Procesamiento del Lenguaje Natural, Revista nº 70, marzo de 2023, pp. 213-223 recibido 16-12-2022 revisado 29-01-2023 aceptado 07-02-2023

Generación y pesado de skipgrams y su aplicación al análisis de sentimientos

Skipgrams Generation and Weighting and its Application to Sentiment Analysis

Javi Fernández, Yoan Gutiérrez, Patricio Martínez-Barco
Department of Software and Computing Systems
University of Alicante
{javifm,ygutierrez,patricio}@dlsi.ua.es

Resumen: El modelado de skipgrams es una técnica para la generación de términos multi-palabra que conserva parte de la secuencialidad y flexibilidad del lenguaje. Sin embargo, en algunos casos el número de skipgrams generados puede ser excesivo a medida que se aumenta la distancia entre palabras. Además, esta distancia no suele ser tenida en cuenta a la hora de valorar los términos que se generan. En este trabajo proponemos una técnica para la generación y filtrado eficientes de skipgrams y un esquema de pesado que tiene en cuenta la distancia entre los términos, dando más importancia a aquellos más cercanos. Aplicaremos y evaluaremos estas propuestas en la tarea de análisis de sentimientos.
Palabras clave: skipgrams, generación de términos, pesado de términos, análisis de sentimientos.

Abstract: Skipgram modelling is a technique for generating multi-word terms that preserves some of the sequentiality and flexibility of the language. However, in some cases the number of skipgrams generated may become excessive as the distance between words increases. Moreover, this distance is often not taken into account when evaluating the terms that are generated. In this paper we propose a technique for efficient skipgram generation and filtering, and a weighing scheme that takes into account the distance between terms, giving more importance to those closer. We will apply and evaluate these proposals in the task of sentiment analysis.
Keywords: skipgrams, term generation, term weighting, sentiment analysis.

1 Introducción

La técnica del *modelado de skipgrams* consiste en obtener términos multi-palabra[1] a partir de un texto, de forma similar a los n-gramas, pero permitiendo omitir algunas palabras intermedias. Más concretamente, en un *k-skip-n-gram*, n determina el número de palabras de los términos generados, y k el número de palabras que se omiten. También se puede trabajar con un número máximo de palabras por término y con un número máximo de omisiones. En este trabajo nos diferenciaremos estos casos denominándolos n_{max} y k_{max} respectivamente. Con esta técnica estamos generando términos adicionales que conservan parte de la secuencialidad de las palabras originales, pero de forma más flexible que los n-gramas. Cabe señalar que los n-gramas pueden definirse como skipgrams en los que $k = 0$ (sin omisiones o saltos).

Sin embargo, la principal desventaja de esta técnica radica en que el número de skipgrams generados puede ser muy grande. Para hacernos una idea del tamaño máximo que podrían alcanzar, obtener todos los términos posibles utilizando n-gramas (de cualquier tamaño) tiene una complejidad de $O(n^2)$ (en este caso n es el número de palabras del texto), pero utilizando skipgrams (cualquier tamaño y cualquier número de palabras omitidas), la complejidad sería del orden de $O(2^n)$. En la Figura 1, se puede ver un ejemplo práctico utilizando uno de los conjunto de datos del TASS 2020 (Vega et al., 2020), concretamente el conjunto `train` en castellano (`es`) de la

[1] En este artículo denominaremos *términos* a una secuencia de palabras cuyo orden importa.

ISSN 1135-5948 DOI 10.26342/2023-70-17

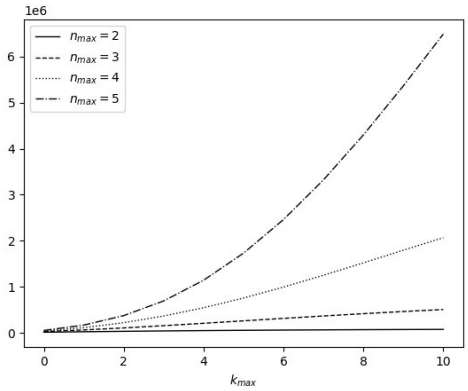

Figura 1: Número de términos generados utilizando skipgrams, según se va aumentando el valor de k_{max}, para diferentes valores de n_{max} máximo, en el conjunto de datos `train` de la tarea `1.1` del TASS 2020 en castellano.

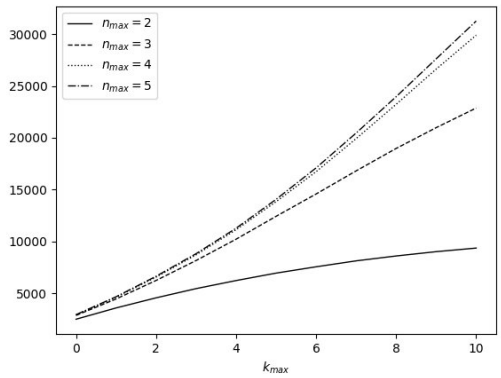

Figura 2: Número de términos generados utilizando skipgrams, según se va aumentando el valor de k_{max}, para diferentes valores de n_{max}, eliminando los que aparecen solo una vez en el conjunto de datos `train` de la tarea `1.1` del TASS 2020 en castellano.

tarea `1.1`, y generando términos con diferentes valores de n_{max} y k_{max}. Este es un conjunto de datos relativamente pequeño, con 1126 documentos y 4222 palabras diferentes, y en el caso de $n_{max} = 5$ y $k_{max} = 10$ máximos el número de términos generados casi alcanza los 6,5 millones.

