Cómo interpretar y analizar automáticamente la información textual

PID_00257761

Joaquim Moré

Tiempo mínimo de dedicación recomendado: 4 horas

Universitat Oberta de Catalunya

Joaquim Moré

El encargo y la creación de este recurso de aprendizaje UOC han sido coordinados por el profesor: Jordi Casas (2019)

Primera edición: septiembre 2019
Autoría: Joaquim Moré

Av. Tibidabo, 39-43, 08035 Barcelona
Realización editorial: FUOC

Índice

Introducción

En este módulo explicaremos los fundamentos del procesamiento del lenguaje natural (PLN) y las técnicas básicas para procesar la información textual de manera que podamos abordar el análisis de opiniones y sentimientos.

Explicaremos los fundamentos del PLN basándonos en el caso de uso que presentamos a continuación:

El caso

El caso de uso está inspirado en los datos de un *kernel* que tiene por nombre *March 2018 NYT Headline Click Bait,* de la competición Kaggle: *New York Times Comments*. Disponible en el enlace: https://www.kaggle.com/aashita/nyt-comments

Los directivos de la versión digital del *New York Times* deciden poner publicidad de sus patrocinadores junto a las noticias que provocan más comentarios.

Por eso piensan en invertir en la predicción de titulares motivadores de comentarios. De este modo, sabrán qué noticias pueden ir acompañadas de publicidad.

Siguiendo las indicaciones de unos consultores, los directivos deciden contratar a alguien que se ocupe de descubrir qué contenidos de los titulares provocan más comentarios.

Los consultores se ofrecen a hacer este servicio por una cantidad elevada; argumentan que tendrían que subcontratar a un ejército de especialistas en comunicación y marketing para analizar a fondo los titulares del *New York Times* de manera manual.

Afortunadamente, los directivos del *New York Times* conocen una pequeña empresa llamada S&S (*Sense and Suitability*), especializada en procesamiento del lenguaje natural. En el pasado, S&S había automatizado algunos procesos de búsqueda de información y análisis de las noticias por parte de los redactores y, por consiguiente, los directivos piensan que el análisis de los titulares también se puede automatizar aplicando técnicas de PLN, lo cual supondría un ahorro importante de dinero.

Los directivos se ponen en contacto con S&S, que se muestra interesada en el proyecto. La empresa plantea una propuesta: en un mes tendrán preparada una prueba de concepto para demostrar que es viable hacer, con métodos de PLN, un análisis de titulares para clasificar los que son motivadores de opinión.

El objetivo de este módulo es dar respuesta a las preguntas metodológicas básicas. Son las preguntas que el equipo de S&S tiene que plantearse y a las que tiene que dar respuesta, asumiendo que los contenidos de un texto, sea este un titular o cualquier otro documento, son en realidad sus palabras. Las preguntas que el equipo de S&S se plantea son las siguientes:

1) ¿Qué consideramos palabra?
2) ¿Cómo se detectan y procesan las palabras?
3) ¿Cómo se pueden obtener los datos que relacionan las palabras de un texto con las opiniones que provoca?
4) ¿Cómo se puede aprovechar esta relación para predecir la reacción de los lectores?
5) ¿Qué herramientas y recursos hay disponibles?

1. ¿Qué se considera palabra?

Desde el punto de vista del procesamiento del lenguaje natural, la respuesta a esta pregunta puede parecer obvia. Dado un texto, la palabra es una secuencia de caracteres entre dos espacios en blanco: entre el inicio de línea y un espacio en blanco o bien entre un espacio en blanco y el final de línea. Y si queremos obtener las palabras del texto, basta romper la secuencia de caracteres (*split*, si utilizamos un término de uso común en los lenguajes de programación) allá donde hay un espacio en blanco. De este modo, podemos obtener las palabras de los titulares del *New York Times*. En la tabla 1, podemos ver la lista de palabras de un titular obtenida según este criterio. En el *notebook* PLA-1 1.1 se muestra cómo se hace en Python.

