrada *IO*, se entrenaron modelos para la predicción de todas las categorías, incluyendo las clases de *Justificación*, *Conclusión* y *No Argumentativo* por separado.

En general, se observó que la capa de *char embeddings* no produce una mejora significativa en las métricas macro-F1 de los modelos; tan solo un 3% de mejoría en el mejor de los casos. Es por este motivo por el cual descartamos esta capa en próximos experimentos, realizando un seteo estático de *char embeddings* a *None*.

Colectivo

Para la clasificación de la clase *Colectivo*, como vemos en la Tabla 6, el mejor modelo con macro-F1 de *0.7539* y F1 de clase minoritaria de 0.5172 utiliza una capa de clasificación CRF, junto a embeddings de Komninos y sin capa de *char embeddings*.

Notamos que los *word embeddings* de mejor resultado son los mismos que obtenidos en el experimento anterior, no así para el caso de la capa de clasificación y de *char embeddings*. Podemos atribuir esta diferencia a que la disminución de espacio de clases debido a la utilización de un formato *IO* permite a la capa CRF realizar un mejor etiquetado en base a los etiquetados vecinos sobre el clasificador Softmax.Sin embargo, debemos destacar que

tanto las capas CRF y Softmax han dado buenos resultados sin ninguna de ellas contar con mayor cantidad de modelos favorables.

Parámetros			Colectivo		
Embedding	Char embedding	Clasificador	Pr	Rec	F1
Twitter FastText	None	CRF	0,61	0,41	0,49
		Softmax	0,51	0,29	0,37
	CNN	CRF	0,50	0,42	0,46
		Softmax	0,48	0,43	0,45
	LSTM	CRF	0,47	0,32	0,38
		Softmax	0,48	0,32	0,38
Komninos	None	CRF	0,64	0,43	**0,52**
		Softmax	0,54	0,39	0,45
	CNN	CRF	0,54	0,30	0,39
		Softmax	0,53	0,46	0,50
	LSTM	CRF	0,59	0,39	0,47
		Softmax	0,53	0,35	0,42
ELMo	None	CRF	0,43	0,45	0,44
		Softmax	0,54	0,38	0,44
	CNN	CRF	0,51	0,39	0,44
		Softmax	0,52	0,48	0,50
	LSTM	CRF	0,44	0,52	0,48
		Softmax	0,44	0,41	0,42

Tabla 6. Resultados del experimento n°2: IO en categoría de Colectivo. Se reportan métricas de la clase minoritaria I-Collective. No se observan mejoras empleando embeddings ricos en contexto a partir de ELMo. El clasificador "linear chain" CRF presenta en 5 de las 9 pares de modelos el mejor desempeño a comparación del clasificador Softmax.

Examinando las predicciones, vemos que un 30% de estas son sobre las palabras *illegal* y su parecida *illegals*, 8% sobre *refugees* y otro 8% sobre *immigrants*.
Para el caso de *illegal*, observamos que el 30% de estas predicciones son erróneas así cómo un 40% para el caso de *illegals* y 50% en *refugees*. Dejamos al lector los siguientes ejemplos de estas predicciones erróneas:

> *13 Americans Victimized by Illegal Migrant Crime in One Week @SpeakerRyan @HouseGOP Why won't you STOP this constant assault on American Citizens and the steady invasion by foreigners on US soil ? We the people Demand an end to this!*

“”"@AVIACUSA” 8% of Illegals requesting DACA had been arrested for rape, murder, and drunk driving. 50% of DACA Illegals commited Fraud to get #DACA 73% of DACA households on welfare #RedNationRising #Trump #MAGA #NoDaca #NAomnesty #SendThemBack

Under Trump, Muslim refugees from Burma (Rohingya) are numero uno https://t.co/aJRhNxJNll via @wordpressdotcom Why the hell we importing trouble for USA? Islamists never assimilate for the simple reason their religion does nor PERMIT assimilation into other societies..Period!

Vemos en los primero dos como el modelo a clasificado siempre la ocurrencia de la palabra Illegal como *Colectivo*, de forma reiterativa en el segundo ejemplo lo cual no es incorrecto pues efectivamente se refiere al mismo *Colectivo*, esto lo podemos clasificar como una improlijidad de los anotadores del tweet que tan solo clasificaron la primera ocurrencia de la palabra pues es el único *Colectivo* que interviene.

En si la relación entre las palabras *illegal* e *illegals* con respecto a la clase *I-collective* es de un IMP de -4.01 y -3.79 respectivamente, lo cual es mucho menor a la media IMP de palabras de esta clase con un valor de -2.97 por lo cual podemos concluir que en los datos de entrenamiento, estas palabras no están muy fuertemente relacionadas de tal forma de entorpecer el entrenamiento del modelo.

El último ejemplo es uno visto en el primer experimento, donde el modelo clasificaba ambas palabras como *B-collective*, pero que en esta instancia no las clasifica como colectivos. Al igual que para las palabras *illegal* e *illegals*, no se ve una relación fuerte entre la palabra *refugees* e *I-collective* con un IMP de -3.54.

Este modelo sigue clasificando de forma incorrecta el 100% de las ocurrencias de la palabra *MS-13* como I-collective:

*#BozoBeto Says, He’s The Best 4 TX He’s For:*Ending #ICE*Open Borders*Legalizing DRUGS*Ending Immigration Laws4 MS-13 Gang Members BRUTALLY Hacked AN Informant To Death W/A Machete In TX! #BuildThatWall#VetoBeto#KeepTexasRed #ChoosCruz*

One-Third of 214 Arrested MS-13 Gangsters Were ‘Unaccompanied Alien Children’ - Breitbart #EndDACA#NoAmnesty#BuilTheWall#ProtectAmerica#EnforceUSLaws #KeepAmericansSafe

Al igual que el experimento anterior hipotetizamos que se trata de un sobre ajuste del modelo sobre esta palabra, un aprendizaje en base al léxico de la misma. Para ello, vemos su IMP con respecto a la clase I-collective lo cual nos arroja un valor de -2.57, superior a la media mencionada previamente. Con esto concluimos que efectivamente el modelo se ve lexicalizado sobre esta palabra, motivo por el cual debemos independizar las predicciones del modelo de su léxico, daremos el abordaje a realizar en el próximo experimento en los siguientes párrafos.

Justificación - Conclusión - No Argumentativo

A raíz de optar por no utilizar *char embeddings*, el número de modelos que predicen las componentes que conforman *JCnA* disminuye, obteniendo como métricas resultantes aquellas vistas en la siguiente Tabla 7.

Parámetros		**No Argumentativo**			**Justificación**			**Conclusión**		
Embedding	Clasificador	Pr	Rec	F1	Pr	Rec	F1	Pr	Rec	F1
Twitter FastText	CRF	0,50	0,42	0,46	0,66	0,78	0,72	0,60	0,40	0,48
	Softmax	0,52	0,40	0,45	0,67	0,67	0,67	0,57	0,45	0,50
Komninos	CRF	0,48	0,56	**0,52**	0,72	0,71	0,71	0,52	0,50	0,51
	Softmax	0,52	0,47	0,49	0,68	0,67	0,68	0,54	0,49	0,51
ELMo	CRF	0,57	0,40	0,47	0,69	0,76	0,72	0,62	0,51	0,56
	Softmax	0,49	0,43	0,45	0,68	0,78	**0,73**	0,63	0,53	**0,58**

Tabla 7. Resultados del experimento n°2: IO en categorías que componen JCnA, sin emplear char embeddings. Se reportan las métricas de las clases minoritarias. La capa de ELMo aumenta el desempeño tanto en Justificación como Conclusión, aportando una gran diferencia en este último, pero empeora los modelos de predicción de tweets no argumentativos de forma considerable. "Linear chain" CRF presenta sistemáticamente mejor desempeño en la categoría No Argumentativo, además de formar parte de 2 de los 3 mejores modelos de predicción de Justificación frente a Softmas, este último clasificador se encuentra en 2 de los 3 modelos con mejor desempeño para la predicción de la componente de Conclusión.

Para analizar las componentes de *Justificación*, *Conclusión* y *No Argumentativo* hipotetizamos sobre su posición en el tweet mismo, generalmente esperaríamos que las justificaciones estén en el principio del tweet (i.e. comenzando dentro de las primeros 5 palabras), las conclusiones al final (i.e. finalizando dentro de las últimas 5 palabras) y que lo *No Argumentativo* abarque la completitud de un tweet, y no que se encuentre disperso. Toda esta información observada en el análisis detallado del dataset.

Además, nos es de interés evaluar la presencia de estas componentes en base a la longitud de los tweets. Por ejemplo, como vimos en el análisis del dataset, los tweets no argumentativos tienen una longitud media menor a aquellos con componentes argumentativos.

Para el caso del mejor modelo para predicción de *Justificación* utiliza una capa Softmax con *word embeddings* generados por ELMo inicializado con Twitter FastText obteniendo un valor de 0.7286.

Examinando cuantitativamente, vemos que 50 (50%) de los ejemplos la justificación se encuentra al inicio del mismo y en 20 de ellos al final (20%). Vemos que estos porcentajes son muy similares a los de entrenamiento, donde un 51% de los ejemplos comienza con una justificación y un 22% de ellos finaliza con la misma.
Además, observamos que en 8 (0.8%) ejemplos el modelo predijo más de una componente de justificación en el mismo tweet, aquí tres ejemplos de ello:

> *White House wants to make it harder for legal migrants to be citizens https://t.co/Rt5WP8UF0j via @MailOnline GOOD!!! ONLY THE BEST!! Welfare leeches Go back home!!!*

> *Danish #Muslims Insists On Wearing A #Niqab (or FaceVeil) & Still Desire To Live As A Non-Person Ghost. Muslims Are a Hopelessly Unable To Assimilate. #SendThemBack. @victorban @realDonaldTrump @GiuseppeConteIT @Peggoynoonannyc https://t.*

> *Salvini prosecuted for defending Italian sovereignty and finally preventing hundreds of migrants to invade Italy. GRANDE SALVINI! Help us preserve the European culture against the invasion. #stop_islamization #complicediSalvini #stopInvasion*

También vemos varios ejemplos donde el modelo extiende la longitud de la componente de justificación diferenciándose de las anotaciones manuales como ocurría en el experimento anterior empleando formato *BIO*, como por ejemplo:

> *Italia stopps again a ship with more than 400 illegal refugees. They must concentrate into spots and should immidatley sended back to Africa. In USA they have to do same. https://t.co/DYiQNw4liN*

> *13 American Victimized by Illegal Migrant Crime in One Week @SpeakerRyan @HouseGOP Why won't you STOP this constant assault on American citizens and the steady invasion by foreigners on US soil? We the people Demand an end to this!*

Las predicciones presentan así una media de longitud de 13.4 palabras con 10.7 de desviación estándar, cercano a los valores de entrenamiento de 13.3 y 11.5 respectivamente.

Un aspecto que llama la atención de las predicciones del modelo es que en 41 (41%) ejemplos la justificación termina con un signo de puntuación, exclamación o interrogación.

> *"Russia is building up its forces in Syria any action against ISIS, should be welcomed after that start talking, No more migrants*

> *@DiamondandSilk Looks like he doesnt keep his front door open.. but they expect everyone else to. #BuildThatWall*

> *@M2theMfknJ Agreed MJ! Any Country that has an #OpenBorders policy is in trouble! #BuildThatWall #VoteRedToSaveAmerica #AmericaFirst*

Podemos hipotetizar que estos signos presentan una alta IMP con respecto a la clase *I-justification* en los datos de entrenamiento.

Evaluando esto, obtenemos que el *"."* presenta un IMP de -4.49, la *","* -4.32, el *"!"* -4.63 y el *"?"* un valor de -4.43, valores muy cercanos entre ellos pero que no demuestran una relación fuerte con respecto a la media IMP de las palabras con la clase I-justification que presenta un valor de -4.03. Por lo tanto el modelo no podría haberse basado en los datos de entrenamiento, pero debido a la utilización de la capa CRF, ésta sí puede haberse visto afectada por la poca cantidad de ejemplos y relativa frecuencia entre estos signos y la clase en el final de las componentes de justificación para generar este comportamiento.

Para la predicción de *Conclusión*, el mejor modelo se compone de una capa de clasificación Softmax, *word embeddings* ricos en contexto por ELMo partiendo de Komninos, con un valor de 0.5751.

Tan solo 12 tweets (12%) contienen esta clase al inicio, y 39 (39%) al final del mismo lo que presenta una tendencia del modelo a predecir palabras como conclusión al final del tweet manteniendo así cierta relación con aquellos presentes en los datos de entrenamiento donde un 22% de los ejemplos comienza con una conclusión y 35% de ellos finaliza con ella.

En general las conclusiones del modelo abarcan pocas palabras con una media de 7.09 por tweet con desviación de 8.91, también similar a los datos de entrenamiento con 7.44 de media y 8.30 de desviación.

En la examinación de las predicciones vemos una tendencia del modelo a cortar la componente de conclusión en varias partes, ejemplos:

*No, bi*tch. You killed him by coming here illegaly! You both made the choice. Get the hell out of our country! #DeportThemAll #DepportImmediately #buildthatwall #GetOut Wife of unducumented Mexican dad blames Trump for his suicide @MailOnline*

Ok, I am asking out of curiosity. Lets suppose a group of immigrant in Netherlands started fighting & killing local security forces and requested that they want some land to form their own government. In your logic, those aren't terrorrists? & sec. forces are legitimate targets? https://t.co/3UglggTDkH

@LBC @clivebull When you have too many people living in one place you are going to get trouble. Sadiq Khan keeps on encouraging more & more people to come. It won't end well immigration has reached a tipping point.

Llegando al caso inclusive de marcar una única palabra sin sentido aparente como conclusión:

@BJPRajnathSingh Must take decisive action for sealing borders with B'desh to prevent rampant illegal migration encouraged by one community

Did you know? Government statitics of crimes committed by illegalsillegals comprise 7% of populationillegals account for 72% of all drug posseession33& of all money laundering29% of all drug traficking22% of all federal murders18%of all frauds"NODACA

#Burundi Burundian refugees should go home! https://t.co/Ox9Twxanlo

Vemos que esto puede tratarse de la capa Softmax que el modelo utiliza junto a la falta de información sobre si un tweet es argumentativo, o no.

Además, notamos que en muchos tweets se clasifican hashtags como *Conclusión*, en total 34 (4.8%) palabras de 709 comienzan con hashtag y fueron abarcadas como pertenecientes a esta categoría concluyendo así en 26 (26%) tweets con esta propiedad.

Un hastag que tiende a clasificar mucho el modelo es *#SendThemBack*, específicamente un 50% de las ocurrencias (de ocho en total) acertando tan solo la mitad de ellas. Aquí un par de ejemplos:

Daily migrant invasions #migrant #buildthewall #illegal #SendThemBack https://t.co/QSLuo17CrJ

“””Please don’t call it “””””rescue””””” - it’s human trafficking #portsclosed #SendThemBacK

Como se hizo anteriormente, hipotetizamos que esto puede deberse a una lexicalización del modelo sobre esta palabra en el momento del entrenamiento. Revisando su IMP con la clase *I-conclusion* vemos que tiene un valor de -3.83 la cual es mayor a la media IMP de las palabras con dicha clase de -3.99 concluyendo así una alta relación entre ambas. Motivo por el cual el modelo se ve sobre ajustado en este hashtag.