Existen varios motivos para filtrar los términos generados, sobre todo en el contexto del aprendizaje automático. Por un lado, el número de skipgrams generados puede ser excesivo en algunos casos. Existen técnicas y modelos que nos sería imposible utilizar si disponemos de pocos recursos (procesamiento, memoria, espacio, tiempo) ya que no pueden manejar un número tan grande de términos o características. Por otro lado, reducir el número de términos también mejora el rendimiento de los sistemas que los utilizan, y puede disminuir el ruido si se seleccionan de la manera adecuada, obteniendo mejores resultados en ciertas tareas. Afortunadamente, una estrategia sencilla como eliminar los términos que aparecen solo una vez en el conjunto de datos puede reducir drásticamente la cantidad de términos, como se puede observar en la Figura 2, con la misma configuración que en el ejemplo anterior, donde el número de términos finales en el caso de $n_{max} = 5$ y $k_{max} = 10$ desciende hasta algo más de 30000 (una reducción de más del 99 % de términos).

No obstante, lo mencionado anteriormente no elimina el hecho de que para reducir el número de términos, primero debemos generarlos *todos* (sobretodo cuando los criterios del filtrado son estadísticos), algo que puede ser inviable en algunos conjuntos de datos o configuraciones. En este trabajo proponemos una técnica para realizar este filtrado de manera simultánea a la generación de los términos, con el objetivo de minimizar el número de generaciones realizadas y mejorar la eficiencia. Esta técnica se explicará en detalle en la Sección 3.1.

Por otra parte, la generación de términos utilizando el modelado de skipgrams ofrece una información valiosa que es frecuentemente ignorada: la distancia entre las diferentes ocurrencias del término. Saber si un término ha sido generado mayoritariamente por palabras adyacentes o cercanas, o al contrario por palabras más alejadas entre sí, es una información que podemos aprovechar tanto para la selección de términos como para la propia tarea en la que vamos a utilizar los términos. En este trabajo proponemos una forma de pesar los términos para aprovechar esta información, detallada en la Sección 3.2.

Este trabajo sigue la siguiente estructura. En la Sección 2 estudiaremos el uso de la técnica de modelado de skipgrams en la actualidad y la problemática asociada. La Sección 3 describirá las técnicas para la generación y pesado de términos mencionadas previamente. A continuación, en la Sección 4 aplicaremos las propuestas a la tarea de análisis de sentimientos en dos conjuntos de datos diferentes, para comprobar en qué medida es-

tas técnicas pueden influenciar los resultados. Finalmente, en la Sección 5 explicaremos las conclusiones sacadas de este trabajo y propondremos nuevas líneas para su continuación en el futuro.

2 Estado actual

La técnica del modelado de skipgrams tuvo auge hace años en el campo del PLN (Guthrie et al., 2006), y a día de hoy existen muchos enfoques que utilizan el modelado de skipgrams para relacionar y contextualizar palabras. Sin embargo, la mayoría de ellos solo contemplan las relaciones entre términos de par en par, utilizan esta técnica para crear un contexto, o siguen utilizando palabras o n-gramas (o un vector que representa una palabra) como unidades básica de información (Mikolov et al., 2013; Church, 2017; Vaswani et al., 2017; Zhao et al., 2017). En la actualidad, prácticamente todas las menciones a los skipgrams se refieren a su uso como contexto para generar *word embeddings* (Peng et al., 2020; Du et al., 2020; Santos et al., 2021).

Como hemos mencionado en la sección anterior, aumentar el número de términos u omisiones puede dar lugar a un número demasiado grande de combinaciones, por lo que a menudo no se aprovecha todo el potencial de esta técnica. El trabajo de Shazeer, Pelemans, y Chelba (2015) muestra como con valores grandes ($n = 5$, $k = 10$) se generan más de 60 mil millones de términos para algunos conjuntos de datos. Uno de los trabajos que más se ha centrado en generar skipgrams de manera eficiente es el de Gompel y van den Bosch (2016), donde en un primer paso se obtienen n-gramas con un filtrado progresivo, tras el cual los skipgrams son generados y filtrados a partir de esos n-gramas pero eliminando ciertas palabras intermedias. El método parece muy eficiente en términos espaciales y temporales. Sin embargo, el foco de este trabajo es la eficiencia en la generación y filtrado de n-gramas, no en los skipgrams, y no se estudia su repercusión en otras tareas. Otros trabajos, como por ejemplo los de Nguyen y Grishman (2016) o Hossny et al. (2020), explican brevemente que se generan skipgrams de manera eficiente pero no se dan detalles de la aproximación utilizada para hacerlo.

Respecto a aprovechar la información sobre la distancia para valorar los términos generados mediante el modelado de skipgrams, en la literatura podemos encontrar algunas aproximaciones, aunque no es algo común. Uno de los trabajos más enfocados en aprovechar esta información es el de Chang, Lee, y Lai (2017), en el que se propone una función gaussiana para valorar la relación de pares de palabras, con el objetivo de mejorar `word2vec` pero no para la generación de términos, aunque se muestra que los resultados mejoran al tener en cuenta esta información. Otros trabajos como Komninos y Manandhar (2016) o Mimno y Thompson (2017) mencionan que tienen en cuenta la distancia pero no se indica el método.

3 Propuesta

En este trabajo realizaremos dos propuestas. Por un lado proponemos una técnica para la generación y filtrado eficientes de skipgrams, con el objetivo de evitar la generación excesiva de términos. Por otro lado, diseñamos un esquema de pesado que tiene en cuenta la distancia entre las palabras utilizadas para crear los términos, que intenta dar mayor importancia a aquellos más cercanos.

3.1 Filtrado progresivo

Lo más común a la hora de generar términos a partir un conjunto de textos (crear una *bolsa de palabras*) es extraerlos todos luego elegir aquellos que dan más información utilizando diferentes técnicas o heurísticas. Sin embargo, como hemos visto previamente, el número de términos puede ser considerablemente grande en algunos casos, y procesar todos ellos puede requerir más recursos de los disponibles (tanto temporales y espaciales). Con el fin de obtener los términos más importantes de manera eficiente, intentando evitar tener que generar la totalidad de ellos, nuestro objetivo es filtrar los términos durante el propio proceso de generación.