Tabla 1. Palabras de un titular según el método de *split*

Titular: The "Daily Show" host said the attacks of the president on fellow Republicans at a meeting reminded him of "a drunk uncle calling everyone out at a wedding".
['The', '"Daily', 'Show"', 'host', 'said', 'the', 'attacks', 'of', 'the', 'president', 'on', 'fellow', 'Republicans', 'at', 'a', 'meeting', 'reminded', 'him', 'of', '"a', 'drunk', 'uncle', 'calling', 'everyone', 'out', 'at', 'a', 'wedding."']

El análisis de los resultados obtenidos por el método de *split* hace que afloren las distintas maneras que tienen los miembros de S&S de abordar el encargo.

Por un lado, Peter, que es el coordinador del equipo de desarrollo, sabe por experiencia que algoritmos muy simples pueden solucionar problemas complejos. Él tiene en mente probar un análisis estrictamente cuantitativo que consiste en verificar si la longitud de los titulares está relacionada con el número de comentarios. Joseph, un ingeniero informático especialista en *machine learning*, está de acuerdo en explorar este método. Para Peter y Joseph, la lista obtenida por el método de *split* les parece adecuada.

En cambio, Beth, Anne y Henry son unos miembros del equipo que creen que debe abordarse el objetivo con criterios cualitativos, concretamente con criterios semánticos. Piensan que el contenido semántico del titular, lo que dice, influye en los lectores y les motiva a escribir comentarios. Con esta idea en mente, Beth, Anne y Henry no están de acuerdo en considerar todos los elementos de la lista de la tabla 1 como palabras. Veamos algunas de sus razones.

1.1. La palabra como unidad de significante y significado

«Wedding. "», por ejemplo, no es una palabra, aunque sí lo es *wedding*. *Wedding*, sin las comillas y el punto, tiene un contenido semántico, mientras que «wedding. "» no lo tiene. Por la misma razón, en vez de *"Daily* se debería considerar *Daily*, en vez de *Show"* se debería considerar *Show*, y en vez de *"a* se debería considerar *a*. Para Beth, que estudió filología y tiene muy interiorizada la noción de palabra que le enseñaron en la facultad, las comillas y los signos de puntuación son elementos tipográficos que se adhieren a la palabra en los textos escritos, pero que no tienen presencia en otros canales de comunicación. La palabra es un signo compuesto de significante y significado en el medio escrito y también en el oral. El significante de una palabra es la forma, que corresponde a su pronunciación, y el significado es el concepto al que la palabra hace referencia. Por lo tanto, las palabras de la lista tendrían que estar despojadas de elementos tipográficos que rompen la relación entre el significante y su significado.

Bibliografía

Sobre la palabra como símbolo compuesto de significante y significado, véase *Cours de linguistique générale*, de **Ferdinand de Saussure**, **Charles Bally** y **Albert Sechehaye** (1916).

Por lo tanto, Beth, Anne y Henry quieren obtener una lista donde los elementos con signos de puntuación no se distingan del elemento sin signo de puntuación. De esta manera, también será posible contar la frecuencia de palabras que tienen el mismo significado. A continuación se compara la frecuencia de las palabras según el método de *split* y la frecuencia de palabras según la relación del significante con el significado (véase *notebook* PLA-1 1.2).