En lo que a predicción de componente *No Argumentativo* se refiere, el mejor modelo con un macro-F1 de 0.5178 cuenta con una capa *“linear chain”* CRF y *word embeddings* de Komninos.

El modelo predijo 21 (21%) tweets completos como *No Argumentativos* equivocándose en 11 (52%) de los mismos. En uno de estos casos, el modelo marco un tweet por completo como no argumentativo mientras que la anotación manual omitía el link final que presentaba, siendo esto un error del anotador mismo.

Germany to hold migrants in detention centres on Austria border https://t.co/dZ2LCLwhqR

Además en algunos ejemplos erróneos, la clasificación no cubrió la totalidad del tweet, sino que comenzó en un punto distinto al inicio y siguió hasta el final, como por ejemplo:

EU LEADERS are upset with Italy for closing its doors to the “cultural enrichment” that Muslim migrants bring https://t.co/ndd17dxKOC via @barenakedislam Enrichment update https://t.co/4OwEEIRJbq

Illegal Alien Awaiting Deportation For Child Rape, Now Caught Producing Child Porn https://t.co/t7FLU2CjEy

"So sad the elite, politicians and virtue signallers just don't get it. #IllegalImmigration is not multiculturism.

La longitud media de los tweets marcados como *No Argumentativo* es de 138.15 caracteres con una desviación de 54.76 caracteres, muy similar a lo visto en el análisis del dataset completo.

Propiedad y Pivote

En la clasificación de *Propiedad* y *Pivote*, se vieron resultados macro-F1 mayores a 0.5 pero cuyo desempeño en la tarea de predecir dichas clasificaciones era de muy mala calidad.

Específicamente un modelo con capa de clasificación CRF, *word embeddings* contextuales de ELMo inicializados con Komninos y *char embeddings* de una red LSTM obtuvo un macro-F1 de 0.6377 para la predicción de *Propiedad*, y un modelo con capa de clasificación Softmax, *embeddings* contextuales de ELMo inicializados con *embeddings* del modelo FastText entrenado con datos de Twitter y sin capa de *char embeddings* obtuvieron un macro-F1 de 0.5809 para predicción del *Pivote*.

A raíz de esto, se optó por dar un foco de atención a las métricas de *recall* y precisión de la clase minoritaria. Esta sería la clase *I-property* e *I-pivot* para las clases *Propiedad* y *Pivote*, respectivamente, que podemos observar para todos los modelos entrenados en la siguiente Tabla 8.

Parámetros			**Propiedad**			**Pivote**		
Embedding	Char embedding	Clasificador	Pr	Rec	F1	Pr	Rec	F1
Twitter FastText	None	CRF	0,41	0,20	0,27	0,18	0,10	0,13
		Softmax	0,44	0,18	0,25	0,17	0,12	0,14
	CNN	CRF	0,57	0,14	0,22	0,22	0,12	0,15
		Softmax	0,39	0,14	0,21	0,20	0,10	0,13
	LSTM	CRF	0,33	0,12	0,18	0,23	0,10	0,14
		Softmax	0,53	0,16	0,25	0,21	0,14	0,17
Komninos	None	CRF	0,33	0,18	0,23	0,10	0,02	0,04
		Softmax	0,44	0,15	0,22	0,13	0,11	0,12
	CNN	CRF	0,61	0,14	0,22	0,19	0,07	0,10
		Softmax	0,34	0,18	0,23	0,17	0,07	0,10

	LSTM	CRF	0,28	0,22	0,24	0,12	0,07	0,09
		Softmax	0,40	0,14	0,21	0,16	0,09	0,11
ELMo	None	CRF	0,41	0,20	0,27	0,24	0,11	0,15
		Softmax	0,48	0,23	0,31	0,29	0,13	**0,18**
	CNN	CRF	0,52	0,20	0,29	0,24	0,11	0,15
		Softmax	0,39	0,25	0,30	0,19	0,11	0,14
	LSTM	CRF	0,46	0,25	**0,32**	0,22	0,09	0,13
		Softmax	0,48	0,24	**0,32**	0,26	0,10	0,14

Tabla 8. Resultados del experimento n°2: IO en categorías de Propiedad y Pivote. Se incorporan char embeddings por el bajo desempeño de ambas, en búsqueda de una mejoría con este extra de información. Se reportan métricas de las clases minoritarias. Esta incorporación de char embeddings no aparenta aportar utilidad en los modelos para mejorar su desempeño, sin embargo observamos que aquellos obtenidos de una red LSTM favorecen muy ligeramente la predicción de Propiedad. Embeddings de ELMo obtuvieron mejor desempeño en ambas categorías, aumentando cerca de 0.03 puntos la F1. Pivote se presenta como la categoría con peor métrica de predicción al momento, seguido de la Propiedad, debido a su complejidad natural que los modelos con su superficialidad no parecen poder explotar.

Vemos así que sus valores son bajos siendo 0.25 y 0.14 los mejores resultados para el *recall*, 0.61 y 0.29 para el caso de la precisión (para *Propiedad* y *Pivote*).

A partir de lo expuesto previamente de estas métricas, vemos que los modelos de clasificación de *Propiedad* son de mediana calidad pero cuya capacidad de clasificar es baja, rozando el 1 de cada 4 predicciones correctas. Pero para el caso del *Pivote*, la calidad de los modelos es muy baja y cuya predicción tampoco destaca llegando a tan solo un 28% de predicciones correctas. Siempre tratándose de los mejores casos de los modelos.

Ante este suceso, realizamos una inspección más detallada a las predicciones de los mejores modelos de clasificación de *Propiedad* y *Pivote*.

En el caso de *Propiedad*, nos llama la atención el comportamiento de las palabras *crime* y su parecida *crimes* con respecto a esta categoría. Según el experto de dominio, se trata de palabras muy relacionadas con la propiedad a predecir.

En los siguientes ejemplos podemos ver cómo la palabra *crime* ha sido etiquetada por el modelo como parte de una propiedad, en el primer caso, de acuerdo con la anotación manual (acierto), y en el segundo caso, a diferencia de la anotación manual (error).

@SecNielsen @MOSecofState @JayAshcroftMO @ElectionSummit @DHSgov American Citizens need @ICEgov Illegals can reunite back across the border in Mexico! We dont need the extra crime that the illegal or refugees invaders bring to America.#NoDACA #NoVisaLotte

@J_VoiceUK Listening to the world service this am criticism of Israel for

wanting to deport young African male migrants it maybe evidence from Europe like France and France many are not settling and are turning to crime better the truth than living a lie

En las predicciones realizadas por el modelo, la mitad de las ocurrencias de la palabra *crime* fueron clasificadas como *I-property*, pero nunca la palabra *crimes*. Entonces, nuestra hipótesis es que el modelo tiene una fuerte asociación de la palabra *crime* con la categoría *I-property.*

Ahora bien, el modelo tiene que haber inferido la asociación entre *crime* e *I-property* de los datos de entrenamiento. Explorando el conjunto de tweets de entrenamiento, observamos que el 0.6% de los ejemplos de *I-property* contienen la palabra *crime*, y un 69% de las ocurrencias de *crime* han sido etiquetadas como *I-property*, mientras que el 0.2% de los ejemplos de *I-property* contienen la palabra *crimes* y un 2% de las ocurrencias de *crimes* han sido etiquetadas como *I-property.*

En los siguientes ejemplos vemos casos residuales en los que *crimes* no ha sido identificado manualmente como parte de *I-property* y, a la inversa, *crimes* sí forma parte de una *I-property*.

Did you know? Government statistics of crimes committed by illegalsIllegals comprise 7% of populationIllegals account for72% of all money laundering29% of all drug trafficking22% of all federal murders18%of all frauds#NODACA

#DREAMER Stomps 83 yr old man to death in TXSSilvano Echavarria stomped the victims head 74 times and punched him 25 times before leaving his body in a parking lotHis rap sheetincludesother crimes.

Podemos representar esta relación de forma compacta con la métrica de información mutua puntual (IMP) que hemos explicado en la sección de Métricas, y utilizada previamente.

Efectivamente, encontramos un valor de información mutua puntual de -3.44 para *crime* e *I-property*, y de -3.98 para *crimes* e *I-property*. Podemos ver entonces qué *crime* está más fuertemente asociado a la categoría que *crimes.*

Además, podemos ver que el promedio de IMP de las palabras que ocurren con *I-property* es de -3.57, con lo cual *crime* está mucho más fuertemente asociado a la categoría que el promedio.

Esto nos motiva a explorar el parámetro configurable *dropout* de la arquitectura, la cual permite a los modelos neuronales modelar de forma más concreta o genérica los ejemplos de entrenamiento. Buscamos poder independizar la clasificación del modelo con respecto a

la palabra específica. Evitando así el apego que presentó este modelo de clasificado de propiedad con la palabra *crime*.

En lo que a *Pivote* se refiere, vemos que no se encuentra fuertemente lexicalizado como sucedía con la categoría *Colectivo* debido a que hay palabras como *ilegal* donde no encontramos errores sistemáticos: hay ocasiones donde la palabra es clasificada por el modelo como *I-pivot* pero no por la anotación gold, y viceversa. Esto lo podemos ver mejor en los siguientes ejemplos con dicha palabra:

> *@bjdzyak @charlaynek @sabine_durden @IngrahamAngle Wrong! illegal immigrants make 5% of the population so of Course More Americans Commit Crimes because its more Citizens! Yet 34% Of Federal Prisoners are illegals these does not include State Prison*

> *@realDonaldTrump @FoxNews TRAVEL ADVISORYDominic Durden & Mollie Tibbetts & Kate Steinle were murdered by illegal aliens! We must #BuildThatWallFollow @AVIACUSA & @sabina_durden & #RETWEET Help out & order bricks or donate today at Follow @BorderWallUSA*

Pero sí observamos cierta lexicalización, pues un 33% del acierto contiene la palabra *ilegal*, mientras que en el caso donde el modelo falla cerca del 10% contiene la palabra *ilegal*.

Evaluando la IMP de esta palabra con la categoría en los datos de entrenamiento su valor es de -3.83, vemos que está por debajo de la media la cual cuenta con un valor de -3.38. Por lo tanto podemos concluir que no existe una fuerte relación, en los datos de entrenamiento, entre la palabra *ilegal* y la clase *I-pivot*. Esto puede ser porque la palabra ilegal es muy frecuente en todo este dataset, ya que es una de las palabras centrales del dominio, y por lo tanto, por su frecuencia alta en el corpus general, su valor de IMP con I-pivot es bajo. Sin embargo, el modelo efectivamente ha inferido una asociación fuerte entre la palabra *ilegal* y la categoría *I-pivot*, sobre ajustando, tal como queda en evidencia por los errores en predicción.

Por otro lado, examinando aún más las predicciones vemos que la palabra *Taxpayer* se encuentra muy relacionada con la clase *I-pivot* con un PMI de -2.91, debido a su única ocurrencia en todos los datos de entrenamiento. Presentando así un PMI mayor a la media.

En las predicciones de esta palabra por el modelo, fue predicha como *I-pivot* el 100% de las veces acertando un 0% de dichas predicciones. Ilustramos con las siguientes predicciones a tweets con dicha palabra:

> *@KamalaHarris Illegals Dump their Kids at the border like Road Kill and Refuse to Unite! They Hope they get Amnesty, Free Education and*

Welfare Illegal #FamiliesBelongTogether in their Country not on the Taxpayer Dime Its a SCAM #NoDACA #NoAmnesty #SendThe

@KHOU Aiding Illegals is against federal Immigration laws a Felon up to 5 years in Prison! Help #SendthemBack instead of Burdening the U.S Taxpayer ! U.S Is NOT nCentral America's dumping ground for their kids we have .,S Homeless kids we need to help f

El modelo se ha sobre ajustado a la palabra *Taxpayer* bajo un único ejemplo lo cual nos motiva aún más a examinar el *dropout* pues concluimos que no están generalizando de forma satisfactoria, lo cual es un problema de mayor calibre para la predicción del pivote, pues es de las más complejas por su carácter explícito o implícito, generando asi debate entre los anotadores mismos del dataset.

Tipos de Justificación y Conclusión

Las métricas fueron tabuladas en la Tabla 9. La predicción de tipos se divide en dos tareas independientes, predicción del *Tipo de Justificación* y predicción del *Tipo de Conclusión*.

Además, como mencionamos en el detalle del experimento, no se trata de una tarea binaria, por lo cual una clase minoritaria no es clara, es por ello que se encuentran la macro-F1 en la tabla pero desarrollaremos las minoritarias a lo largo de este análisis.

Parámetros		**Tipo de Justificación**			**Tipo de Conclusión**		
Embedding	Clasificador	Pr	Rec	macro-F1	Pr	Rec	macro-F1
Twitter FastText	CRF	0,35	0,36	0,35	0,39	0,38	0,38
	Softmax	0,41	0,38	**0,39**	0,40	0,41	0,41
Komninos	CRF	0,30	0,32	0,31	0,44	0,39	0,40
	Softmax	0,34	0,35	0,34	0,39	0,37	0,38
ELMo	CRF	0,34	0,35	0,34	0,41	0,42	0,41
	Softmax	0,35	0,36	0,36	0,70	0,43	**0,44**

Tabla 9. Resultados del experimento n°2: IO en categorías de Tipo de Justificación y Tipo de Conclusión. Se reportan métricas macro-F1 de los modelos. Se observa que todos los modelos superan un umbral de 0.3 en su macro F1, presentando el Tipo de Conclusión un mejor desempeño en predicción frente al Tipo de Justificación, recordando que este último presentó casi un nulo acuerdo entre los anotadores manuales. El recall es mayor en el Tipo de Justificación frente a la precisión, lo cual es inverso para el Tipo de Conclusión. ELMo mejora el desempeño de los modelos en la predicción de esta última categoría.

Comenzando por la predicción del *Tipo de Justificación*, observamos un macro-F1 de 0.3853 al modelo que utiliza una Softmax como capa de clasificación y *word embeddings* de Twitter FastText. Vemos además en la tabla como la utilización de este último obtiene un mayor macro-F1 que utilizando Komninos, sistemáticamente.

En la siguiente Tabla 10 se observan las métricas individuales de los distintos tipos para este modelo.

Tipo	Justificación		
	Pr	Rec	F1
Hecho	0,62	0,27	0,66
Valor	0,28	0,11	0,16
Política	0	0	0

Tabla 10. Resultados ampliados en la predicción del Tipo de Justificación para sus métricas de clase minoritaria, según el mejor modelo obtenido. Mejor desempeño en la clase predominante, de Hecho, seguido del tipo de Valor. Nulo desempeño para la predicción del tipo de Política debido a los pocos ejemplos que se cuentan para esta clase. Se ve una alta precisión, lo que traducimos como predicciones de calidad, pero bajo recall destacando la poca cantidad de predicciones correctas.