Para ello aprovecharemos el hecho de que los términos con un cierto número de palabras tienen características en común con las palabras (u otros términos) que la forman. Por ejemplo, si tenemos un término multipalabra $t_1 = (w_1 w_2)$, en el que w_1 es la primera palabra y w_2 la segunda, podemos asegurar que t_1 aparecerá como máximo en tantos documentos como w_1 o w_2. Si nuestro criterio de filtrado es aparecer en un mínimo de documentos, si w_1 no lo cumple, entonces t_1 tampoco. Tampoco lo cumpliría ningún otro término derivado de t_1, como por ejem-

plo $t_2 = (w_1 w_2 w_3)$. Por lo tanto, sabiendo que w_1 no cumple este criterio, podemos evitar generar cualquier derivado, tanto t_1 como t_2, mediante un algoritmo de *ramificación y poda* (ver Figura 3), donde los términos generados se generan de izquierda a derecha a partir de otros generados previamente.

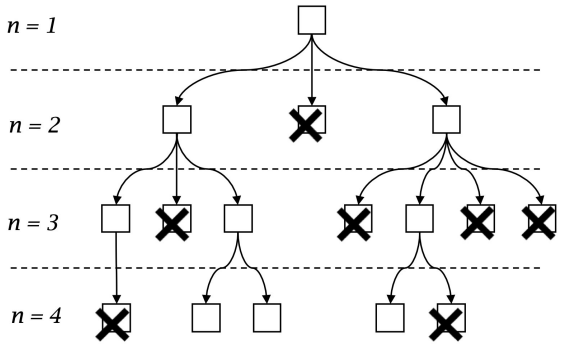

Figura 3: Ejemplo de ramificación y poda. Con una `X` se marcan aquellos términos que no seguían el criterio de filtrado, evitando la generación de todo un subárbol de términos derivados.

Sin embargo no todos los criterios de filtrado pueden hacerse durante la generación. Existen filtrados (la mayoría) que si se realizan durante la generación eliminarán más términos de los esperados, y nuestro objetivo es mejorar la eficiencia pero con el mismo resultado que filtrando al final. Por ejemplo, el porcentaje de ocurrencia en una categoría de un conjunto de datos etiquetado (normalmente denominado en estadística $P(t, c)$) no es candidato para realizarse de manera progresiva. Si un término $t_1 = (w_1 w_2)$ aparece un porcentaje de veces en la categoría c, no podemos saber si el término $t_2 = (w_1 w_2 w_3)$ aparecerá en más o en menos proporción hasta que no lo hayamos generado, por lo que este filtrado debería realizarse tras la generación. En el ejemplo de la Figura 4, si nuestro criterio de filtrado es que el 75 % de las ocurrencias de un término deben ser en documentos positivos, el primer término «qué» y el siguiente «qué máquina» no lo cumplirían, pero aún así no debemos dejar de generar el resto de términos ya que perderíamos el término «qué máquina eres» que sí que lo cumple y puede ser interesante.

Como ejemplos de criterios de filtrado progresivo podemos mencionar *a)* un mínimo de ocurrencias en el conjunto de datos, *b)* un mínimo de ocurrencias en una categoría concreta, *c)* combinaciones de palabras prohibidas,

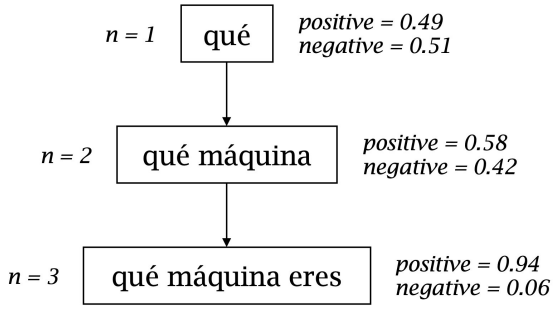

Figura 4: Ejemplo de generación de términos de hasta $n = 3$ palabras, donde *positive* el porcentaje de documentos positivos y *negative* el porcentaje de documentos negativos (ejemplo ficticio).

d) combinaciones de categorías gramaticales (PoS) prohibidas, etc. Como criterios de filtrado que no puede realizarse de manera progresiva podemos destacar *a)* una proporción mínima de ocurrencias en una categoría concreta, *b)* un umbral de puntuación obtenida a partir de un algoritmo de selección de características, etc. En el presente trabajo solo estudiaremos el mínimo de ocurrencias con el fin de mantener la aproximación simple y evitar propagar errores de herramientas externas en nuestro estudio.

En pseudocódigo para este procedimiento se puede observar en el Algoritmo 1, donde `Tokenize` se encarga de extraer las palabras de los textos, `Generate` realiza la generación de términos de izquierda a derecha a partir de otros términos calculados anteriormente (añadiendo palabras y realizando las omisiones oportunas), `Filter` realiza el filtrado progresivo seleccionado y `FilterFinal` realiza el filtrado final con los criterios que no se pueden comprobar progresivamente.

Datos: dataset
$X \longleftarrow$ `Tokenize`$(dataset)$;
$T_1 \longleftarrow$ `Generate`(X, k_{max});
$T_1 \longleftarrow$ `Filter`(T_1);
para $n \leftarrow 2$ **a** n_{max} **hacer**
  $T_n \longleftarrow$ `Generate`(X, k_{max}, T_{n-1});
  $T_n \longleftarrow$ `Filter`(T_n);
fin
$T \longleftarrow$ `FilterFinal`$(T_1, ..., T_{n_{max}})$;

Algoritmo 1: Algoritmo para la generación y filtrado progresivo de términos utilizando skipgrams.

En la Sección 4 realizaremos algunos ex-

perimentos para ver en qué medida se reduce la generación de términos utilizando esta técnica en diferentes conjuntos de datos.