Tabla 2. Frecuencias de las palabras del titular según el método de *split* y de relación significante y significado

Titular: The "Daily Show" host said the attacks of the president on fellow Republicans at a meeting reminded him of "a drunk uncle calling everyone out at a wedding".	
Método *split*	'at': 2, 'a': 2, 'The': 1, '"Daily': 1, 'Show"': 1, 'host': 1, 'said': 1, 'the': 1, 'president's': 1, 'attacks': 1, 'on': 1, 'fellow': 1, 'Republicans': 1, 'meeting': 1, 'reminded': 1, 'him': 1, 'of': 1, '"a': 1, 'drunk': 1, 'uncle': 1, 'calling': 1, 'everyone': 1, 'out': 1, 'wedding."': 1
Relación significante con significado	'a ': 3, 'at ': 2, 'The ': 1, 'Daily': 1, 'Show ': 1, 'host ': 1, 'said ': 1, 'the ': 2, 'attacks': 1, 'president ': 1, 'on': 1, 'fellow': 1, 'Republicans': 1, 'meeting': 1, 'reminded': 1, 'him': 1, 'of': 2, 'drunk': 1, 'uncle': 1, 'calling': 1, 'everyone': 1, 'out': 1, 'wedding': 1

Es evidente que el hecho de que *the* esté en mayúscula o minúscula no afecta su sentido. Por lo tanto, *The* y *the* se pueden contar como una sola palabra. Si no distinguimos entre mayúsculas y minúsculas la frecuencia de las palabras (todas en minúsculas) queda así:

Tabla 3. Frecuencias de las palabras del titular sin distinguir mayúsculas y minúsculas

Titular: The "Daily Show" host said the attacks of the president on fellow Republicans at a meeting reminded him of "a drunk uncle calling everyone out at a wedding".

Relación significante con significado	'a': 3, 'the': 3, 'at': 2, 'daily': 1, 'show': 1, 'host ': 1,' said ': 1,' president ': 1,' attacks ': 1,' on ': 1,' fellow ': 1,' republicans ': 1,' meeting ': 1, 'reminded': 1, 'him': 1, 'of': 2, 'drunk': 1, 'uncle': 1, 'calling': 1, 'everyone': 1, 'out': 1, 'wedding': 1

1.2. Términos *monopalabra* y *multipalabra*

Ante la lista de la tabla 3, Anne se da cuenta de que se ha roto la relación estrecha que existía entre las palabras *daily* y *show*. Estas son ahora dos palabras independientes, con su significante y significado, cuando el redactor había utilizado el recurso de poner unas comillas («*Daily Show*») para indicarle al lector que las dos palabras se referían a una sola cosa. Henry incluso cree que el redactor había considerado la combinación *Daily Show host* como la referencia a un único concepto.

Anne, Beth y Henry plantean la posibilidad de considerar también combinaciones de dos o más palabras como una sola unidad de significado, al igual que *New York Times*. *New York Times* es una unidad formada por tres palabras que hace referencia a un periódico y no a *New*, ni a la ciudad de *York* ni a *Times* por separado.

Los tres consideran que estas combinaciones de palabras forman una palabra en sí misma. Así pues, una palabra es también la combinación de dos o más palabras. Eso sí, el significado de esta palabra trasciende o es diferente de la unión de los significados de las palabras que la forman. Esta consideración de palabra es distinta de la consideración de palabra como una sola unidad que tienen Peter y Joseph, que es la más popular.

Para evitar malentendidos, Beth, Anne y Henry deciden adoptar la noción de término. Un término puede ser monopalabra o multipalabra. En el campo de la lingüística computacional, los términos multipalabra se denominan también *colocaciones* o *multiwords*.

2. ¿Preprocesar o no preprocesar?

Como ya hemos comentado, Peter —que es el coordinador— quiere explorar la relación entre la longitud de los titulares y el número de comentarios, mientras que Beth defiende una aproximación basada en la relación entre el significante y el significado. En la segunda reunión de equipo, Peter decide que Joseph explore la relación entre la longitud de los titulares y el número de comentarios y decide también que, mientras Joseph explora la vía cuantitativa, Beth, Anne y Henry trabajen en la vía más semántica.

El análisis de los titulares por parte de Peter no requiere de un preprocesado del texto, sin embargo, el análisis que tienen que realizar Beth, Anne y Henry, sí —como explicaremos en este apartado.

Peter decide reunirse tres días después para contrastar los resultados obtenidos y decidir cuál es la mejor aproximación.