El *Tipo de Justificación Política* fue predicho una única vez por el modelo de forma errónea, además las predicciones de este tipo que debió haber realizado fueron confundidas por el tipo de *Hecho*, es por ello su nulo desempeño. Dejamos en los próximos tweets la única predicción del modelo, seguido de un par de estos casos donde el modelo predijo otro tipo de justificación:

> *We simply cannot allow people to pour into the United States undetected, undocumented, unchecked and circumventing the line of people who are witing patienly, diligently, and lawfully to become immigrants in this country. Barack Obama - 2005#BuildThatWall #MAGA #Trump*

> *“@ClodaghSnarks @nftwaybill @BBCNews We must stop importing refugee terrorists, we should never play Russian roulette with lives of British citizens. #westminister #parsonsgreen #carolinelucas #refugeesnotwelcome”*

> *@rajnathsingh Sir, pl don’t bow to mad Mamta/TMC whims & threats. Centre must go ahead & deport all illegal migrants from all over India. India must not be made home to Muslim illegal migrants. Pl don’t abt consequences. Entire India is with u in throwin*

Vemos además que el modelo tiende a mezclar tipos en una misma justificación, cuando en las anotaciones manuales cada justificación es de un único tipo. Esto sucedió sobre todo con el tipo de *Valor* cuyas palabras clasificadas de este tipo se ven rodeadas de otras

palabras del tipo *Hecho*. Dejamos los próximos ejemplos ilustrativos donde los dos primeros fueron anotados manualmente como de *Hecho*, y el último de *Valor*:

> *@SenSchumer You know something Chcuky. We the people dont give a rats behind what you say. You and your party have done enough to destroy America. We are done with your ideals. #MAGA #WalkAway #DrainTheSwamp #BuildThatWall*

> *"@john_w_wilcox @MagzyG1963 @PrinceBraith9th Correct/Incorrect - doesnt really matter, the point is that it should not be allowed. The number of immigrants coming to England and taking all they can is ridiculous. No wonder they fight tirelessly to get through all the other countries into England. #stopimmigration"*

> *#BuildThatWallNow I do not want those vile thugs in our country! #EndChainMigration #EndSanctuaryCities #EndVisaLottery #AngelFamilies*

En sí vemos una longitud media de palabras de tipo *Valor* como *Justificación* por tweet baja (de 12.35 palabras), lo cual se corresponde con lo visto en la sección del análisis del dataset.

Encontramos una tendencia a clasificar ejemplos no argumentativos (10%), sobre extender el etiquetado y también de partir estos etiquetados a lo largo del tweet como vimos en los ejemplos anteriores. Estos imperfectos son justificables a partir de entender que el modelo de cierta forma está prediciendo la presencia de una *Justificación* además del *Tipo* de la misma.

Vemos en la siguiente Figura 12 la matriz de confusión de las predicciones del modelo donde observamos mejor lo que se describió anteriormente

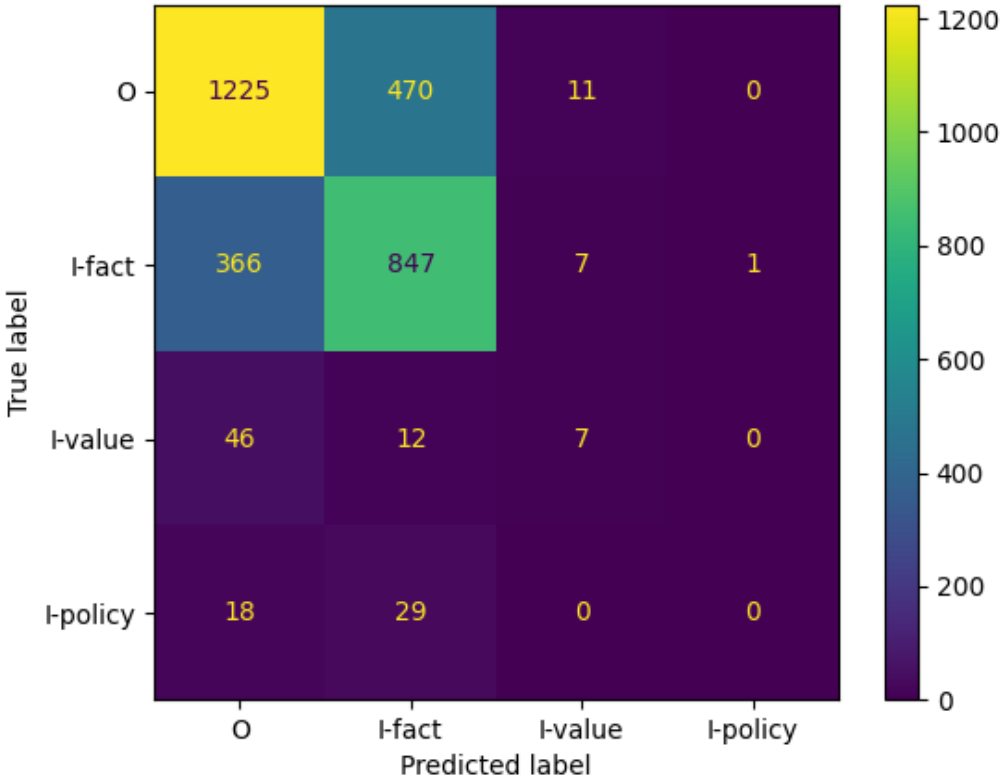


Figura 12. Matriz de confusión del mejor modelo obtenido en el experimento n°2: IO para la predicción del Tipo de Justificación. Observamos como la predicción de la clase O fue predominante con 470 predicciones erróneas asignadas al Tipo de Hecho, en donde vemos que el modelo tiende a clasificar fuertemente palabras como tipo pertenecientes a este tipo predominante en las anotaciones manuales. El Tipo de Valor fue propensamente mal categorizado como O, es decir, sin tipo, mientras que en el tipo de Política el modelo tendió a errar prediciendo cómo de Hecho.

De esta matriz observamos una ligera predisposición a clasificar justificaciones de *Valor* como de Hecho, esto es algo que los anotadores manuales tuvieron dificultades. Poder diferenciar entre uno u otro se basó explícitamente en el hecho de dejarse claro que se trataba de una opinión subjetiva del sujeto.

Para el caso de clasificación de conclusión según su tipo, también se utiliza la capa de clasificación Softmax pero utilizando *word embeddings* de ELMo inicializados con *embeddings* de Komninos obteniendo un macro-F1, mayor al del *Tipo de Justificación*, de 0.4360, y como vemos en la Tabla 11 las siguientes métricas individuales según su tipo:

Tipo	**Conclusión**		
	Pr	Rec	F1
Hecho	0,5	0,2538	0,3367
Valor	1.0	0,0127	0,0251
Política	0,532	0,5746	0,5525

Tabla 11. Resultados ampliados en la predicción del Tipo de Conclusión para sus métricas de clase minoritaria, según el mejor modelo obtenido. Al igual que sucede con el Tipo de Justificación, se observa un alto desempeño en el tipo predominante, de Política, seguido del tipo de Hecho. Además,

notamos un bajo desempeño el tipo minoritario de Valor, que cuenta con el valor de precisión más alto debido a sus dos únicas palabras correctas predichas como pertenecientes a esta clase.

Observamos un F1 mayor en la clase de tipo *Política* que hipotetizamos, conociendo el dataset, proviene de la mayor frecuencia de esta clase en los datos.

La precisión del tipo Valor se debe en parte a su poca predicción, con un total de dos, siendo ambas correctas como vemos en el próximo ejemplo:

@mattfharris @ChiefBordeleau @billcarrolltalk @CFRAOttawa @ctvottawa That was not bought in a store! Black market merchandise. Yet Libs want to crack down on legal owners? Typical delusion. #Rifle #NRA #BuildThatWall

Vemos además que las predicciones también se encuentran divididas a lo largo de 21 (21%) de los ejemplos, lo que produce la discrepancia con la anotación manual. Dejamos un caso residual donde por como partimos las palabras, detallado en el dataset, el modelo únicamente clasificó el apóstrofe que compone la palabra won't, pero no las demás partes:

13 Amercans Victimized by Illegal Migrant Crime in One Week @SpeakerRyan @HouseGOP Why won't you STOP this constant assault on American citizens and the steady invasion by foreigners on US soil? We the People Demand an end to this! #DeportThemAll

En la siguiente Figura 13 observamos la matriz de confusión del modelo, en ella observamos que la clase *O* fue la más predicha de forma correcta con 1863 palabras y una mayor equivocación con el tipo de Hecho en otras 203, le sigue la clase *I-policy* con 258 predicciones correctas presentando 162 casos de error con la clase *O*. La predicción de *I-value* obtuvo la menor cantidad de predicciones como vimos anteriormente con 131 palabras erróneamente predichas como *O*.

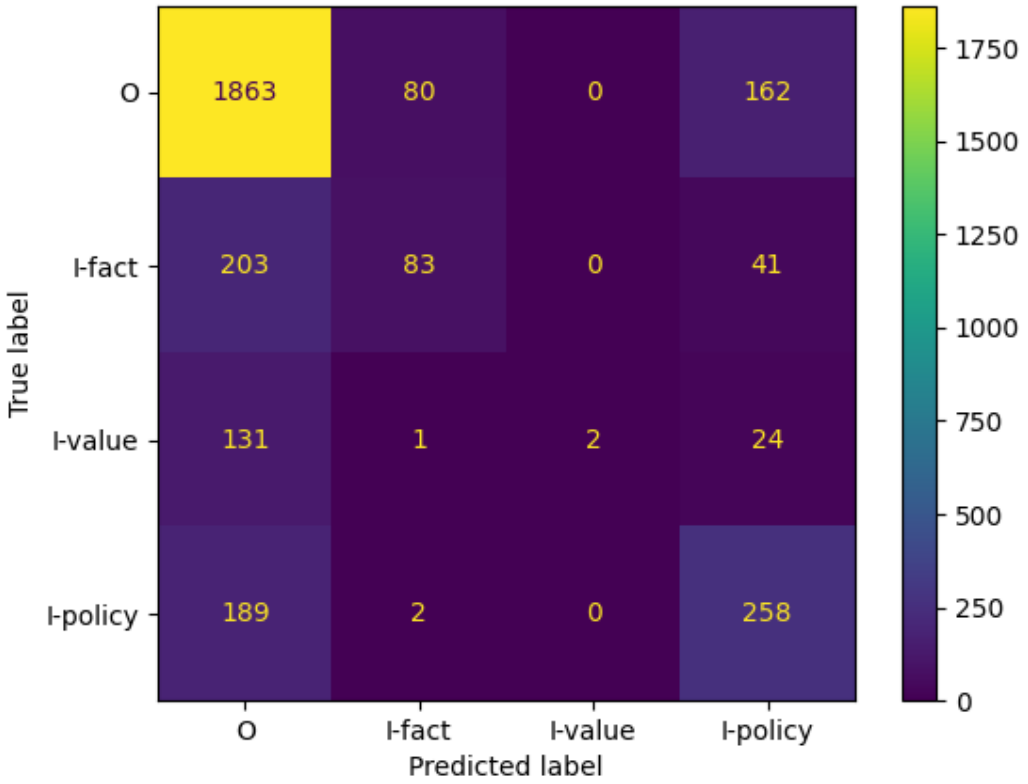


Figura 13. Matriz de confusión del mejor modelo obtenido en el experimento n°2: IO para la predicción del Tipo de Conclusión. Observamos como se tiende a predecir de forma incorrecta las diferentes clases como I-policy, por tratarse del tipo predominante. El tipo de Valor obtuvo el peor rendimiento, pero dichas predicciones fueron correctas. Por otro lado, el tipo de Hecho fue mal clasificado cerca de la mitad de las veces con la clase O. El tipo de Política, presenta también esta tendencia a clasificar la mitad de las palabras como O.

Algo que observamos examinando las predicciones de este modelo, es lo que parece una lexicalización sobre la palabra *#BuildThatWall* con la clase I-*policy*, como ocurría con otras categorías vistas.

Observamos que de 26 ocurrencias, 18 (69.23%) de ellas eran clasificadas como *I-policy*, el resto como *O*. De estas 18 predicciones sobre el hashtag *#BuildThatWall*, 9 (50%) de ellas fueron un error.

Calculando la IMP entre esta palabra y categoría, se obtiene un valor de -3.82 el cual es un menor a la media entre la categoría con las demás palabras con un valor de -3.80, debido a que esta clasificación comprende más de una palabra, esta discrepancia de 0.02 no es del todo significante por lo que podemos concluir en que el modelo se ve pegado al léxico de ciertas palabras al momento de clasificar el tipo de conclusión como *Política*. Esta lexicalización no se vio en los modelos de *Tipo de Justificación*.

(Furman et al., 2022) demuestra en su trabajo que suministrando al modelo de lenguaje RoBERTa información sobre la presencia de componentes como *Justificación* y *Conclusión* aumenta en cerca de 0.20 puntos la métrica macro-F1.

Con nuestra arquitectura experimentaremos algo similar utilizando una técnica denominada *en Cascada* donde buscaremos entrenar un clasificador que cuente con la información de presencia de estos componentes para realizar una predicción de tipos más correcta.

Discusión de los resultados

En base al análisis realizado observamos que la regresión logística, el *baseline*, es peor que la mejor aproximación neuronal, pero no por mucho, sobre todo utilizando BOW. Esto hace que un clasificador lineal, poco costoso en recursos como energía y tiempo, sea una opción viable a considerar.

El modelo neuronal más costoso y novedoso, RoBERTa, obtuvo mejor performance en mayoría de las tareas de clasificación pero no de forma desproporcionada, lo que abre paso a seguir considerando arquitecturas como las utilizadas en este trabajo pues presentan un mucho menor coste computacional que los modelos de lenguaje.

Hilando más profundamente por las categorías desarrolladas observamos sistemáticamente que tanto el *Pivote* como la *Propiedad* fueron las categorías más arduas de predecir, el mayor problema que presenta el *Pivote* es que probablemente existan muchas formas de decir lo mismo lo que dificulta la predicción de esta categoría, mientras que la *Propiedad* cuenta con pocos ejemplos que produce una lexicalización constante de los modelos que afectan negativamente al desempeño por el tipo de componente tan versátil.

El *Colectivo* por otra parte es una tarea mucho más fácil, también se encuentra lexicalizado lo que generó en muchos casos la predicción correcta del colectivo en el tweet cuando fue anotado como no argumentativo, seguramente se beneficiaria de técnicas para evitar el *overfitting*, aunque las estrategias más obvias (como el *dropout*) no tuvieron un impacto positivo.

Las categorías de *Justificación* y *Conclusión* tuvieron un buen desempeño en general, consiguiendo superar a RoBERTa en esta última. No se observó lexicalización fuerte de los modelos sobre la composición léxica de las palabras pero sí una tendencia favorable a nivel estructural de los ejemplos cómo sería la alta probabilidad que la justificación se encuentre al inicio y se componga, por lo general, de 13 palabras. Además en *Justificación* se obtuvieron los mejores resultados en casi todos los modelos pasando los 0.55 puntos de la métrica F1 de clase minoritaria. La *Conclusión* por otro lado, también obtuvo un buen desempeño general aunque tiende a predecir varias palabras separadas como conclusiones producto posiblemente de la gran cantidad de ejemplos donde tan solo un hashtag acarrea toda la conclusión del tweet.
Seguramente estos modelos pueden ser muy útiles en tareas *downstream* de generación de contranarrativas.