3.2 Pesado por densidad

Las aproximaciones más comunes a la hora de pesar los términos dentro de un documento en una tarea de clasificación son la *binaria* (se pesa con 1 si el término aparece en el texto o 0 si no aparece) o el *conteo básico* (se pesa con el número de veces que aparece el término en el documento). A partir de ahí existen diferentes técnicas de normalización para que los valores se mantengan en el rango $[0, 1]$, como por ejemplo el *tf-idf*.

Estos pesados de términos están pensados para palabras o n-gramas, donde tenemos claro si realmente el término aparece o no en el texto. Sin embargo, en el modelado de skipgrams un término puede aparecer en un texto pero no de manera tan estricta. Por ejemplo, en el texto «La pantalla es muy brillante», podemos decir que el término *pantalla brillante* aparece si realizamos 2 omisiones o saltos. Sin embargo, en otro texto «Tiene una pantalla brillante» el término también aparece, pero sin realizar ninguna omisión. Con el fin de saber si esta distancia influye, proponemos una nueva forma de pesado que tiene en cuenta el número de omisiones, y que hemos denominado *pesado por densidad*. En la Ecuación 1 podemos ver la fórmula propuesta, donde el w_t es el peso del término t, y k es el número de omisiones realizadas para generarlo.

$$w_t = (1 + k)^{-1} \tag{1}$$

En el ejemplo anterior, el término «pantalla brillante» tendría un peso de $w_t = (1 + 2)^{-1} = 0{,}67$ para el primer texto y un peso de $w_t = (1 + 0)^{-1} = 1$ para el segundo texto, dando el peso máximo en el segundo porque no hay omisiones. En el caso de que haya varias ocurrencias de un mismo término en un documento, podemos tomar dos aproximaciones: la primera sería similar al pesado binario, indicando si el término aparece o no, pero en su lugar devolviendo el máximo peso por densidad en ese documento (ver Ecuación 2), y la otra similar al conteo de ocurrencias, devolviendo la suma de todos los pesos por densidad (ver Ecuación 3). En dichas ecuaciones, se calcula el peso $w_{t,d}$ de un término t en un documento d, donde $O_{t,d}$ es el conjunto de todas las ocurrencias del término t en el documento d y k_i es el número de omisiones utilizadas para generar la ocurrencia i.

$$w_{t,d} = max_{i \in O_{t,d}} (1 + k)^{-1} \tag{2}$$

$$w_{t,d} = \sum_{i \in O_{t,d}} (1 + k)^{-1} \tag{3}$$

En la Sección 4 realizaremos la experimentación para comprobar en qué situaciones esta información es valiosa, concretamente en el contexto del aprendizaje automático aplicado a la tarea de análisis de sentimientos.

4 Experimentación y resultados

Todos los experimentos realizados en esta sección se han realizado en conjuntos de datos de textos etiquetados para análisis de sentimientos, concretamente para la tarea clasificación de polaridad. Los conjuntos de datos elegidos están formados por textos cortos o frases, para no generar términos con palabras de frases diferentes y evitar el uso de herramientas de división de frases en textos. El preprocesamiento realizado en cada texto es básico: pasar los textos a minúsculas, eliminar acentos, eliminar repeticiones de caracteres (más de 3), y reemplazar menciones y hashtags por las cadenas `USER` y `HASHTAG` respectivamente (para conjuntos de datos de Twitter[2]). Para extraer las palabras también utilizaremos una aproximación simple, una expresión regular que divide por espacios y signos de puntuación, extrayendo solo palabras formadas por letras y/o números: `(?u)\b\w+\b`.

4.1 Reducción mediante filtrado progresivo

Para comprobar en qué medida el filtrado progresivo puede reducir el número de términos generados, realizaremos diferentes experimentaciones con diferentes conjuntos datos. La primera experimentación la hemos realizado en el dataset del TASS 2020 (Vega et al., 2020), concretamente el conjunto `train` en castellano (`es`) de la tarea `1.1`. Es un conjunto de datos que contiene tweets en castellano (textos cortos obtenidos de Twitter), formado por 1126 documentos y 5314 palabras diferentes, y con un tamaño medio por tweet de 14,82 palabras. En la Tabla 1 se pueden ver los diferentes números de términos

[2] `https://twitter.com`

para diferentes valores de n_{max} y k_{max}, donde SF es el número de términos sin filtrado, FP es el número de términos generados mediante filtrado progresivo, y FT es el número de términos tras el filtrado final. Por ejemplo, en el caso de $n = 3$ y $k = 4$ tenemos reducido los términos generados al 35,80 %, lo que significa que no hemos tenido que filtrar finalmente (FT) el 64,19 % de los términos ya que ni siquiera se han llegado a generar. Destacar que estos porcentajes no se refieren al número de términos filtrados finalmente sino al número de términos que se ha conseguido no tener que generar. En este trabajo nos hemos centrado en la generación, pero realmente el número de términos filtrados es mucho mayor, por ejemplo para el ejemplo anterior con $n = 3$ y $k = 4$ el número de términos final es de 11834 (una reducción del 94,75 %).

La segunda experimentación la hemos realizado en el dataset Movie Reviews (Pang y Lee, 2005), más específicamente el conjunto de frases con polaridad `sentence polarity dataset v1.0`, un conjunto de datos de críticas de películas en inglés, formado por 10695 frases y 18285 palabras diferentes. El tamaño medio por documento es de 18,06 palabras, más grande que el anterior, por lo que esperamos una cantidad mayor de términos generados. En la Tabla 2 se pueden ver los diferentes números de términos para diferentes valores de n y k. Podemos ver que a partir de esas 18285 palabras diferentes se llegan a generar casi 26 millones de términos en el caso de $n_{max} = 5$ y $k_{max} = 5$, reduciéndose a 359939 términos, de ahí la importancia del filtrado al utilizar skipgrams.