2.1. El corpus de análisis

Peter pide al *New York Times* una muestra de titulares para poder preparar la propuesta. Al día siguiente, el *New York Times* entrega a S&S una hoja de cálculo que recoge todos los titulares del mes de marzo de 2018, con los respectivos cuerpos de noticia y el número de comentarios.

2.2. Preprocesado del texto y sus herramientas

Como ya hemos comentado, el método de obtención de palabras vía *split* ya le va bien a Joseph, aunque desde el punto de vista gramatical y filológico sea discutible. Sin embargo, para Beth, Anne y Henry, la detección de las palabras requiere de un procesado previo del texto. Este procesado se realiza con las herramientas que explicaremos a continuación a medida que Beth, Anne y Henry tengan que solventar las dificultades normales en esta tarea.

2.2.1. Tokenizer

El **tokenizer** es la herramienta básica del procesamiento del lenguaje natural. *Tokenizer* es un término inglés que es difícil de traducir al español con una sola palabra. Es por ello que nos referiremos a esta herramienta con el término original, que es de uso común entre especialistas en PLN.

Dado un texto, el tokenizer lista los *tokens* de ese texto. *Token* es otro término inglés de uso común entre especialistas en PLN. El *token* es la unidad textual mínima procesada que no siempre corresponde a un término, como se puede ver en la tabla 4 (véase *notebook* PLA-1 2.1)

Tabla 4. *Tokens* de un titular

Titular: The "Daily Show" host said the attacks of the president on fellow Republicans at a meeting reminded him of "a drunk uncle calling everyone out at a wedding".	
Tokens	['The', '``', 'Daily', 'Show', "''", 'host', 'said', 'the', 'attacks', 'of', 'the', 'president', 'on', 'fellow', 'Republicans', 'at', 'a', 'meeting', 'reminded', 'him', 'of', '``', 'a', 'drunk', 'uncle', 'calling', 'everyone', 'out', 'at', 'a', 'wedding', '.', "''"]

Como se ve, el tokenizer ha distinguido como *tokens* símbolos que no son términos: puntos, comillas, etc. Por lo tanto, estos *tokens*, que comparten el hecho de que no tienen un carácter alfanumérico, deberían ser filtrados. Lenguajes de programación como Java, Perl o Python, incluso los comandos del sistema operativo Unix, tienen los llamados paquetes *regex* (*regular expressions* o 'expresiones regulares'), con los cuales se pueden distinguir los *tokens* que no contienen caracteres alfanuméricos. En el *notebook* PLA-1, 2.2, se puede ver cómo se filtran estos *tokens* en Python.

2.2.2. Etiquetador de categoría gramatical

Beth, que también es traductora y tiene un conocimiento lingüístico más teórico, siempre dice que la teoría aporta una visión más abstracta del problema, no deja que uno se empantane en las excepciones y en los detalles y ayuda a encontrar una solución más general y efectiva.

Ella sugiere un criterio para filtrar los *tokens* que no son palabras, gracias al cual no hay que entretenerse encontrando soluciones *ad hoc* para cada excepción. El criterio es el siguiente: **los tokens que no tengan una categoría gramatical clásica serán filtrados**.

Cuando Anne y Henry le preguntan qué quiere decir con *categoría gramatical-clásica,* ella responde: «Pues el **nombre**, el **verbo**, el **adjetivo**, el **adverbio**, el **pronombre**, el **determinante**, la **preposición** y la **conjunción**».

Para poner en práctica esta idea, utilizan lo que se conoce como un *etiquetador de categoría gramatical*. Este etiquetador asigna a cada *token* una etiqueta que describe su categoría gramatical.

Los etiquetadores de categoría gramatical también se conocen como *PoS taggers,* y el proceso de asignación de una etiqueta a cada *token* se conoce por su término en inglés: *PoS tagging*. PoS es el acrónimo de *part of speech*, que es equivalente a 'categoría gramatical' en inglés.