En cuanto a predicción de tweets *No Argumentativos*, fue otra categoría donde se obtuvo mejor desempeño que el modelo de lenguaje. Se observaron falencias en aprovechar propiedades estructurales como la longitud del ejemplo, pues tweets no argumentativos suelen ser más cortos que aquellos que sí lo son, además de realizar predicciones incompletas o palabras sueltas.

En cuanto a los *Tipos* ya sea de *Justificación* o *Conclusión*, se observó un overfitting por la clase predominante, de *Hecho* para la primera y *Política* para la segunda. Se requeriría una mayor cantidad de ejemplos para poder balancear estas clases y seguramente obtener mejores predicciones en los tipos minoritarios respectivos. Aunque hay que recordar que los tipos de *Hecho* y *Valor* fueron incluso difíciles de diferenciar entre los anotadores.

En cuanto a la utilización de diferentes seteos de la arquitectura, resultando en arquitecturas teóricamente diferentes, observamos que tanto el *Multitask* como en *Cascada* no tuvieron buen desempeño a comparación de la red neural por sí misma que, junto al incremento de recursos requeridos, descartamos su viabilidad. Específicamente para la técnica en *Cascada* observamos que debido al pobre desempeño general de las predicciones, acarrear estos errores por las distintas fases de la arquitectura impacta negativamente las predicciones finales.

Cerramos esta sección recordando que tan solo contamos con 999 tweets, un corpus muy pequeño pero con el cual obtuvimos resultados medianamente aceptables, podría ser peor. Claramente todas las categorías se beneficiarían ampliamente de contar con más ejemplos en total.

Conclusiones

El discurso de odio es una realidad que experimentamos casi de forma diaria, donde las redes sociales son un punto clave del individuo.

La censura, una técnica sencilla pero que abusa de la libertad de expresión, ha sido la preferencia para tratar este problema. A lo largo de los años notamos como en muchas casos a falta de una visión clara sobre el cual aplicar esto ha provocado una sobre censura en tópicos específicos, llegando incluso a abusar de ella por intereses propios.

En los últimos años las contranarrativas están tomando importancia en este contexto como una técnica de moderación. Estas dejan en evidencia el discurso de odio y sirven para educar a aquellos individuos que alguna vez lo inicializan. Un problema clave que enfrenta este camino es la redacción de ellas, algo que requiere de conocimiento tanto del tema que trata el discurso de odio como de abordajes a la solución.

En el auge del aprendizaje automático se trabaja de forma constante para encontrar modelos que sean capaces de realizar dicha redacción algo que de momento no es del todo satisfactible debido a la poca cantidad de ejemplos que se cuentan para ello y la superficialbilidad de los modelos. El problema de generación de contranarrativas no viene acompañada de un interés social masivo, ni siquiera industrial.

Debido a la escasez de ejemplos, diferentes investigadores han buscado formas de solventar el problema, desde generación de nuevos ejemplos a partir de modelos entrenados con los originales, obtención de discursos de odio de forma masiva pero no de calidad o experimentación ardua con los ejemplos que se cuentan para hallar aquel modelo que satisface la generación de contranarrativas bajo esta limitación.

En este trabajo utilizamos un dataset que optó por otra estrategia, anotar los escasos discursos de odio con información valiosa y de calidad que permitan al modelo guiarse en la generación de las narrativas. Sin embargo, en este trabajo no se encaminó por la generación de contranarrativas mismas, sino que examinamos y evaluamos el desempeño de modelos resultantes a raíz de la información que proveen estas anotaciones de calidad.

Contribuciones

Utilizando una arquitectura BiLSTM altamente configurable, exploramos inicialmente la detección de *Colectivo* y, lo que nombramos, *JCnA* tarea que se compone en predecir aquellas palabras que conforman una *Justificación*, *Conclusión* o son *No Argumentativas* bajo un formato de entrada *BIO*. Este experimento fue muy útil para conocer la arquitectura y un primer desempeño además de evaluar costos computacionales que acarrea diferentes configuraciones de hiper parámetros.

A raíz de esto, descartamos por ejemplo el apilamiento de redes BiLSTM para la predicción de una única componente pues no observamos una mejora significativa de los modelos pese a un incremento en recursos que requiere este seteo de arquitectura. También descartamos el formato *BIO* pues notamos que los modelos no podían ni siquiera

reproducirlo en sus predicciones, quizá debido a los pocos ejemplos con los que se cuenta. No obstante, manteniendo aún el formato *BIO*, exploramos la posibilidad de realizar un seteo manual de la capa *"linear chain"* CRF para solucionar este problema pero cuyos resultados no presentaron una mejoría significativa para plantearnos realizar un seteo de pesos individuales por categoría a predecir.

También evaluamos como los *word embeddings* de FastText impactaban sobre el desempeño de los modelos de predicción si provenían de un modelo FastText genérico entrenado en Wikipedia, o a partir de un modelo entrenado con tweets de Twitter, donde también fueron extraídos los ejemplos del dataset utilizado. Observamos un notorio incremento de la métrica F1 empleando este último modelo por lo cual optamos por su utilización en los demás experimentos en pos del modelo genérico.

Bajo un formato *IO* y separando la clase *JCnA* por sus componentes, exploramos el desempeño de los diferentes modelos entrenados para todas las categorías. De esta forma obtuvimos lo que serían las mejores métricas reportadas con la arquitectura utilizada. Hicimos una exploración exhaustiva de cada categoría observando una lexicalización presente en muchas de ellas de forma sistemática. Viendo así la necesidad de aplicar técnicas de *smoothing* para evitar el overfitting del modelo, optamos por variar diferentes valores y tipos de *dropout* para evaluar su desempeño y confirmar si se puede, efectivamente, disminuir la dependencia léxica de los modelos frente a las palabras. Pero antes de pasar a dicho experimento, descartamos la utilización de *char embeddings*, puesto que estos *embeddings* requieren de un mayor coste computacional (montar una red LSTM o 1D-CNN en la arquitectura) y no reportan mejoras significativas en las métricas.

En la variación de *dropout* evaluamos tanto la no utilización, *naive dropout* y *dropout* variacional. La arquitectura contaba por defecto de un *dropout* variacional de 0,25 para las salidas entre de las unidades ocultas de la BiLSTM y de 0,25 para las salidas de la red, pues según los que implementan la arquitectura concluyeron que dicho *dropout* disminuia el impacto que podrían generar aquellos valores aleatorios con los cuales se inicializa una red como son, por ejemplo, los pesos de la misma.
En este experimento corroboramos esto viendo que los mejores resultados se obtuvieron con el *dropout* por defecto en tres de las seis tareas evaluadas, donde además las otras tres tuvieron una mejora de a lo sumo 0.02 puntos en la métrica F1. Debido a esto, mantuvimos el *dropout* por defecto y exploramos dos arquitecturas alternativas.

La primera de ellas consiste en el *Multitask*, que se trata del entrenamiento en paralelo de diferentes tareas compartiendo una representación en común para explotar de esta formas similitudes. Con esta arquitectura, observamos una búsqueda de equilibrio en ciertas categorías como lo fueron *Propiedad* y *Colectivo*, donde la métrica de la primera mejoró pero la de la segunda empeoró. En sí, ninguna categoría obtuvo un mejor desempeño que aquellas obtenidas con el experimento *IO* pero si observamos cómo ciertas categorías no impactaban en el entrenamiento de otras como es el caso de *No Argumentativo*, *Justificación* y *Conclusión*. Debido a los altos recursos computacionales que requiere esta arquitectura y los resultados desalentadores descartamos la predicción multiobjetivo.

La otra variación de la arquitectura utilizada fue en Cascada, que similar al apilamiento nativo de redes BiLSTM, se diferencian en el agregado de información que brinda una red a

la siguiente además de tratarse de diferentes tareas, siendo esta nueva información el resultado de la predicción realizada por la red anterior. De esta forma nuestro objetivo fue mejorar el desempeño de los modelos en la tarea de predicción de *Tipos*, ya sea de *Justificación* o *Conclusión*, motivados por resultados obtenidos con modelos de lenguajes desarrollado por (Furman et al., 2022) que se asemejan a esta técnica.

Los resultados obtenidos en Cascada no fueron alentadores, debido primordialmente al hecho de que la predicción de una componente depende enormemente de las predicciones anteriores realizadas. Con esto queremos decir que debido a que la predicción de componentes como son *Justificación* o *Conclusión* no obtuvieron desempeños extraordinarios, sus predicciones en muchos casos fueron incorrectas lo que impactó en las predicciones posteriores que debió realizar la siguiente red sobre de qué tipo se tratan. Es por ello que descartamos esta técnica, hasta el momento de hallar arquitecturas en el estado del arte para la predicción de componentes como son *Justificación* y *Conclusión*.

Bajo todo este camino de experimentos y pruebas, concluimos que un modelo neuronal presenta un mejor desempeño que uno lineal como fue el *baseline* de Regresión Logística. Además, observamos que entre los modelos neuronales dicho desempeño no presentó mucha diferencia pero si el coste computacional que requieren los modelos de lenguajes.

Observamos también una complejidad clara en la predicción de componentes como son el *Pivote* o la *Propiedad* debido a su naturaleza a diferencia del *Colectivo*, que generalmente si bien lexicalizado los modelos, realizaron predicciones correctas aun siquiera el ejemplo fuera anotado como no argumentativo, información que el mismo no contaba.

Las categorías que más palabras abarcan como son las que componen *JCnA* tuvieron un buen desempeño en general aunque se observó en ciertas ocasiones tendencias a sobre anotar palabras como pertenecientes a una componente como fue en el caso de la *Justificación*. Además, en *Justificación* y *Conclusión* la estructura genérica, analizada en profundidad en el análisis del dataset, permitió a los modelos basarse para mejorar su desempeño, sin lexicalizaciones claras, pero para el caso de *No Argumentativo* no fue del todo satisfactorio debido a ejemplos donde no se predijo un tweet completo como no argumentativo, tan solo se anotaba un conjunto de palabras.

La clasificación de tipos es, al momento, también compleja. Observamos un overfitting sobre la categoría de *Hecho* para el caso de *Tipo de Justificación*, y de *Política* para el caso de *Tipo de Conclusión*, las cuales son las clases predominantes. Llegando al punto, como observamos en la predicción del *Tipo de Justificación*, donde el mejor modelo predijo una única palabra como perteneciente a la clase del tipo de *Política*, y fue erróneo. Un aumento en los ejemplos de entrenamiento ayudaría esta clase buscando además un balance entre los diferentes tipos.

Como mencionamos la información de presencia y composición de categorías en un tweet deben poder guiar a un modelo en su generación de contranarrativas. Categorías como *Justificación* y *Colectivo* brindarian información confiable para conocer la comunidad objetivo del ataque y las creencias del hecho. La identificación de *Conclusión* y *No Argumentativo* pueden impactar negativamente por su desempeño mediocre pudiendo así generar contranarrativas en casos donde ni siquiera se debe plantear, o que el modelo

busque contrarrestar una conclusión incorrecta. Identificar la *Propiedad* y el paso lógico de una *Justificación* a una *Conclusión,* que llamamos *Pivote,* por el bajo desempeño al momento impactaran negativamente en la narrativa generada pues muy seguro que no se cuente con certeza qué *Propiedad* se le está asignado a la comunidad objetivo.

Trabajo futuro

Vemos aquí entonces un amplio abanico de posibilidades para seguir desarrollándo. La más clara de todas es la obtención y anotado de calidad de muchos más ejemplos para componer un corpus de mayor tamaño que beneficie a los modelos entrenados a partir de él. Incluyendo, quizás la búsqueda de un balance entre las distintas categorías para evitar problemas como ocurrió con los *Tipos de Justificación* y *Conclusión*.

Los tweets pueden ser obtenidos, idealmente, de forma manual para preservar las propiedades descritas en el dataset o manteniendo una observación constante de ellos si se automatiza, o utilizando técnicas de aumentación de datos con la desventaja de contener ruido y no incorporar ejemplos más diversificados.

Otra posibilidad es la búsqueda de una arquitectura específica para la predicción de estas categorías, de tal forma que pueda explotar tanto las características estructurales como gramaticales que se presentan en este dominio de tareas.

Si se desea mantener la arquitectura BiLSTM, explorar otras técnicas de *smoothing* para evitar el *overfitting* o evaluar intensamente el apilamiento de más de dos redes que no exploramos pero que se asegura un mejor desempeño de los modelos a un mayor coste computacional, claro está. Propiedades como el gradiente de la arquitectura no fueron explorados que junto a la cantidad de *epochs* de entrenamiento, son más caminos a experimentar para conocer el impacto de estos hiper parámetros en las tareas de predicción de unidades argumentativas.

Se encuentra también la posibilidad de explorar otros *embeddings*, idealmente profundamente contextuales obtenidos de modelos de lenguajes como BERT para comparar su desempeño frente aquellos obtenidos de modelos de aprendizaje profundo como los utilizados en este trabajo de ELMo, que en varias tareas obtuvieron el mejor desempeño.

Quizá aplicando la técnica de *Multitask*, sería interesante evaluar el desempeño de las tareas apiladas de predicción de *Colectivo* y *Part-Of-Speech tagging* notando como esta última tiene un impacto en la primera por tratarse generalmente de sustantivos. Además, queda pendiente evaluar el apilamiento de tareas cuando se encuentran las predicciones de *Tipos* aunque no esperamos una mejora en rendimiento, sería interesante conocer qué otras tareas impactan positivamente o no en estas.

Como observamos, la técnica en Cascada es muy dependiente en sus etapas por lo cual la descartamos hasta no contar mínimamente con un modelo en el estado del arte para las predicciones de *Justificación* y *Conclusión*. Sin embargo, si esto sucediera estaríamos frente a una tarea donde solo se clasifican las palabras etiquetadas como *Justificación* o *Conclusión*, mismamente. Se podría llevar a cabo así un experimento que simula esto,

donde se entrene a los modelos para predecir los tipos bajo las secciones de las sentencias etiquetadas como *Justificación* o *Conclusión*, para luego predecir las presentes en la partición de desarrollo y evaluar este desempeño.

Otra alternativa es la búsqueda de nuevos métodos para mejorar las categorías más complejas de predecir cómo podría ser el agregado de información, como de entidades nombradas por ejemplo, para así ayudar a los modelos en su entrenamiento y predicciones, o el subdividir estas categorías en subcategorías aunque de mal plantearlo podría dificultar el anotado e impactar drásticamente de forma negativa en los modelos.

También es interesante explorar nuevas métricas para evaluar y comparar los modelos, quizá siendo estas dependiente de la categoría a predecir. Un ejemplo de ello podría ser una métrica de *overlap* para las clases de *Justificación* y *Conclusión* para poder cuantificar y comparar la intersección de palabras que acierta un modelo con la anotación manual correspondiente a la categoría.

De cualquier forma, con los modelos presentes se pueden hacer pruebas, al menos exploratorias, de la generación de contranarrativas y evaluar cómo dicho modelo resultante podría beneficiarse de otros atributos que quizá no estamos teniendo en cuenta en este trabajo.