Comparando los datos de ambos conjuntos, podemos observar que en el segundo conjunto genera muchos más términos que el primero. Esto es algo de esperar ya que el número de palabras, el número de documentos y el tamaño medio por documento es mayor. Sin embargo, el porcentaje de reducción es menor en el segundo caso. Una posible explicación es que en el segundo conjunto los términos son más susceptibles de observarse más de una vez, algo también explicable por ser un conjunto mayor. En cualquier caso, observamos una reducción significativa de términos generados en valores altos de n_{max} y k_{max}, precisamente en los casos en los que consideramos que es más necesaria.

4.2 Pesado por densidad en análisis de sentimientos

En esta experimentación evaluaremos el comportamiento del modelado de skipgrams en la tarea de análisis de sentimientos. Para ello realizaremos experimentos utilizando *máquinas de soporte vectorial* (SVM) por sus buenos resultados para texto (Yadav et al., 2020). Utilizaremos la implementación de Scikit Learn (Pedregosa et al., 2011) `LinearSVC` con los parámetros por defecto.

Como características utilizaremos los términos extraídos de los conjuntos de datos anteriores (misma aproximación) con diferentes pesados: binario, *tf-idf* y nuestra propuesta de pesado por densidad (3.2, Ecuación 2). También utilizaremos una combinación de ambos pesados para para la experimentación (*tf-idf* y pesado por densidad), donde utilizaremos la fórmula del pesado *tf-idf* pero sustituyendo los número de ocurrencias por el pesado por densidad descrito en la Ecuación 3, dando como resultado la fórmula en la Ecuación 4, donde $w_{t,d}$ es el resultado del pesado por densidad.

$$tfidf_{t,d} = tf_{t,d} \cdot idf_t \rightarrow w_{t,d} \cdot idf_t \tag{4}$$

La evaluación la realizaremos mediante *validación cruzada estratificada* con 10 particiones, utilizando la métrica *F1* con promediado *macro* (misma importancia a todas las polaridades). Nuestro punto de partida o *baseline* serán las configuraciones sin pesado por densidad, y nuestro objetivo será mejorar los resultado de estas configuraciones.

La primera experimentación la realizaremos en el conjunto de datos del TASS 2020 (mencionado en la sección anterior) para diferentes valores de n_{max} y k_{max}. En la Tabla 3 podemos ver los resultados de esta primera experimentación, donde B indica que se ha realizado un pesado binario, D que se ha utilizado el pesado por densidad (fórmula en la Ecuación 2), T que se ha realizado un pesado por *tf-idf*, $D+T$ que se ha realizado un pesado por densidad (fórmula en la Ecuación 3) seguido de un pesado por *tf-idf*, y $DF = 1$, $DF = 2$ y $DF = 3$ que se ha utilizado un criterio de filtrado de términos con ocurrencia mayor o igual a 1, 2 y 3 respectivamente (cabe destacar que en los experimentos donde $DF = 1$ no se realiza ningún filtrado ya que todos los términos aparecen al menos una vez).

$k_{max} \rightarrow$	0	1	2	3	4	5
$n_{max} = 2$, *SF*	16301	27384	37101	45628	53104	59684
$n_{max} = 2$, *FP*	11021	17250	22511	26960	30809	34082
$n_{max} = 2$, *FT*	2661	3868	4971	5951	6840	7647
$n_{max} = 3$, *SF*	31021	69367	116106	168812	225472	284122
$n_{max} = 3$, *FP*	14837	28709	44750	62310	80741	99411
$n_{max} = 3$, *FT*	3094	4925	7052	9366	11834	14556
$n_{max} = 4$, *SF*	45244	122602	240101	399100	598443	834869
$n_{max} = 4$, *FP*	15623	31511	51769	76401	105138	138139
$n_{max} = 4$, *FT*	3164	5134	7456	10055	12999	16377
$n_{max} = 5$, *SF*	58426	183611	409139	762196	1264757	1931543
$n_{max} = 5$, *FP*	15734	31943	52873	78727	109558	145836
$n_{max} = 5$, *FT*	3184	5190	7560	10223	13217	16691

Tabla 1: Número de términos sin filtrar (*SF*), número de términos generados utilizando el filtrado progresivo (*FP*) y número de términos incluyendo el filtrado final (*FT*). El criterio de filtrado es que los términos deben aparecer en más de un documento ($df \geq 2$) en el conjunto de datos `train` de la tarea `1.1` del TASS 2020 en castellano.

$k_{max} \rightarrow$	0	1	2	3	4	5
$n_{max} = 2$, *SF*	124320	229385	323444	407768	483434	551530
$n_{max} = 2$, *FP*	109423	200351	281110	352897	416937	474173
$n_{max} = 2$, *FT*	30583	48491	64134	78428	91451	103555
$n_{max} = 3$, *SF*	284760	701386	1231744	1850792	2536975	3271609
$n_{max} = 3$, *FP*	186643	432725	733293	1075730	1448698	1843316
$n_{max} = 3$, *FT*	41149	75377	113432	156141	202733	252977
$n_{max} = 4$, *SF*	453546	1351736	2783988	4798134	7413739	10625918
$n_{max} = 4$, *FP*	213460	533488	975839	1545202	2242864	3068329
$n_{max} = 4$, *FT*	44397	84651	132123	189098	256428	334776
$n_{max} = 5$, *SF*	616087	2125177	4981590	9634849	16506362	25963045
$n_{max} = 5$, *FP*	220216	560629	1046063	1693926	2522137	3547672
$n_{max} = 5$, *FT*	45223	87235	137480	198747	272537	359939

Tabla 2: Número de términos sin filtrar (*SF*), número de términos generados utilizando el filtrado progresivo (*FP*) y número de términos incluyendo el filtrado final (*FT*). El criterio de filtrado es que los términos deben aparecer en más de un documento ($df \geq 2$) en el conjunto de datos `sentence polarity dataset v1.0` de Movie Reviews en inglés.