Las etiquetas siguen unos estándares. El estándar más utilizado es el conocido como *Penn Treebank*, de la Universidad de Pensilvania. En el notebook PLA-1, 2.3 se muestra cómo se hace el *PoS tagging* de los *tokens* de un titular, y también explicamos las etiquetas estándar que se han asignado a estos *tokens*. El resultado es el que aparece en la tabla 5.

Tabla 5. *Tokens* de un titular etiquetados con un PoS Tagger

Titular: The "Daily Show" host said the attacks of the president on fellow Republicans at a meeting reminded him of "a drunk uncle calling everyone out at a wedding".	
PoS tagging	[('the', 'DT'), ('``', '``'), ('daily', 'JJ'), ('show', 'NN'), ("'", "'"), ('host', 'NN'), ('said', 'VBD'), ('the', 'DT'), ('attacks', 'NNS'), ('of', 'IN'), ('the', 'DT'), ('president', 'NN'), ('on', 'IN'), ('fellow', 'JJ'), ('republicans', 'NNS'), ('at', 'IN'), ('a', 'DT'), ('meeting', 'NN'), ('reminded', 'VBD'), ('him', 'PRP'), ('of', 'IN'), ('``', '``'), ('a', 'DT'), ('drunk', 'JJ'), ('uncle', 'NN'), ('calling', 'VBG'), ('everyone', 'NN'), ('out', 'RB'), ('at', 'IN'), ('a', 'DT'), ('wedding', 'NN'), ('.', '.'), ("'", "')]

Siguiendo el criterio de Beth, las palabras serían los *tokens* que tienen etiquetas que empiezan por *N* (nombre), por *DT* (determinante) y por *V* (verbo). También las preposiciones (con etiqueta «IN»), pronombres (con etiqueta «PRP») y adverbios (con etiqueta «RB»).

Existe la etiqueta de nombre propio (NPP) y hay etiquetadores que son sensibles al hecho de que el *token* tenga una mayúscula inicial y etiquetan un determinante, o cualquier *token* que no es un nombre, como un nombre propio. Para evitar confusiones, Beth, Anne y Henry deciden poner los titulares en minúscula.

2.2.3. Buscador de n-gramas

Anne recuerda a sus compañeros que quedan por detectar automáticamente los términos multipalabra. Anne sabe que para detectar los términos multipalabra primero se deben encontrar los n-gramas del texto y, de estos n-gramas, escoger los que son más susceptibles de ser términos multipalabra. Pero ¿qué es un n-grama?

Un **n-grama** es una secuencia de *tokens* consecutivos que tiene un orden de complejidad *n*. El orden de complejidad corresponde al número de *tokens* consecutivos del n-grama. Los n-gramas de orden 1 se conocen como **unigramas**; los de orden 2, se conocen como **bigramas**; y los de orden 3 se conocen como **trigramas**.

Anne propone asumir que los términos multipalabra más largos en inglés son trigramas. Por eso utilizan un buscador de los bigramas y trigramas que hay en los titulares del *New York Times*. En el *notebook* PLA-1. 2.4 se enseña cómo se buscan los bigramas y trigramas de un texto. Los bigramas y trigramas encontrados se muestran en la tabla 6.