Anexos

Anexo I : Análisis del Dataset

En este trabajo se utilizó únicamente la partición de tweet en inglés que fueron un total de 999 ejemplos de los cuales 254 (25,43%) son no argumentativos.

Observamos que la longitud media de los tweets son de 165 caracteres con una desviación estándar de 70.37. En la siguiente Figura 14 se puede observar la distribución de frecuencias por longitud de tweet.

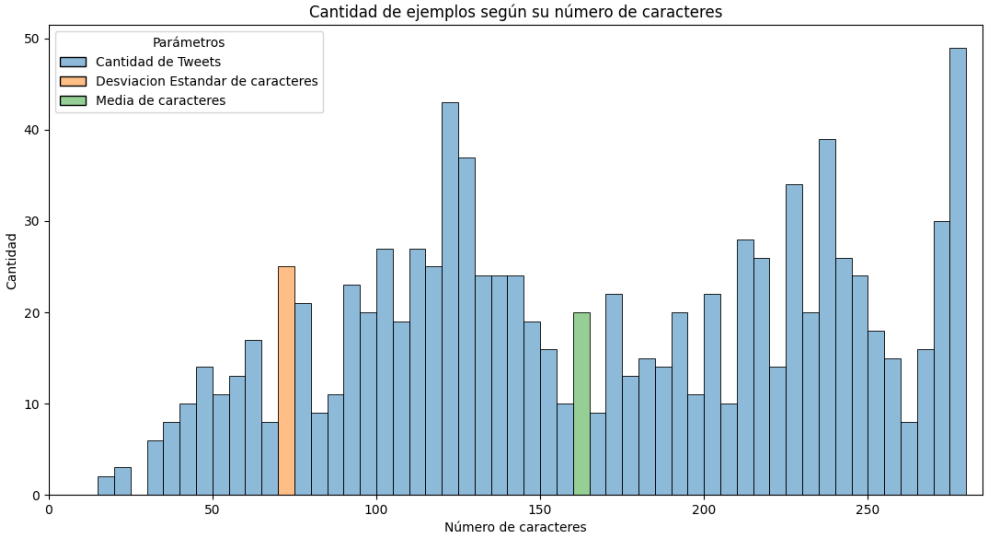


Figura 14. Distribución de tweets por longitud. Observamos que los tweets con menos de 100 caracteres son menos frecuentes, y que la mayor parte de tweets tienen entre 100 y 250 caracteres, con una distribución pareja entre diferentes longitudes.

Específicamente, vemos que en los ejemplos no argumentativos, junto a la Figura 15 se tiene una media de 119 caracteres por tweet con una desviación de 64.54. Esto nos indica que los tweets anotados como no argumentativos son generalmente más cortos en relación a los argumentativos.

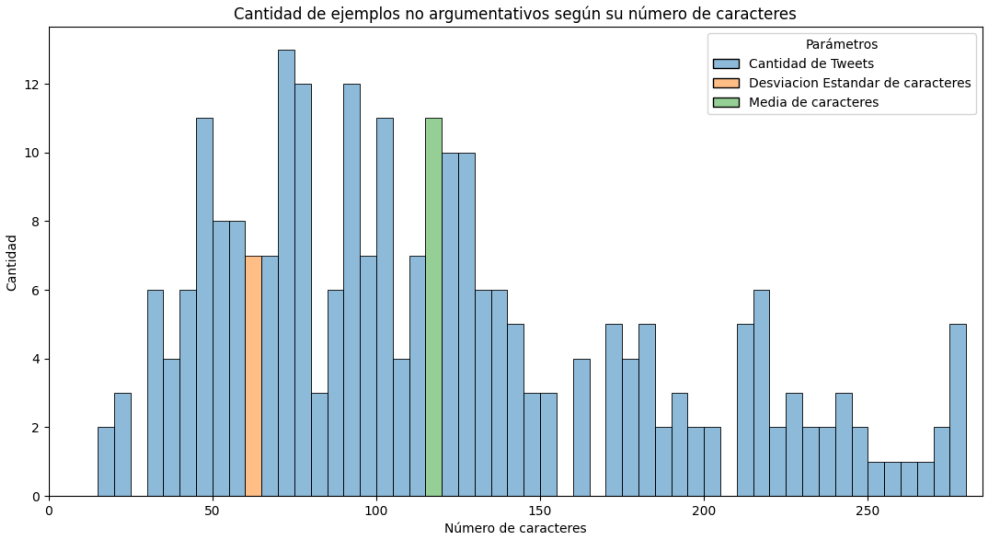


Figura 15. Distribución de tweets no argumentativos por longitud. Observamos que estos tweets presentan mayor frecuencia entre los 50 a 125 caracteres, mientras que tweets no argumentativos con más de 150 caracteres son muy infrecuentes.

De los 745 tweets argumentativos, en 436 (58,52%) ejemplos se identificó el *Colectivo*, en 433 (58.12%) la *Propiedad*, y en 428 (57.45%) se identificaron ambos en el mismo tweet.

Examinando el *Colectivo*, presenta una media de ocurrencia de 0.61 por tweet y 0.92 de desviación estándar, los valores más bajos entre todas las categorías, llegando a un total de 605 palabras anotadas donde *refugees* presenta la mayor ocurrencia con 49 (8.1%) seguido de *migrants* con 44 (7.22%) ocurrencias. En la siguiente Figura 16 podemos ver la cantidad de colectivos por tweet y su longitud media, en ella observamos cómo tweet argumentativos con longitud media no suelen contener un colectivo, pero van apareciendo a medida que el tweet aumenta la cantidad de palabras.

Notar que la presencia de más de 5 colectivos, es en general muy infrecuente, es por ello que su longitud media es alta, pero podemos ver la frecuencia total por colectivo en la Figura 17 donde vemos que en más de 300 tweets argumentativos, no se ha anotado el colectivo, y el otro caso extremo, 10 colectivos se vio únicamente en un único ejemplo:

> *@CBSNews Just deport all these illegal aliens and their kids back to their 3rd world nations. American citizens taxpayers are tired of being robbed by the 3rd world nation paying for these illegal aliens and their kids ! #MAGA #BuildTheWall #NoDACA #NoAmne*

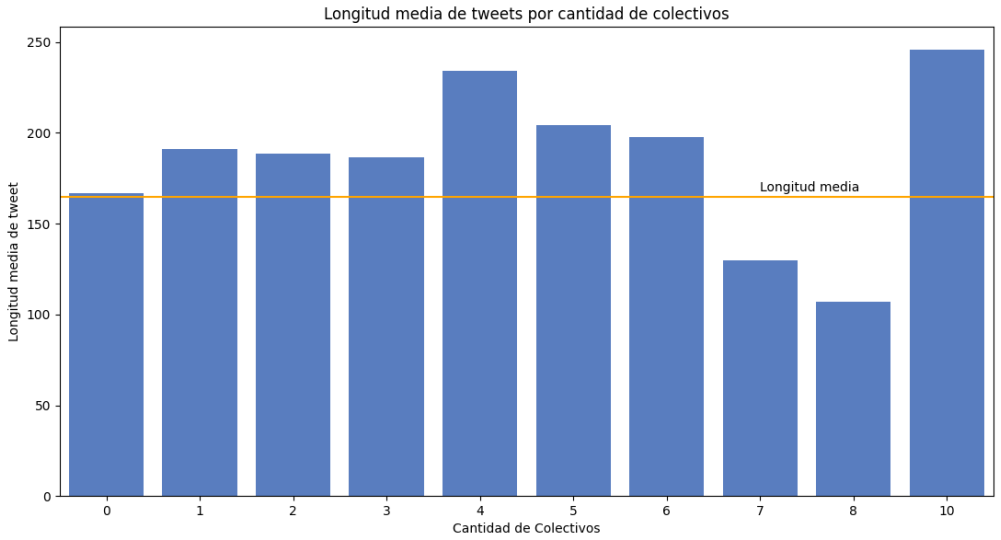


Figura 16. Distribución de cantidad de colectivos según longitud de tweets. Tweets de longitud media no presentan colectivo generalmente, pero a medida que esta longitud crece comienzan a evidenciarse al menos uno. Se observa un balance entre uno y tres colectivos de longitud similar, y entre cinco o seis con una longitud de tweet ligeramente mayor. Ejemplos con más de 7 son escasos, o únicos para el caso de 10 colectivos, motivo por el cual su longitud media es alta.

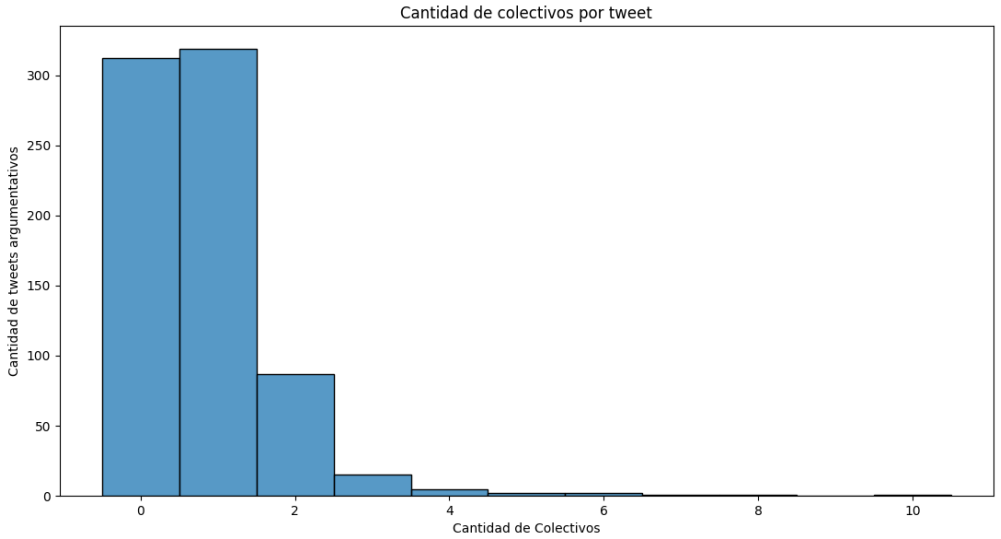


Figura 17. Histograma de cantidad de colectivos en el corpus utilizado. Cerca del 40% de los ejemplos no cuenta con colectivo, mientras que otro 40% aproximadamente el colectivo se compone de una única palabra; más palabras como colectivo por tweet son infrecuentes a partir de aquí.

En el caso de la *Propiedad*, se tiene una media de ocurrencias de 2.29 con 4.09 de desviación estándar llegando a ser anotadas 2291 palabras donde solo 93 (21.33%) de los ejemplos anotaron esta propiedad en el comienzo (dentro de las cinco primer palabras), y 87 (19.95%) se encontraron en el final del tweet (dentro de las últimas cinco palabras). Vemos entonces que en 256 (58.72%) de los tweets, la propiedad se encuentra en el medio de ella.

Omitiendo signos de exclamación, conectores, preposiciones y artículos, las tres palabras con mayor etiquetado de propiedad son *people*, *want*, y *crimes* con 12 (0.52%) ocurrencias en total cada una.

En la Figura 18 podemos ver la relación entre la media de propiedades y colectivos por longitud de tweet. Generalmente la cantidad de propiedades es mayor a la cantidad de colectivos media por longitud de tweet y a medida que el tweet se alarga, la cantidad media de propiedades aumenta, casi linealmente a partir de los 80 caracteres.

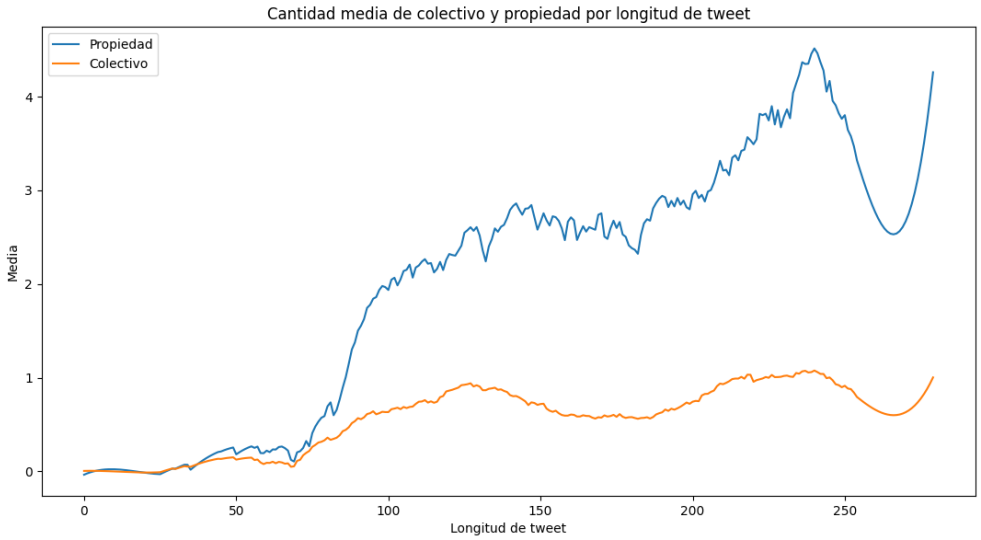


Figura 18. Distribución de Propiedad y Colectivo por longitud de tweet. Como se observó, el colectivo generalmente ronda entre una o dos palabras, pero vemos como la propiedad que se le asigna es baja en cantidad de palabras cuando el tweet se compone de menos de 100 caracteres, pero crece casi linealmente a partir de dicho punto.

En lo que a *Pivote* respecta, se anotaron ocurrencias en 334 (44.83%) ejemplos de los cuales en 217 (64.97%) se encontraba el *Colectivo* y en 215 (64.37%) se encontraban el *Colectivo* y la *Propiedad* simultáneamente junto al *Pivote*.

Esta clase presenta una ocurrencia media de 0.88 y desviación estándar de 1.56 por tweet, llegando a anotarse 887 palabras de las cuales, omitiendo signos de exclamación, *refugees* y *migrants* contaron con mayor cantidad de anotaciones como *Pivote* llegando a las 27 (3.04%) y 20 (2.25%) etiquetados, respectivamente. Además, comprobamos que en 114 (34.13%) ejemplos el pivote se encuentra al inicio del tweet y en otros 91 (27.25%) se encuentra en el final, concluyendo que en 129 (38.07%) el pivote se encuentra inmerso en el medio del tweet.

En la Figura 19 observamos la cantidad de pivotes media por longitud de tweet. Cuando el tweet contiene menos de la media de caracteres, hay una baja probabilidad de un pivote explícito, pero a medida que aumenta la longitud se vuelve más probable que se encuentre al menos una palabra como *Pivote*. La línea horizontal representa la media de pivotes por tweet.