Observando los resultados obtenidos podemos ver que utilizar el pesado por densidad mejora el rendimiento de los skipgrams con cualquier configuración ($D > B$, $D+T > T$), menos en el caso de n-gramas, lo que es lógico ya que los n-gramas siempre tienen densidad 1, y únicamente en el caso de skipgrams con $n_{max} = 2, k_{max} = 1, df = 1$. La mejora media utilizando el pesado por densidad es de un 3,35 %, llegando al 7,67 % en el mejor caso. Por lo tanto podemos decir que el uso de la información sobre la distancia en los skipgrams es una información valiosa a la hora de pesar los skipgrams ya que ayuda a mejorar los resultados en esta tarea de análisis de sentimientos, en algunos casos de manera significativa. También podemos observar que sin el pesado por densidad (B y T), la utilización de skipgrams nunca mejora los resultados respecto a los n-gramas. Añadir la información sobre el número de omisiones no es solo recomendable sino también necesaria si queremos

$k_{max} \rightarrow$	0	1	2	3	4	5
$n_{max} = 2$, *DF=1, B*	0,501	0,504	0,495	0,496	0,504	0,497
$n_{max} = 2$, *DF=1, D*	0,501	0,509	0,506	0,506	0,509	0,509
$n_{max} = 2$, *DF=1, T*	0,499	0,503	0,489	0,489	0,481	0,483
$n_{max} = 2$, *DF=1, D+T*	0,499	0,499	0,504	0,502	0,503	0,501
$n_{max} = 2$, *DF=2, B*	0,506	0,497	0,494	0,483	0,478	0,478
$n_{max} = 2$, *DF=2, D*	0,506	0,506	0,508	0,513	0,504	0,506
$n_{max} = 2$, *DF=2, T*	0,515	0,503	0,493	0,482	0,493	0,489
$n_{max} = 2$, *DF=2, D+T*	0,515	**0.517**	0,509	0,505	0,510	0,509
$n_{max} = 2$, *DF=3, B*	0,491	0,488	0,478	0,475	0,481	0,475
$n_{max} = 2$, *DF=3, D*	0,491	0,493	0,496	0,489	0,489	0,491
$n_{max} = 2$, *DF=3, T*	0,496	0,502	0,481	0,484	0,492	0,485
$n_{max} = 2$, *DF=3, D+T*	0,496	0,504	0,498	0,499	0,494	0,491
$n_{max} = 3$, *DF=1, B*	0,509	0,493	0,488	0,477	0,470	0,474
$n_{max} = 3$, *DF=1, D*	0,509	0,505	0,503	0,504	0,506	**0.510**
$n_{max} = 3$, *DF=1, T*	0,502	0,488	0,467	0,463	0,457	0,458
$n_{max} = 3$, *DF=1, D+T*	0,502	0,497	0,495	0,488	0,486	0,489
$n_{max} = 3$, *DF=2, B*	0,495	0,488	0,494	0,487	0,471	0,469
$n_{max} = 3$, *DF=2, D*	0,495	0,500	0,503	0,499	0,497	0,503
$n_{max} = 3$, *DF=2, T*	0,507	0,494	0,478	0,472	0,472	0,474
$n_{max} = 3$, *DF=2, D+T*	0,507	0,505	0,501	0,493	0,501	0,497
$n_{max} = 3$, *DF=3, B*	0,486	0,482	0,487	0,481	0,469	0,466
$n_{max} = 3$, *DF=3, D*	0,486	0,492	0,488	0,487	0,485	0,491
$n_{max} = 3$, *DF=3, T*	0,499	0,491	0,479	0,477	0,476	0,468
$n_{max} = 3$, *DF=3, D+T*	0,499	0,493	0,489	0,491	0,488	0,486

Tabla 3: Evaluación realizada con la unión de los conjuntos de datos `dev` y `train` de la tarea `1.1` del TASS 2020 en castellano (`es`) para diferentes valores de n_{max} y k_{max}, utilizando la medida $F1$ con promediado *macro*, donde B indica que se ha realizado un pesado binario, D que se ha utilizado el pesado por densidad, T que se ha realizado un pesado por *tf-idf*, $D + T$ que se ha realizado un pesado por densidad seguido de un pesado por *tf-idf*, y $DF = 1$, $DF = 2$ y $DF = 3$ que se ha utilizado un criterio de filtrado de términos con ocurrencia mayor o igual a 1, 2 y 3 respectivamente.

utilizar skipgrams de manera efectiva. Destacar que el pesado por *tf-idf* por sí solo ha obtenido buenos resultados, pero que se han visto mejorados al añadir el pesado por densidad, que parece ser la mejor combinación para pesar todo tipo de términos.

Respecto al filtrado, en este conjunto de datos no podemos deducir si, además de mejorar la eficiencia reduciendo los recursos espaciales y temporales, se mejora también la efectividad de los modelos en análisis de sentimientos. Pero a pesar de ser reducciones tan agresivas (que pueden llegar al orden del 90 % como hemos visto en la Sección 3.1), los resultados son similares y en algunos casos pueden conseguirse mejoras significativas.

Con el fin de poner en contexto esta aproximación con el estado de la cuestión, comparamos la mejor de nuestras configuraciones (*DF=2, D+T*) con los resultados oficiales de la competición del TASS 2020, que se sitúan en el rango $[0{,}37, 0{,}67]$. Sin utilizar conocimiento externo, utilizando como entrenamiento los conjuntos `dev` y `train` provistos por la competición, y evaluando con el conjunto de `test`, para castellano (`es`), obtenemos una puntuación en el rango $[0{,}505, 0{,}522]$ (según diferentes valores de n_{max} y k_{max}), que nos situaría en una posición intermedia, superando a algunas aproximaciones que utilizan técnicas como *word embeddings* o *neural networks*. Si aumentamos el entrenamiento utilizando los conjuntos `dev` y `train` de todos los idiomas de la competición (variantes del español), para poder aumentar nuestro vocabulario, y evaluamos igualmente solo con el conjunto de `test` en castellano (`es`), la puntuación aumentaría hasta el ran-

go [0,531, 0,554], lo que nos dejaría cerca de algunas de las aproximaciones que utilizan *deep learning* o *transformers*. Es cierto que nuestros resultados se han obtenido una vez terminada la competición, pero se ha intentado simular el contexto para poder comprobar en qué situación quedaría nuestra propuesta respecto a técnicas más recientes.