Tabla 6. Bigramas y trigramas de un titular

Titular: The "Daily Show" host said the attacks of the president on fellow Republicans at a meeting reminded him of "a drunk uncle calling everyone out at a wedding".	
bigramas	[('the', '``'), ('``', 'daily'), ('daily', 'show'), ('show', """), ("""", 'host'), ('host', 'said'), ('said', 'the'), ('the', 'attacks'), ('attacks', 'of'), ('of', 'the'), ('the', 'president'), ('president', 'on'), ('on', 'fellow'), ('fellow', 'republicans'), ('republicans', 'at'), ('at', 'a'), ('a', 'meeting'), ('meeting', 'reminded'), ('reminded', 'him'), ('him', 'of'), ('of', '``'), ('``', 'a'), ('a', 'drunk'), ('drunk', 'uncle'), ('uncle', 'calling'), ('calling', 'everyone'), ('everyone', 'out'), ('out', 'at'), ('at', 'a'), ('a', 'wedding'), ('wedding', '.'), ('.', """)]
trigramas	[('the', '``', 'daily'), ('``', 'daily', 'show'), ('daily', 'show', """), ('show', """, 'host'), ("""", 'host', 'said'), ('host', 'said', 'the'), ('said', 'the', 'attacks'), ('the', 'attacks', 'of'), ('attacks', 'of', 'the'), ('of', 'the', 'president'), ('the', 'president', 'on'), ('president', 'on', 'fellow'), ('on', 'fellow', 'republicans'), ('fellow', 'republicans', 'at'), ('republicans', 'at', 'a'), ('at', 'a', 'meeting'), ('a', 'meeting', 'reminded'), ('meeting', 'reminded', 'him'), ('reminded', 'him', 'of'), ('him', 'of', '``'), ('of', '``', 'a'), ('``', 'a', 'drunk'), ('a', 'drunk', 'uncle'), ('drunk', 'uncle', 'calling'), ('uncle', 'calling', 'everyone'), ('calling', 'everyone', 'out'), ('everyone', 'out', 'at'), ('out', 'at', 'a'), ('at', 'a', 'wedding'), ('a', 'wedding', '.'), ('wedding', '.', """)]

2.2.4. Filtradores de n-gramas

Henry dice que cuando Peter, el coordinador, vea la tabla de n-gramas le dará un ataque. Teme que cuando vea que para un solo titular es necesario procesar tantos n-gramas con palabras combinadas con signos de puntuación, y con combinaciones tan poco informativas como *'out at a'*, *'republicans at'*, etc., dirá que la vía que han querido tomar es inaplicable.

Anne piensa que las combinaciones con signos de puntuación se podrían evitar si estos signos se sacaran del titular antes de hacer la búsqueda de n-gramas. Beth dice que esto no es correcto porque precisamente los signos de puntuación separan las palabras y las frases. La eliminación de los signos de puntuación significaría tomar como n-gramas combinaciones de palabras que originalmente estaban en frases distintas.

1) Filtrado con diccionarios y otras bases de datos léxicas

Beth plantea un filtrado de n-gramas que consiste en tomar aquellos n-gramas de orden superior a 1 que aparecen como entradas en diccionarios electrónicos, como Wiktionary, fuentes enciclopédicas como Wikipedia u ontologías como Wordnet. Hablaremos más extensamente sobre Wordnet en otros apartados, pero de momento, diremos que Wordnet es una base de datos léxica de la lengua inglesa, desarrollada por la universidad de Princeton y organizada como una ontología. Más adelante, explicaremos las características de una ontología y su potencial.

Adaptaciones de Wordnet

Actualmente, hay bases de datos léxicas que son adaptaciones del Wordnet a otras lenguas. Hay Wordnets para el español, catalán, gallego, euskera, entre otras lenguas europeas, financiados bajo el paraguas del proyecto europeo EuroWordnet (LE-2 4003), en el que participaron la UPC, la UB, la Universidad del País Vasco, la UNED y la Universidad de Vigo.

Anne plantea algunas objeciones al planteamiento de Beth. La principal objeción es que en los titulares aparecen términos multipalabra que no están recogidos en ningún recurso léxico que hemos mencionado, sea porque son de un dominio muy específico o porque son expresiones que tienen un uso intenso durante un periodo corto de tiempo y luego se dejan de usar.

Nota

Si los titulares estuvieran escritos en una lengua minoritaria, la objeción principal sería la falta de recursos léxicos en esta lengua para ser consultados por una herramienta de procesamiento del lenguaje natural.