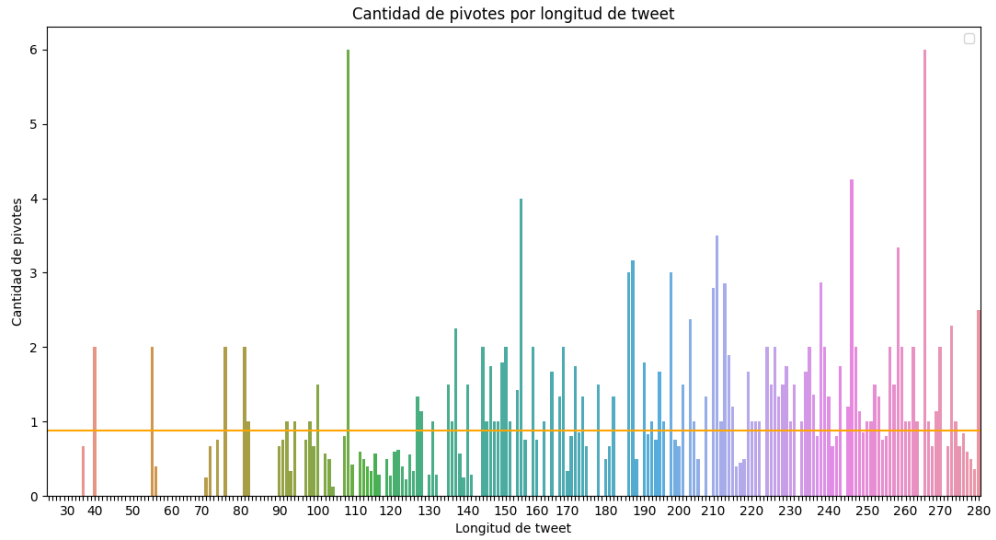


Figura 19. Histograma de pivotes por longitud del tweet. Al superar la longitud media de palabras, se observa una mayor frecuencia de contar con al menos una palabra anotada como pivote. Entre los ejemplos que cuentan con a lo sumo 70 caracteres, la presencia de un pivote es muy infrecuente.

Examinando las componentes argumentativas referidas a *Justificación* y *Conclusión*, vemos en la Figura 20 como la primera ocurre en la totalidad de tweets argumentativos mientras que la segunda en 737 (98.92%). Nunca sucede que un tweet sea anotado como conclusión sin antes previamente encontrarse al menos una justificación pero la inversa si sucede en 8 (1.07%) ejemplos, anotados por completo con solo *Justificación*.

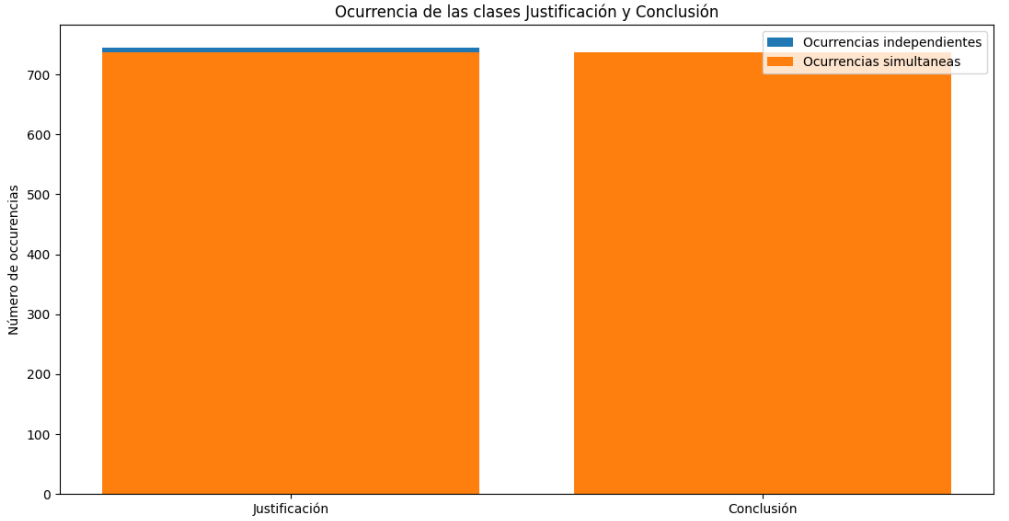


Figura 20. Histograma de tweets con componente de Justificación y Conclusión. Observamos como en 8 casos, el tweet cuenta con únicamente la componente justificativa pero no conclusiva, la inversa nunca ocurre.

En 512 (68.72%) tweets, la *Justificación* se encuentra al inicio del mismo, y en 317 (42.55%) al final (contando además aquellos tweets que solo se componen de *Justificación*), lo que es inverso en el caso de la *Conclusión* que en su mayoría con 469 (63.33%) ejemplos el tweet

concluye con ella y en tan solo 237 (32.16%) comienza con la misma. Además podemos observar lo siguiente con la Figura 21:

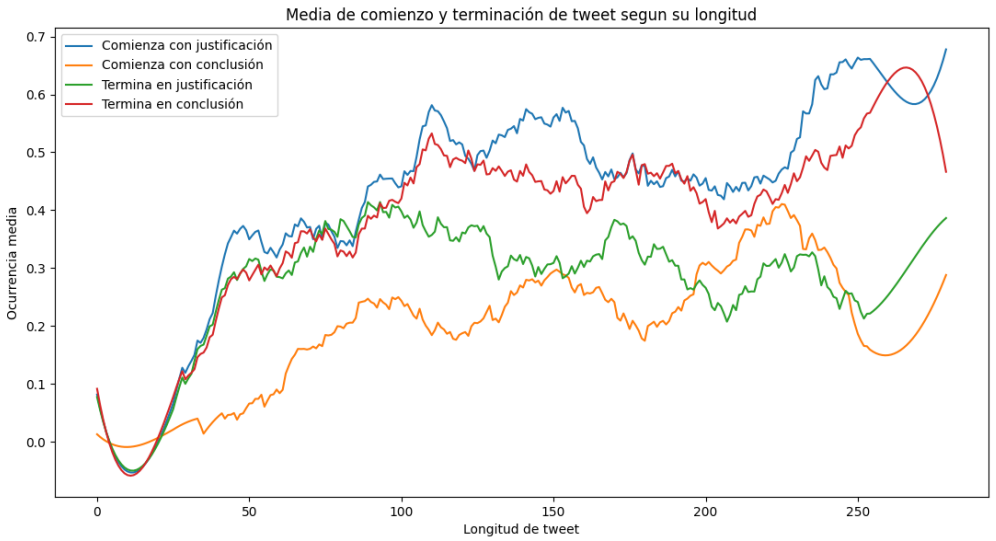


Figura 21. Distribución de ejemplos que comienzan o terminan con una componente justificativa o conclusiva. Los tweets suelen comenzar (primero cinco palabras) con una justificación y finalizar (últimas cinco palabras) con una conclusión. Sin embargo, a medida que la longitud crece observamos como la tendencia a comenzar con una conclusión aumenta, aunque la terminación en justificación decrece. Esta propiedad puede ser determinante para los modelos entrenados y su capacidad de explotarlos.

Vemos cómo a medida que aumenta la longitud del tweet, es más probable que este comience con una *Justificación* y que finalice en una *Conclusión*. También, las probabilidades de finalizar con una *Justificación* es en todos los casos menor a finalizar con una *Conclusión*, lo cual en la mayoría de los ejemplos es lo que sucede. Pero vemos como la tendencia a que un tweet comience con una *Conclusión* aumenta, ligeramente, a medida que el mismo crece en longitud.

Como la ocurrencia de *Justificación* abarca los 745 ejemplos argumentativos, su relación con las demás propiedades descritas al momento es equivalente.

Las ocurrencias de *Conclusión* en tweets vienen acompañadas 433 (58.75%) veces con un *Colectivo*, 331 (44.91%) veces con *Propiedad* y *Colectivo*, otras 331 (44.91%) veces con un *Pivote* explícito, y 212 (28.77%) veces se encuentran todas las categorías en simultáneo.

Concluimos viendo que la clase *Justificacion* presenta una media de 12.97 palabras por tweet y desviación de 11.27 llegando a anotarse 12960 palabras pertenecientes a esta categoría; para el caso de la *Conclusión* una media de 8.19 y desviación estándar de 8.58 anotando así un total de 8177 palabras.

Como mencionamos previamente se clasificaron, en lo posible, el *Tipo de Justificación* y *Conclusión* según sea de *Hecho*, *Valor* o *Política*.

En esta categoría, 667 (89.53%) ejemplos de *Justificación* fueron clasificados como de *Hecho*, 24 (3.22%) de *Valor* y 25 (3.36%) de *Política*. Vemos aquí un gran desbalance entre los tipos.

Justificaciones de *Hecho* abarcan una media de 17.68 palabras, de *Valor* 19.08 y de *Política* de 16.64, junto a una desviación estándar de 9.57, 7.08 y 8.72 palabras, respectivamente. Esto produjo así 11792 palabras justificativas como *Hecho*, 458 como *Valor* y 416 de *Política*. Notamos como las justificaciones de *Política*, si bien ligeramente más frecuentes, se componen de menos palabras.

En total, solo 259 conclusiones fueron anotadas como de *Hecho*, 44 de *Valor* y 422 de *Política*, al contrario que las justificaciones, vemos una tendencia de conclusiones a tomar acción sobre alguna medida ha haberse tratado de una opinión u hecho.

Las medias de palabras son de 13.37, 13.41 y 9.71 con desviaciones de 7.41, 7.75 y 8.43 para los tipos de *Hecho*, *Valor* y *Política*, respectivamente, lo que culmina en 3464, 590 y 4097 palabras anotadas según cada categoría nombrada en orden.

Anexo Ⅱ: Ejemplo de formato de los tweets

.txt

two children washed up on a barb wired beach another scum photographer another attempt to get the EU to take more migrants enough propaganda

.ann

T1 Premise2Justification 0 70 two children washed up on a barb wired beach another scum photographer
T2 Premise1Conclusion 71 140 another attempt to get the EU to take more migrants enough propaganda
T3 Collective 114 122 migrants
T5 Property 130 140 propaganda
T4 pivot 0 44 two children washed up on a barb wired beach
T6 pivot 114 122 migrants
R1 related Arg1:T4 Arg2:T6
A1 QuadrantType T2 policy
A2 QuadrantType T1 fact
T7 CounterNarrativeA 0 1 t
#1 AnnotatorNotes T7 Are you implying that someone drowned two little children on purpose as an attempt to get the EU to take more migrants? Or are you implying that the news of their dead should not be covered because it´s propaganda?
T8 CounterNarrativeB 1 2 w
#2 AnnotatorNotes T8 Are you blaming immigrants for the dead of two of their children calling it ¨propaganda¨?
T9 CounterNarrativeC 2 3 o
#3 AnnotatorNotes T9 Why are you calling ¨scum photographer¨ to someone who wanted to show the world this horrible tragedy?

.conll

Palabra	JCnA	Colectivo	Propiedad	Pivote	Justificación	Conclusión	No Arg.	Tipo de Just.	Tipo de Concl.
two	I-justification	O	O	I-pivot	I-justification	O	O	I-fact	O
children	I-justification	O	O	I-Pivot	I-justification	O	O	I-fact	O
washed	I-justification	O	O	I-Pivot	I-justification	O	O	I-fact	O
up	I-justification	O	O	I-Pivot	I-justification	O	O	I-fact	O
on	I-justification	O	O	I-Pivot	I-justification	O	O	I-fact	O
a	I-justification	O	O	I-Pivot	I-justification	O	O	I-fact	O
barb	I-justification	O	O	I-Pivot	I-justification	O	O	I-fact	O
wired	I-justification	O	O	I-Pivot	I-justification	O	O	I-fact	O
beach	I-justification	O	O	I-Pivot	I-justification	O	O	I-fact	O

another	I-justification	O	O	O	I-justification	O	O	I-fact	O
scum	I-justification	O	O	O	I-justification	O	O	I-fact	O
photographer	I-justification	O	O	O	I-justification	O	O	I-fact	O
another	I-Conclusion	O	O	O	O	I-Conclusion	O	O	I-policy
attempt	I-Conclusion	O	O	O	O	I-Conclusion	O	O	I-policy
to	I-Conclusion	O	O	O	O	I-Conclusion	O	O	I-policy
get	I-Conclusion	O	O	O	O	I-Conclusion	O	O	I-policy
the	I-Conclusion	O	O	O	O	I-Conclusion	O	O	I-policy
EU	I-Conclusion	O	O	O	O	I-Conclusion	O	O	I-policy
to	I-Conclusion	O	O	O	O	I-Conclusion	O	O	I-policy
take	I-Conclusion	O	O	O	O	I-Conclusion	O	O	I-policy
more	I-Conclusion	O	O	O	O	I-Conclusion	O	O	I-policy
migrants	I-Conclusion	I-Collective	O	I-Pivot	O	I-Conclusion	O	O	I-policy
enough	I-Conclusion	O	O	O	O	I-Conclusion	O	O	I-policy
propaganda	I-Conclusion	O	I-Property	O	O	I-Conclusion	O	O	I-policy

Anexo Ⅲ:

Resultados ampliados

En las siguientes Tablas 12, se reportan los mejores resultados para cada categoría en todas las arquitecturas empleadas en este trabajo. Se puede entender como un resumen de los mejores modelos obtenidos según la arquitectura recordando que RoBERTa fue evaluada en (Furman et al., 2022).

Se reporta la métrica de *accuracy* total, *precisión*, *recall* y F1 de clase minoritaria, junto al *macro*, *micro* y *average* F1, en dicho orden. Para el caso de *Tipo de Justificación* y *Tipo de Conclusión*, se omite las métricas de clase minoritaria, por lo cual su *precisión* y *recall* son según *macro*.

	No Argumentativo						
	Acc	Pr	Rec	F1	macro-F1	micro-F1	avg-F1
Regression Logistica c/emb	0,87	0,25	0,01	0,01	0,47	0,87	0,81
Regression Logistica BOW	0,85	0,31	0,12	0,17	0,54	0,85	0,82
BiLSTM	0,86	0,48	0,56	0,52	0,72	0,86	0,87
Multitask	0,87	0,56	0,30	0,39	0,66	0,87	0,86
Cascada	-	-	-	-	-	-	-
RoBERTa	-	0,50	0,57	**0,53**	-	-	-

Tabla 12 a. Observamos cómo la arquitectura BiLSTM empleada para la predicción individual triplica aproximadamente el desempeño de las regresiones en cuanto a clase minoritaria, pero se encuentra a tan solo 0.01 puntos del modelo de lenguaje, despreciable.

	Justificación						
	Acc	Pr	Rec	F1	macro-F1	micro-F1	avg-F1
Regression Logistica c/emb	0,62	0,57	0,64	0,60	0,62	0,62	0,62
Regression Logistica BOW	0,63	0,61	0,53	0,57	0,62	0,63	0,63
BiLSTM	0,74	0,68	0,78	0,73	0,74	0,74	0,74
Multitask	0,75	0,71	0,73	0,72	0,75	0,75	0,75
Cascada	-	-	-	-	-	-	-
RoBERTa	-	0,76	0,82	**0,79**	-	-	-

Tabla 12 b. En esta categoría, todas las arquitecturas presentaron resultados competentes. Destacamos las regresiones logísticas que presentan cerca de 0.2 puntos de menor desempeño, según clase minoritaria, sobre el modelo de lenguaje pero cuyo coste computacional es significativamente menor. La arquitectura de aprendizaje profundo, también más eficiente que el RoBERTa presenta tan solo 0.06 puntos de diferencia, un valor muy pequeño a una gran diferencia de coste.