La siguiente experimentación la realizaremos en el conjunto de datos de Movie Reviews (también mencionado en la sección anterior) igualmente para diferentes valores de n_{max} y k_{max}. En la Tabla 4 se pueden observar los resultados con la misma nomenclatura que en los experimentos anteriores.

Los resultados de esta experimentación nos ofrecen una visión similar sobre el pesado por densidad. Como en el conjunto de datos anterior, nuestra propuesta de pesado mejora el rendimiento de los skipgrams con cualquier configuración ($D > B$, $D + T > T$) menos en el caso de n-gramas. La mejora media utilizando el pesado por densidad es de un 1,65 %, llegando al 5,65 % en el mejor caso. A pesar de que el conjunto de datos tiene más documentos, más palabras diferentes, un tamaño medio mayor, y está en un idioma diferente, la información sobre la distancia en los skipgrams sigue siendo valiosa a la hora de pesar los skipgrams en análisis de sentimientos. Igualmente, sin el pesado por densidad los skipgrams no ofrecen mejoras respecto a n-gramas.

5 Conclusiones y trabajo futuro

En este trabajo hemos realizado dos propuestas para mejorar el uso del modelado de skipgrams. Por un lado, una técnica para la generación y filtrado progresivos, con el objetivo de reducir el número de términos que se generan a la hora de extraer términos automáticamente de un conjunto de datos. Hemos aplicado esta técnica en dos corpus diferentes y hemos visto una reducción significativa de términos generados, que depende del conjunto de datos utilizado, pero que es considerablemente mayor cuanto más términos y omisiones se realizan, precisamente los casos en los que es más necesaria por la gran cantidad que se genera. Esta reducción puede llegar a un 95 %, pero viendo la tendencia creemos que puede ser mucho mayor si realizamos la experimentación con mayores valores de n_{max} y k_{max}. En la experimentación posterior hemos observado que para tareas como el análisis de sentimientos, este filtrado no reduce el rendimiento sino que en algunos casos los puede incluso mejorar.

Por otro lado, hemos propuesto un esquema de pesado de términos que tiene en cuenta el número de omisiones realizadas al generar skipgrams. Se ha observado que el uso de skipgrams en análisis de sentimientos necesita de este tipo de pesado, ya que en caso contrario los resultados empeoran al utilizar skipgrams. Pero cuando se usa el pesado por densidad, los resultados pueden mejorar hasta un 7 % en los mejores casos. Además, mencionar que el esquema propuesto combinado con el clásico tf-idf es lo que mejores resultados ofrece en el contexto estudiado.

Este trabajo abre la puerta a nuevas investigaciones relacionadas, entre las que podemos destacar:

- Buscar y diseñar nuevos tipos de filtrado progresivo eficientes que sean capaces de eliminar los términos menos relevantes pero manteniendo la efectividad de los sistemas que los utilizan, empezando por técnicas de *selección de características* y *reducción de la dimensionalidad*.
- Estudiar esquemas de pesado existentes o diseñar nuevos, y probar diferentes combinaciones entre ellos para ver si pueden mejorar los resultados.
- Aplicar el modelado de skipgrams en diferentes conjuntos de datos, géneros textuales, idiomas e incluso tareas de PLN.
- Analizar las curvas de aprendizaje de diferentes modelos utilizando skipgrams, para comprobar en qué medida pueden ampliar la cobertura de los sistemas.
- Utilizar los términos generados para crear un lexicón o diccionario de sentimientos que pueda ser utilizado como recurso para otras herramientas.

Agradecimientos

Esta investigación ha sido financiada por la Universidad de Alicante, el Ministerio de Ciencia e Innovación de España, la Generalitat Valenciana y el Fondo Europeo de Desarrollo Regional (FEDER) a través de la siguiente financiación: a nivel nacional, se concedieron los proyectos *TRIVIAL* (PID2021-122263OB-C22), *SocialTrust* (PDC2022-133146-C22) y *CLEARTEXT* (TED2021-130707B-I00), financiados

$k_{max} \rightarrow$	0	1	2	3	4	5
$n_{max} = 2$, *DF=1, B*	0,774	0,774	0,769	0,762	0,759	0,757
$n_{max} = 2$, *DF=1, D*	0,774	0,774	0,775	0,774	0,773	0,777
$n_{max} = 2$, *DF=1, T*	0,784	0,783	0,778	0,774	0,773	0,771
$n_{max} = 2$, *DF=1, D+T*	0,784	**0.789**	0,787	0,787	0,787	0,787
$n_{max} = 2$, *DF=2, B*	0,760	0,762	0,758	0,756	0,754	0,751
$n_{max} = 2$, *DF=2, D*	0,760	0,765	0,766	0,767	0,765	0,764
$n_{max} = 2$, *DF=2, T*	0,777	0,778	0,776	0,772	0,772	0,770
$n_{max} = 2$, *DF=2, D+T*	0,777	0,781	0,784	0,782	0,785	0,786
$n_{max} = 2$, *DF=3, B*	0,750	0,751	0,751	0,748	0,745	0,744
$n_{max} = 2$, *DF=3, D*	0,750	0,755	0,757	0,758	0,756	0,757
$n_{max} = 2$, *DF=3, T*	0,771	0,771	0,771	0,768	0,77	0,768
$n_{max} = 2$, *DF=3, D+T*	0,771	0,775	0,775	0,778	0,78	0,781
$n_{max} = 3$, *DF=1, B*	0,773	0,764	0,747	0,737	0,730	0,726
$n_{max} = 3$, *DF=1, D*	0,773	0,771	0,772	0,769	0,768	0,767
$n_{max} = 3$, *DF=1, T*	0,777	0,768	0,761	0,753	0,751	0,745
$n_{max} = 3$, *DF=1, D+T*	0,777	0,777	0,777	0,775	0,776	0,776
$n_{max} = 3$, *DF=2, B*	0,761	0,759	0,753	0,754	0,748	0,750
$n_{max} = 3$, *DF=2, D*	0,761	0,764	0,764	0,765	0,765	0,766
$n_{max} = 3$, *DF=2, T*	0,780	0,776	0,776	0,772	0,772	0,768
$n_{max} = 3$, *DF=2, D+T*	0,780	0,783	**0.785**	0,783	0,784	**0.785**
$n_{max} = 3$, *DF=3, B*	0,751	0,747	0,748	0,745	0,743	0,744
$n_{max} = 3$, *DF=3, D*	0,751	0,753	0,755	0,757	0,757	0,756
$n_{max} = 3$, *DF=3, T*	0,770	0,771	0,767	0,765	0,768	0,766
$n_{max} = 3$, *DF=3, D+T*	0,770	0,774	0,774	0,776	0,780	0,779