2) Filtrado con criterios tipográficos

Las comillas marcan el inicio y el final de un término multipalabra, como, por ejemplo, “Daily Show”. Anne plantea buscar cuatrigramas (n-gramas de orden 4) que tengan las comillas como *token* inicial y final (p. ej. ('”', 'daily', 'show', '”')). La colocación estaría formada por los dos *tokens* que ocupan las posiciones intermedias (‘daily show’).

Este filtrado tiene sus limitaciones, puesto que no todos los autores de los titulares marcan explícitamente las colocaciones con unas comillas.

3) Filtrado con métodos estadísticos

Henry sugiere encontrar los n-gramas que corresponden a términos multipalabra a partir de métodos estadísticos identificando los n-gramas, cuyos componentes están tan íntimamente relacionados que son considerados como colocaciones. Según él, un n-grama como *'host said that'* no es un término multipalabra que hay que considerar, porque esta combinación es producto de la coocurrencia de estas palabras en un titular determinado. Es diferente el caso, por ejemplo, de *tea party*, donde las dos palabras coocurren en cualquier texto que hable de este grupo de influencia favorable al partido republicano.

Henry conoce cálculos estadísticos como el *t-test*, el *chi-square*, el *Pointwise Mutual Information* (PMI) o la *log-likehood* para encontrar colocaciones. Estas métricas capturan la discrepancia entre las probabilidades de que las palabras coocurran en cualquier texto y las probabilidades de que estas aparezcan por separado asumiendo que la ocurrencia de una palabra no afecta la probabilidad de ocurrencia de otra palabra (son independientes).

Métricas *t-test, chi-square, PMI* y *log-likehood*

Estas métricas se explican en el capítulo 5 del libro de Manning y Schütze (2000), *Foundations of Natural Language Processing*. También en los apartados 3.1 y 6.7 del libro de Jurafsky y Martin (2018), *Speech and Language Processing*.

Colocación

Según Natural Language Processing with Python: *A collocation is a sequence of words that occur together unusually often.*

Por tanto, una colocación se puede considerar como una secuencia de palabras que muy frecuentemente aparecen juntas, lo cual es inusual. En el *notebook* PLA-1 2.5 se puede ver cómo se buscan los bigramas y trigramas, que son colocaciones utilizando la métrica PMI. El resultado se muestra en la tabla 7.

Tabla 7. Filtración de los n-gramas de un un titular calculando la métrica PMI

Texto: The "Daily Show" host said the attacks of the president on fellow Republicans at a meeting reminded him of "a drunk uncle calling everyone out at a wedding".

	N-gramas que superan el filtrado
bigramas	'calling_everyone', 'daily_show', 'drunk_uncle', 'everyone_out', 'fellow_republicans', 'host_said', 'meeting_reminded', 'on_fellow', 'president_on', 'reminded_him'
trigramas	'calling_everyone_out', 'drunk_uncle_calling', 'meeting_reminded_him', 'on_fellow_republicans', 'president_on_fellow', 'uncle_calling_everyone', 'everyone_out_at', 'fellow_republicans_at', 'reminded_him_of', 'a_drunk_uncle'

El hecho de tener un texto tan corto es la causa de que el cálculo estadístico del PMI no sea lo suficientemente significativo como para evitar falsos positivos (p. ej. *everyone_out*). Por otro lado, hay combinaciones como *president_on* o *meeting reminded* que, de entrada, no se deberían considerar términos multipalabra.

Ahora bien, cuando el texto es grande, las coocurrencias de *everyone_out* y también de *meeting reminded* pierden excepcionalidad, mientras que otras coocurrencias, como podrían ser *fellow republicans* o *daily show*, la mantienen y son merecedoras de ser consideradas como colocaciones.

4) Filtrado con *stop words*

N-gramas como *president on* o *a drunk uncle* no son términos, porque, o bien el *token* inicial o bien el *token* final son *stop words*. ¿Qué es un *stopword*?