	Conclusión						
	Acc	Pr	Rec	F1	macro-F1	micro-F1	avg-F1
Regression Logistica c/emb	0,71	0,43	0,02	0,03	0,43	0,71	0,61
Regression Logistica BOW	0,70	0,44	0,22	0,29	0,55	0,70	0,66
BiLSTM	0,78	0,63	0,53	**0,58**	0,71	0,78	0,77
Multitask	0,76	0,65	0,50	0,56	0,70	0,76	0,75
Cascada	-	-	-	-	-	-	-
RoBERTa	-	0,58	0,44	0,52	-	-	-

Tabla 12 c. La arquitectura BiLSTM presenta el mejor desempeño para la predicción de esta categoría sobre 0.06 para el modelo de lenguaje y 0.02 para la estrategia multiobjetivo. Presentando además casi el doble de mejoría sobre las regresiones logísticas.

	Tipo de Justificación					
	Acc	macro-Pr	macro-Rec	macro-F1	micro-F1	avg-F1
Regression Logistica c/emb	0,61	0,30	0,30	0,30	0,61	0,59
Regression Logistica BOW	0,61	0,30	0,30	0,30	0,61	0,59
BiLSTM	0,68	0,41	0,38	**0,39**	0,68	0,68
Multitask	-	-	-	-	-	-
Cascada	0,70	0,58	0,36	0,38	0,70	0,67
RoBERTa	-	0,34	0,32	0,36	-	-

Tabla 12 d. Resultados parejos entre las arquitecturas, notando que se trata del macro-F1. BiLSTM presenta el mejor desempeño, cuando Cascada por su naturaleza de dependencia lo empeora ligeramente aunque con mejor calidad de modelo por su precisión. El modelo de lenguaje no aparenta ser la mejor estrategia a implementar para esta categoría por el coste que conlleva y los resultados tabulados.

	Tipo de Conclusión					
	Acc	macro-Pr	macro-Rec	macro-F1	micro-F1	avg-F1
Regression Logistica c/emb	0,69	0,38	0,25	0,21	0,69	0,57
Regression Logistica BOW	0,67	0,32	0,29	0,29	0,67	0,60
BiLSTM	0,73	0,70	0,43	0,44	0,73	0,69
Multitask	-	-	-	-	-	-
Cascada	0,72	0,40	0,34	0,34	0,72	0,65
RoBERTa	-	0,48	0,51	**0,50**	-	-

Tabla 12 e. En esta categoría, el modelo de lenguaje presenta el mejor desempeño con una diferencia de 0.06 puntos de macro-F1 con la red BiLSTM, aunque la última presenta una alta precisión. La técnica en Cascada, por su lado, presenta un alto declive de desempeño frente a la detección individual.

	Colectivo						
	Acc	Pr	Rec	F1	macro-F1	micro-F1	avg-F1
Regression Logistica c/emb	0,98	0,50	0,07	0,13	0,56	0,98	0,97
Regression Logistica BOW	0,98	0,43	0,04	0,08	0,53	0,98	0,97
BiLSTM	0,98	0,64	0,43	0,52	0,75	0,98	0,98
Multitask	0,98	0,54	0,42	0,47	0,73	0,98	0,98
Cascada	-	-	-	-	-	-	-
RoBERTa	-	0,65	0,55	**0,59**	-	-	-

Tabla 12 f. Los modelos neuronales presentan casi el cuádruple de desempeño frente a las regresiones logísticas. Encontrándose el modelo de lenguaje y la arquitectura de aprendizaje profundo diferenciados por 0.07 puntos según métrica F1 de clase minoritaria. Esta categoría estuvo presente en cerca del 60% del corpus empleado, por lo que observamos que RoBERTa es capaz de explotar ligeramente mejor la presencia de esta categoría en su entrenamiento para sus predicciones.

	Propiedad						
	Acc	Pr	Rec	F1	macro-F1	micro-F1	avg-F1
Regression Logistica c/emb	0,92	0,52	0,04	0,08	0,52	0,92	0,88
Regression Logistica BOW	0,91	0,39	0,08	0,13	0,54	0,91	0,89
BiLSTM	0,91	0,46	0,25	0,32	0,64	0,91	0,90
Multitask	0,91	0,45	0,32	0,38	0,67	0,91	0,91
Cascada	-	-	-	-	-	-	-
RoBERTa	-	0,53	0,51	**0,52**	-	-	-

Tabla 12 g. RoBERta presenta un alto desempeño a comparación de las demás arquitecturas. Su enorme entrenamiento previo parece ayudar al modelo en la predicción de esta categoría luego de realizar el finetunning necesario. La arquitectura Multitask presenta una ligera mejoría de 0.06 puntos frente a la predicción individual, la única tarea que se vio beneficiada de esto, apilada junto a la categoría de Colectivo.

	Pivote						
	Acc	Pr	Rec	F1	macro-F1	micro-F1	avg-F1
Regression Logistica c/emb	0,97	0	0	0	0,49	0,97	0,95
Regression Logistica BOW	0,97	0	0	0	0,49	0,97	0,95
BiLSTM	0,96	0,29	0,13	0,18	0,58	0,96	0,96
Multitask	0,97	0,23	0,14	0,17	0,58	0,97	0,96
Cascada	-	-	-	-	-	-	-

RoBERTa	-	0,45	0,25	**0,32**	-	-	-

Tabla 12 h. La categoría más compleja, el Pivote. Las regresiones logísticas presentaron un nulo desempeño. La red BiLSTM presentó una mejoría a estas últimas, aunque no del todo satisfactorio, realizar un entrenamiento en paralelo con otras tareas generó peores predicciones por parte del modelo resultante. Al igual que en la categoría de Propiedad, el modelo de lenguaje RoBERTa presenta el mejor desempeño, duplicando el obtenido por la arquitectura de aprendizaje profundo.

Exploración de valores de *dropout*

Explorando desempeños, vemos de interés comparar los resultados obtenidos de los modelos BiLSTM de predicción individual frente a aquellos que obtuvieron mejor desempeño al momento de variar su máscara de *dropout*. Estos resultados se presentan en la siguiente Figura 22.

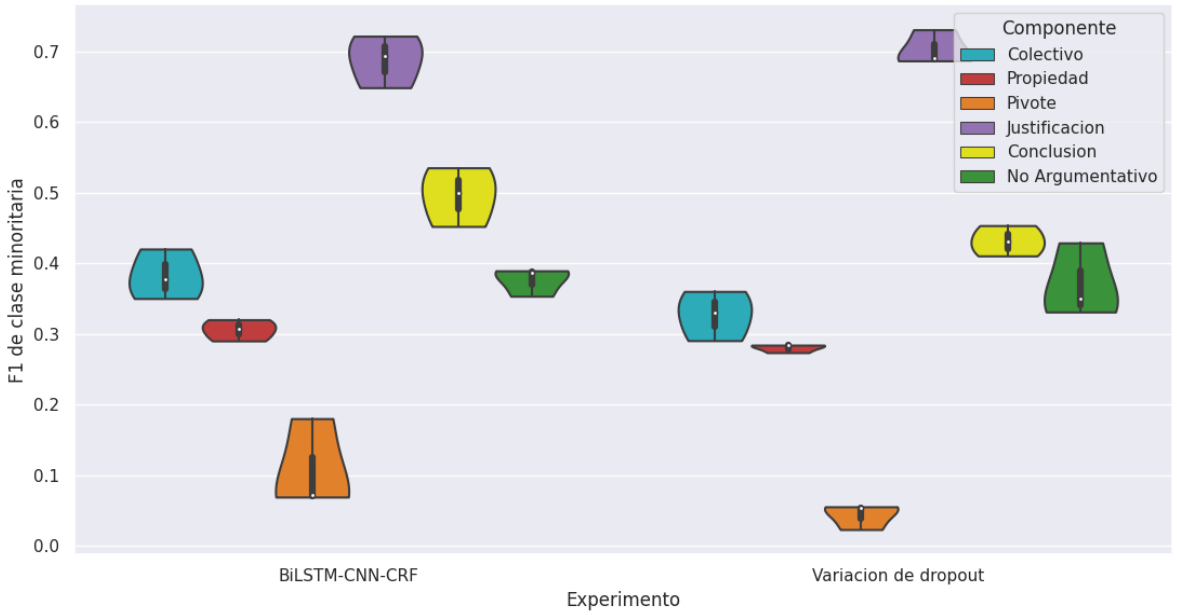


Figura 22. Gráfico de violines de los mejores modelos del experimento n°2: IO y n°3: Variaciones de dropout, bajo 3 exploraciones de la partición de testeo. Observamos que el desempeño de aplicar diferentes técnicas y valores de dropout culmina en una menor variabilidad del desempeño de cada categoría a costa de, en ciertos casos, un menor rendimiento. La categoría No Argumentativo no utiliza dropout en su violín a la derecha, por lo cual presenta una alta variabilidad de resultados a comparación del dropout por defecto que implementa la arquitectura utilizada.

Observamos cómo aplicando diferentes *dropouts*, la variabilidad de los modelos decremento en todos los casos exceptuando para el *No Argumentativo*, el cual no contó con la técnica de *dropout* pero aun así obtuvo una mayor F1 en su evaluación detallada en las próximas subsecciones.

Además de *No Argumentativo*, la categoría de *Justificación* también obtuvo un ligero mayor desempeño variando la técnica de *dropout*, utilizando aquí uno variacional de (0.5, 0.5).

Todas las demás, menor variación pero con peor desempeño. Observamos aquí que el *dropout* variacional de (0.25, 0.25) es efectivamente un cierto balance entre el desempeño final del modelo y los valores aleatorios con el cual se inicializa.

Examinando con mayor detenimiento, se tabularon los distintos desempeños en las siguientes Tablas 13 para los distintos tipos de dropout aplicados a la red.

Dropout		No Argumentativo			Justificación			Conclusión		
Tipo	Valor	Pr	Rec	F1	Pr	Rec	F1	Pr	Rec	F1
None	-	0,55	0,53	**0,54**	0,65	0,64	0,65	0,63	0,47	0,54
Naive	0.5	0,47	0,53	0,50	0,68	0,72	0,70	0,56	0,45	0,50
	0.75	0,39	0,57	0,46	0,68	0,74	0,71	0,60	0,51	0,55
Variacional	(0.25, 0.25)	0,48	0,56	0,52	0,68	0,78	0,73	0,63	0,53	**0,58**
	(0.5, 0.5)	0,58	0,41	0,48	0,68	0,84	**0,75**	0,57	0,59	**0,58**
	(0.5, 0.7)	0,62	0,42	0,50	0,69	0,72	0,70	0,61	0,53	0,57

Tabla 13 a. Resultados del experimento n°3: Variaciones de dropout. Se reportan métricas de clase minoritaria. Vemos como la categoría No Argumentativa se favorece ligeramente de no utilizar dropout debido, quizá, a la poca cantidad de ejemplos que existen pero para Justificación esto empeora su desempeño en gran medida. Además, esta última categoría y Conclusión se favorecen de un dropout variacional (0.5, 0.5) siendo su segundo mejor dropout (empatando para el caso de Conclusión) aquel por defecto de (0.25, 0.25). En estas categorías no se observa una mejoría significativa del desempeño con las diversas exploraciones realizadas. Notamos también cómo utilizar un dropout alto como el variacional de (0.5, 0.7) empeora el desempeño en las categorías de Justificación y Conclusión.

Dropout		Colectivo			Propiedad			Pivote		
Tipo	Valor	Pr	Rec	F1	Pr	Rec	F1	Pr	Rec	F1
None	-	0,49	0,39	0,44	0,48	0,21	0,29	0,22	0,11	0,15
Naive	0.5	0,48	0,41	0,44	0,46	0,26	**0,33**	0,21	0,11	0,14
	0.75	0,56	0,41	0,47	0,38	0,24	0,30	0,22	0,11	0,15
Variacional	(0.25, 0.25)	0,64	0,43	**0,52**	0,46	0,25	0,32	0,29	0,13	**0,18**
	(0.5, 0.5)	0,48	0,48	0,48	0,51	0,21	0,30	0,21	0,13	0,16
	(0.5, 0.7)	0,50	0,45	0,47	0,38	0,17	0,23	0,19	0,12	0,15

Tabla 13 b. Tanto la categoría de Colectivo como Pivote presentan más alto desempeño con el dropout por defecto, mientras que la Propiedad, de un naive dropout inicializado en 0.5, aunque esto representa una mejora de tan solo 0.01 sobre el dropout variacional (0.25, 0.25). Al igual que ocurre con las demás categorías, no utilizar dropout empeora los resultados, pero aplicar uno muy alto también lo empeora.

Viendo la tabla notamos que en tres propiedades a predecir (*Colectivo*, *Conclusión* y *Pivote*) el dropout utilizado previamente nos da el mejor resultado en base al macro-F1 (no presente en la Tablas 13 para evitar la sobrecarga de datos) donde además dos de ellos (*Colectivo* y *Pivote*) coinciden con el mejor F1 de la clase minoritaria y el otro (*Conclusión*), segundo mejor.

Para el caso de *Justificación* y *Propiedad*, el valor que maximiza el *dropout* es variacional de (0.5, 0.5) y *naive* inicializado en 0.5, respectivamente. Pero vemos en ambos, que el

segundo mejor se trata del que utiliza *dropout* variacional de (0.25, 0.25) tratándose a lo sumo de una diferencia de 0.08.

Para el caso *No Argumentativo* no utilizar *dropout* maximiza la métrica, esto puede deberse a la poca cantidad de ejemplos de esta clase que disminuyendo aún más, producto de un *dropout*, la red no consigue ajustar sus predicciones. De cualquier forma, el segundo mejor modelo utiliza un *dropout* variacional alto de (0.5, 0.7) con una diferencia de 0.02 puntos, seguido del por defecto (0.25, 0.25) con una diferencia al segundo mejor de 0.002.

Vemos así que la red BiLSTM sistemáticamente se ve beneficiada para la predicción de estas propiedades con un dropout variacional de (0.25, 0.25). Es interesante notar que efectivamente debemos utilizar un *dropout* para que el modelo no base sus predicciones tan fuertemente en el léxico de la palabra, pero tampoco debe ser muy alto de tal forma que se desprenda del mismo completamente. Notamos además que no encontramos una correlación alta entre los valores de *dropout* y el F1 de las clases minoritarias, siendo este valor -0.0236.

Es por ello, y debido a que ningún *dropout* de este experimento mejoró en gran medida (al menos 0.5 puntos) los macro-F1 y F1 minoritarios obtenidos en el experimento anterior, descartamos seguir explorando estas configuraciones y optamos por el *dropout* utilizado hasta el momento de tipo variacional de (0.25, 0.25).

Las clases donde se vio una fuerte lexicalización fueron de *Colectivo*, *Propiedad* y *Pivote*, naturalmente debido a sus características de tratarse en general de una o dos palabras. En las demás clases, no notamos esta lexicalización de forma tan significativa.