Tabla 4: Evaluación realizada en el conjunto de datos `sentence polarity dataset v1.0` de Movie Reviews en inglés para diferentes valores de n_{max} y k_{max}, utilizando la medida $F1$ con promediado *macro*, donde B indica que se ha realizado un pesado binario, D que se ha utilizado el pesado por densidad, T que se ha realizado un pesado por *tf-idf*, $D + T$ que se ha realizado un pesado por densidad seguido de un pesado por *tf-idf*, y $DF = 1$, $DF = 2$ y $DF = 3$ que se ha utilizado un criterio de filtrado de términos con ocurrencia mayor o igual a 1, 2 y 3 respectivamente.

por MCIN/AEI/10.13039/501100011033 y European Union NextGenerationEU/PRTR; a nivel regional, la Generalitat Valenciana (Conselleria d'Educació, Investigació, Cultura i Esport), concedió financiación para NL4DISMIS (CIPROM/2021/21). Además, contó con el apoyo de dos acciones COST: CA19134 - "Distributed Knowledge Graphs" y CA19142 - "Leading Platform for European Citizens, Industries, Academia, and Policymakers in Media Accessibility".

Bibliografía

Chang, C.-Y., S.-J. Lee, y C.-C. Lai. 2017. Weighted word2vec based on the distance of words. En *2017 International Conference on Machine Learning and Cybernetics (ICMLC)*, volumen 2, páginas 563–568. IEEE.

Church, K. W. 2017. Word2vec. *Natural Language Engineering*, 23(1):155–162.

Du, X., J. Yan, R. Zhang, y H. Zha. 2020. Cross-network skip-gram embedding for joint network alignment and link prediction. *IEEE Transactions on Knowledge and Data Engineering*.

Gompel, M. v. y A. van den Bosch. 2016. Efficient n-gram, skipgram and flexgram modelling with colibri core. *Journal of Open Research Software*, 4:1–10.

Guthrie, D., B. Allison, W. Liu, L. Guthrie, y Y. Wilks. 2006. A closer look at skip-gram modelling. En *Proceedings of the fifth international conference on language resources and evaluation (LREC'06)*.

Hossny, A. H., L. Mitchell, N. Lothian, y

G. Osborne. 2020. Feature selection methods for event detection in twitter: A text mining approach. *Social Network Analysis and Mining*, 10(1):1–15.

Komninos, A. y S. Manandhar. 2016. Dependency based embeddings for sentence classification tasks. En *Proceedings of the 2016 conference of the North American chapter of the association for computational linguistics: human language technologies*, páginas 1490–1500.

Mikolov, T., K. Chen, G. Corrado, y J. Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Mimno, D. y L. Thompson. 2017. The strange geometry of skip-gram with negative sampling. En *Empirical Methods in Natural Language Processing*.

Nguyen, T. H. y R. Grishman. 2016. Modeling skip-grams for event detection with convolutional neural networks. En *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, páginas 886–891.

Pang, B. y L. Lee. 2005. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. En *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, páginas 115–124.

Pedregosa, F., G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, y E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Peng, H., J. Li, H. Yan, Q. Gong, S. Wang, L. Liu, L. Wang, y X. Ren. 2020. Dynamic network embedding via incremental skip-gram with negative sampling. *Science China Information Sciences*, 63(10):1–19.

Santos, F. A. O., T. D. Bispo, H. T. Macedo, y C. Zanchettin. 2021. Morphological skip-gram: Replacing fasttext characters n-gram with morphological knowledge. *Inteligencia Artificial*, 24(67):1–17.

Shazeer, N., J. Pelemans, y C. Chelba. 2015. Sparse non-negative matrix language modeling for skip-grams. *Proceedings Interspeech 2015*, 2015:1428–1432.

Vaswani, A., N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, y I. Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Vega, M. G., M. C. Díaz-Galiano, M. Á. G. Cumbreras, F. M. P. del Arco, A. Montejo-Ráez, S. M. J. Zafra, E. M. Cámara, C. A. Aguilar, M. A. S. Cabezudo, L. Chiruzzo, y others. 2020. Overview of tass 2020: Introducing emotion detection. En *IberLEF@ SEPLN*.

Yadav, B. P., S. Ghate, A. Harshavardhan, G. Jhansi, K. S. Kumar, y E. Sudarshan. 2020. Text categorization performance examination using machine learning algorithms. En *IOP Conference Series: Materials Science and Engineering*, volumen 981, página 022044. IOP Publishing.

Zhao, Z., T. Liu, S. Li, B. Li, y X. Du. 2017. Ngram2vec: Learning improved word representations from ngram co-occurrence statistics. En *Proceedings of the 2017 conference on empirical methods in natural language processing*, páginas 244–253.