Nos es de interés examinar un poco la predicción de *Propiedad*, cual macro-F1 mayor utiliza un *dropout* distinto al del experimento anterior en el que notamos una lexicalización o apego del modelo sobre las palabras *crime* y *crimes*.

Aquí, las cinco (100%) ocurrencias de *crime* fueron clasificadas como *I-property* siendo una (20%) de ellas incorrecta. Para el caso de *crimes* el modelo predijo dos de las tres como *I-property* siendo ambas incorrectas. Dejamos aquí los tres casos de error en las predicciones:

> *What the White House has come up with has been shot down by Crime loving Democrats #BuildThatWall #MondayMotivation #BuildThatDamnWallNow*

> *Did you know? Government statitics of crimes committed by illegalsillegals comprise 7% of populationillegals account for 72% of all drug posseession33& of all money laundering29% of all drug traficking22% of all federal murders18%of all frauds”NODACA*

@bjdzyak @charlaynek @sabine_durden @IngrahamAngle Wrong! illegal immigrants make 5% of the population so of Course More Americans Commit Crimes because its more Citizens! Yet 34% Of Federal Prisoners are illegals these does not include State Prison

Notar como en el segundo ejemplo, a diferencia del experimento anterior, se clasifica la palabra crimes como una propiedad. Vemos entonces que este modelo utilizando un *naive dropout* inicializado en 0.5 si bien tiene buenas métricas macro-F1, se ve más fuertemente lexicalizado en las palabras *crime* y *crimes* que el experimento anterior.

Predicción multiobjetivo (Multitask)

Los resultados de este experimento se encuentran en la Tabla 14, en la misma se tabuló la pila de tareas en paralelo ejecutadas para el entrenamiento del modelo, así también como los modelos resultantes destinados a predecir cada una de las componentes de la pila.

Embedding	**Stack**	**Predicción**	Pr	Rec	F1
Komninos	Colectivo Propiedad	Colectivo	0,51	0,27	0,36
		Propiedad	0,42	0,30	0,35
	Justificación Conclusión	Justificación	0,67	0,72	0,69
		Conclusión	0,55	0,48	0,51
	Justificación Conclusión Pivote	Justificación	0,71	0,73	0,72
		Conclusión	0,59	0,39	0,47
		Pivote	0,19	0,10	0,13
	No Argumentativo Justificación Conclusión	No Argumentativo	0,55	0,28	0,37
		Justificación	0,67	0,72	0,69
		Conclusión	0,63	0,42	0,50
	No Argumentativo Justificación Conclusión Colectivo Propiedad Pivote	No Argumentativo	0,55	0,19	0,28
		Justificación	0,67	0,71	0,69
		Conclusión	0,58	0,47	0,52
		Colectivo	0,47	0,35	0,40
		Propiedad	0,41	0,27	0,33
		Pivote	0,08	0,06	0,07

Tabla 14. Resultados del experimento n°4: Multitask empleando word embeddings de Komninos. Se reportan métricas de clase minoritaria. Combinar las diferentes categorías no otorga mejor desempeño pero en ciertos casos como Colectivo y Propiedad vemos un balance, el primero mejora ligeramente su desempeño mientras que el segundo empeora. Vemos cómo incluir Justificación y Conclusión en diversas pilas de entrenamiento, no genera un cambio en sus desempeños más allá del declive que se genera al utilizar esta técnica. El pivote presenta una enorme baja en desempeño, la representación intermedia sobre la cual trabaja el modelo no beneficia esta categoría por se tan compleja en naturaleza.

Este experimento no arrojó mejores resultados de los obtenidos en el experimento *IO* en base a analizar tanto la F1 de las clases minoritarias como la macro-F1 del conjunto de las clases. Si observamos claramente como la información que brinda una tarea sobre la otra resulta en una búsqueda de equilibrio entre las F1 de las clases minoritarias.

Específicamente nuestro mejor resultado de clase minoritaria para la clasificación de *Propiedad* fue de 0.3221 y de *Colectivo* 0.5172, pero en el entrenamiento en Multitask se obtuvo 0.3564, lo que consiste en un aumento de 0.03, y 0.3467 lo que significa una disminución de 0.17 para las clases minoritarias, respectivamente.

Para *Justificación* y *Conclusión*, vemos un peor desempeño en ambas tareas perdiendo puntaje en su métrica F1 simultáneamente. Que junto al momento de incorporar el *Pivote* resulta en lo mismo.

Utilizando una pila de tareas *No Argumentativo*, *Justificación* y *Conclusión*, vemos una disminución de 0.15 puntos en la métrica F1 para la primera, de 0.2 para la segunda y tan solo 0.07 para la última. Así notamos como la incorporación de la tarea de clasificación de ejemplos no argumentativos el modelo final si bien pierde desempeño en su clasificación de las demás componentes, estas no se degradan aún más por la incorporación de dicha tarea. Por lo que la arquitectura no pudo explotar de forma eficiente la tarea de predicción *No Argumentativa*.

Al momento de contar con todas las tareas de predicción en la pila, excluyendo aquellas de tipos omitidas en este experimento por su complejidad en naturaleza dejado para el proximo y ultimo experimento, vemos un degrade en todas las tareas con un pico en una de las tareas más complejas, la predicción del *Pivote*. Notamos que si bien se ve una mayor disminución a lo mencionado en el párrafo anterior, las tareas de predicción de *Justificación* y *Conclusión* no presentan un degradé substancial en esta pila de entrenamiento total, por lo que observamos que la arquitectura no consigue explotar otros elementos de las demás tareas cuando ambas se encuentran en la pila de entrenamiento.

Todo lo mencionado anteriormente corresponde sin la utilización de la capa ELMo, una vez utilizada solo observamos que el degradé general de las tareas producto del entrenamiento en Multitask es menor, cerca de 0.06 puntos aproximadamente. Lo podemos observar en la Tabla 15.

Embedding	**Stack**	**Predicción**	Pr	Rec	F1
ELMo	Colectivo Propiedad	Colectivo	0,54	0,42	0,47
		Propiedad	0,45	0,32	0,38
	Justificación Conclusión	Justificación	0,68	0,69	0,69
		Conclusión	0,63	0,46	0,53
	Justificación Conclusión Pivote	Justificación	0,67	0,71	0,69
		Conclusión	0,63	0,48	0,54
		Pivote	0,23	0,14	0,17

	No Argumentativo Justificación Conclusión	No Argumentativo	0,56	0,30	0,39
		Justificación	0,69	0,74	0,71
		Conclusión	0,65	0,50	0,56
	No Argumentativo Justificación Conclusión Colectivo Propiedad Pivote	No Argumentativo	0,56	0,23	0,33
		Justificación	0,65	0,74	0,69
		Conclusión	0,58	0,51	0,54
		Colectivo	0,52	0,38	0,44
		Propiedad	0,32	0,20	0,25
		Pivote	0,16	0,08	0,11

Tabla 15. Resultados del experimento n°4: Multitask empleando word embeddings profundamente contextuales a partir de ELMo. Se reportan métricas de clase minoritaria. Se observa que el declive de desempeño es menor al utilizar embeddings de Komninos, pero no una mejoría a la predicción individual.

Además de buscar un mejor desempeño de los modelos en cuanto a nuestras métricas de interés, otro objetivo en este experimento era confirmar si el entrenamiento en Multitask ayudaría al modelo a predecir componentes cuando su par dependiente en la pila se encuentra presente también, es decir, en el caso de *Propiedad* y *Colectivo* buscamos que el *Colectivo* se encuentre presente siempre y cuando exista una *Propiedad* que se le asigna.

Pero este no es el caso, pues encontramos ejemplos como los siguientes, donde los modelos predicen un colectivo, pero no una propiedad, cuando ésta en el primer ejemplo, si se encuentra presente en la anotación manual.

More Than 130 Homes and 170 Square Miles Burned due to Wildfire Allegedly Started by Illegal Alien #Trump #MAGA #SendThemHome #RedNationRising https://t.co/b9Ut1x0V3D via @fairimmigration

And, last but not the least. That NRC issue is a political gimmick. No real action would be taken. No illegal immigrants are going to be deported. Go home. It's an AM:PM tiki taka! You are free to dance like a monkey, from either side.

Además, en el caso de la pila que comprende las tareas de predicción *No Argumentativo*, *Justificación* y *Conclusión*, vemos un solapamiento entre las predicciones de *No Argumentativo* con *Justificación* y de *No Argumentativo* con *Conclusión*, pero en ningún ejemplo las tres categorías se solapan entre sí.

Dejamos aquí ejemplos residuales donde el remarcado gris comprende el solapamiento de las categorías, los dos primeros se tratan de palabras sueltas clasificadas como *Justificación* y los ultimos como *Conclusión*:

PHOTOS: Trump's border wall under construction #BuildThatWall #BuildTheWall #WalkAwayFromDemocrats2018

“””Please don't call it “””rescue””” - it's human trafficking #portsclosed #sendthemback #benefitseekers”””

Our ancestors called migrants invaders we should do the same

@DiamonandSilk #BuldThatWall #BuildTheWall #BuildTheDamnWall NOW! AMERICA WANTS JUSTICE NOW #VoteDemsOut #VoteRed #Walkaway #SAVEAmerica

La implementación Multitask de la arquitectura viene dado por (Søgaard & Goldberg, 2016) donde los autores concluyen que la utilización de esta técnica presenta beneficios cuando las tareas a desarrollar son semejantes, como en sus pruebas de multitask combinando las tareas de *Part-Of-Speech tagging* y CCG que son de naturaleza sintáctica.

En nuestro caso, las tareas a predecir no son del todo semejantes debido a la diferentes naturalezas que cada categoría abarca lo cual nos ayuda a entender un poco mejor el motivo del bajo desempeño frente al entrenamiento individual.

Referencias

Abkenar, M. Y., & Stede, M. (2021). *Neural Argumentation Mining on Essays and Microtexts with Contextualized Word Embeddings*. 5.

Basile, V., Bosco, C., Fersini, E., Nozza, D., Patti, V., Rangel Pardo, F. M., Rosso, P., & Sanguinetti, M. (2019). SemEval-2019 Task 5: Multilingual Detection of Hate Speech Against Immigrants and Women in Twitter. *Proceedings of the 13th International Workshop on Semantic Evaluation*, 54–63. https://doi.org/10.18653/v1/S19-2007

Bojanowski, P., Grave, E., Joulin, A., & Mikolov, T. (2017). *Enriching Word Vectors with Subword Information* (arXiv:1607.04606). arXiv. http://arxiv.org/abs/1607.04606

Chernodub, A., Oliynyk, O., Heidenreich, P., Bondarenko, A., Hagen, M., Biemann, C., & Panchenko, A. (2019, August 21). *TARGER: Neural Argument Mining at Your Fingertips*. https://doi.org/10.18653/v1/P19-3031

Chung, Y.-L., Guerini, M., & Agerri, R. (2021). *Multilingual Counter Narrative Type Classification* (arXiv:2109.13664). arXiv. http://arxiv.org/abs/2109.13664

Cohen, J. (1960). A Coefficient of Agreement for Nominal Scales. *Educational and Psychological Measurement*, *20*(1), 37–46. https://doi.org/10.1177/001316446002000104

Cotik, V., Debandi, N., Luque, F., Miguel, P., Moro, A., Perez, J. M., Serrati, P., Zajac, J., & Zayat, D. (2020). *A study of Hate Speech in Social Media during the COVID-19 outbreak*. 6.

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *ArXiv:1810.04805 [Cs]*. http://arxiv.org/abs/1810.04805

Eger, S., Daxenberger, J., & Gurevych, I. (2017). *Neural End-to-End Learning for Computational Argumentation Mining* (arXiv:1704.06104). arXiv. http://arxiv.org/abs/1704.06104

Ezen-Can, A. (2020). *A Comparison of LSTM and BERT for Small Corpus*

(arXiv:2009.05451). arXiv. http://arxiv.org/abs/2009.05451

Furman, D. A., Torres, P., Rodriguez, J. A., Martinez, L., Alemany, L. A., Letzen, D., & Martinez, M. V. (2022). *Parsimonious Argument Annotations for Hate Speech Counter-narratives* (arXiv:2208.01099). arXiv. http://arxiv.org/abs/2208.01099

Huang, Z., Xu, W., & Yu, K. (2015). Bidirectional LSTM-CRF Models for Sequence Tagging. *ArXiv:1508.01991 [Cs]*. http://arxiv.org/abs/1508.01991

Jahan, M. S., & Oussalah, M. (2021). A systematic review of Hate Speech automatic detection using Natural Language Processing. *ArXiv:2106.00742 [Cs]*. http://arxiv.org/abs/2106.00742

Komninos, A., & Manandhar, S. (2016). Dependency Based Embeddings for Sentence Classification Tasks. *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 1490–1500. https://doi.org/10.18653/v1/N16-1175

Mensonides, J.-C., Harispe, S., Montmain, J., & Thireau, V. (2019). Automatic Detection and Classification of Argument Components using Multi-task Deep Neural Network. *Proceedings of the 3rd International Conference on Natural Language and Speech Processing*, 25–33. https://aclanthology.org/W19-7404

Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). *Efficient Estimation of Word Representations in Vector Space* (arXiv:1301.3781). arXiv. http://arxiv.org/abs/1301.3781

Pennington, J., Socher, R., & Manning, C. (2014). Glove: Global Vectors for Word Representation. *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 1532–1543. https://doi.org/10.3115/v1/D14-1162

Peters, M. E., Ammar, W., Bhagavatula, C., & Power, R. (2017). *Semi-supervised sequence tagging with bidirectional language models* (arXiv:1705.00108). arXiv. http://arxiv.org/abs/1705.00108

Peters, M. E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., & Zettlemoyer, L.

(2018). *Deep contextualized word representations* (arXiv:1802.05365). arXiv. http://arxiv.org/abs/1802.05365

Reimers, N., & Gurevych, I. (2017). Reporting Score Distributions Makes a Difference: Performance Study of LSTM-networks for Sequence Tagging. *ArXiv:1707.09861 [Cs, Stat]*. http://arxiv.org/abs/1707.09861

Reimers, N., & Gurevych, I. (2019). *Alternative Weighting Schemes for ELMo Embeddings* (arXiv:1904.02954). arXiv. http://arxiv.org/abs/1904.02954

Schaefer, R., & Stede, M. (2020). Annotation and Detection of Arguments in Tweets. *Proceedings of the 7th Workshop on Argument Mining*, 53–58. https://aclanthology.org/2020.argmining-1.6

Schaefer, R., & Stede, M. (2021). Argument Mining on Twitter: A survey. *It - Information Technology*, *63*(1), 45–58. https://doi.org/10.1515/itit-2020-0053

Søgaard, A., & Goldberg, Y. (2016). Deep multi-task learning with low level tasks supervised at lower layers. *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, 231–235. https://doi.org/10.18653/v1/P16-2038

Tekiroğlu, S. S., Chung, Y.-L., & Guerini, M. (2020). Generating Counter Narratives against Online Hate Speech: Data and Strategies. *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 1177–1190. https://doi.org/10.18653/v1/2020.acl-main.110

Los abajo firmantes, miembros del Tribunal de evaluación del presente Trabajo Especial, damos fe que el presente ejemplar impreso se corresponde con el aprobado por este tribunal.