Un ***stop word*** es una palabra que tiene como función cohesionar un texto, pero no aporta nada al significado del texto ni, evidentemente, al significado del n-grama. Son *stopwords*: los artículos, las preposiciones, las conjunciones, etc.

Por lo tanto, para buscar términos multipalabra deberían filtrarse los n-gramas que tienen un *stopword* al inicio o al final. En el *notebook* PLA-1 2.6 se puede ver cómo se filtran los n-gramas con la lista de *stopwords* del NLTK, que es una librería en Python especializada en procesamiento del lenguaje natural. El resultado se muestra en la tabla 8.

Tabla 8. Filtrado de n-gramas con *stopwords* al principio o al final

Texto: The "Daily Show" host said the attacks of the president on fellow Republicans at a meeting reminded him of "a drunk uncle calling everyone out at a wedding."	
	N-gramas que superan el filtrado
bigramas	'calling_everyone', 'daily_show', 'drunk_uncle', 'fellow_republicans', 'host_said', 'meeting_reminded'
trigramas	'drunk_uncle_calling', 'president_on_fellow', 'uncle_calling_everyone'

En la red es fácil encontrar listas de *stop words*, sobre todo del inglés. Beth, sin embargo, cree que su utilidad depende del criterio de las personas que las han elaborado. Por ejemplo, piensa que *everyone* debería estar en esta lista, y también recuerda haber visto, en la lista, palabras que no le interesaba filtrar.

5) Filtrado según el PoS de los constituyentes del n-grama

Beth cree que hay una alternativa al filtraje con *stop words* que es más eficaz y que consiste en aplicar un criterio gramatical que es el siguiente: se filtran los n-gramas cuya palabra inicial o final es un determinante, pronombre, verbo modal, preposición, adverbio o conjunción. También considera que los verbos en gerundio[1] y flexionados[2] si están en pasado no pueden aparecer ni al principio ni al final. En el *notebook* PLA-1 2.7 se enseña cómo se obtienen las colocaciones según este criterio. El resultado se muestra en la siguiente tabla.

(1)Con etiqueta VBG

(2)Por ejemplo, con etiqueta VBD

Tabla 9. Filtrado de n-gramas de un texto según el PoS de sus constituyentes

Texto: The "Daily Show" host said the attacks of the president on fellow Republicans at a meeting reminded him of "a drunk uncle calling everyone out at a wedding".	
	N-gramas que superan el filtrado
Bigramas	'daily_show', 'drunk_uncle', 'fellow_republicans',
Trigramas	'president_on_fellow', 'uncle_calling_everyone'

Vemos que el criterio de Beth no ha evitado que aparezca *everyone*. La razón es la discrepancia entre el criterio de Beth y el del *PoS tagger*. Beth clasificaría *everyone* como un pronombre, mientras que el *PoS tagger* lo clasifica como un nombre. En el siguiente apartado volveremos sobre la cuestión de las discrepancias en el etiquetaje.

2.2.5. *Parsers* y buscadores de patrones sintácticos

Beth propone también aplicar una asunción basada en su experiencia como estudiosa de la gramática de las lenguas. La asunción es que los términos en inglés multipalabra relevantes para el proyecto serán combinaciones Nombre + Nombre. Por esta razón, propone hacer un análisis sintáctico del titular con un analizador (conocido también como *parser*), para identificar las combinaciones de términos que tienen el patrón Nombre + Nombre. En el *notebook* PLA-1 2.8 podemos ver cómo se identifican términos que tienen el patrón sintáctico Nombre y Nombre + Nombre. Estos términos son los siguientes:

Tabla 10. Términos multipalabra del texto que tienen el patrón Nombre y Nombre + Nombre

Texto: The "Daily Show" host said the attacks of president Trump on fellow Republicans at a meeting reminded him of "a drunk uncle calling everyone out at a wedding".
['show', 'host', 'attacks', 'president', 'trump', 'republicans', 'meeting', 'uncle', 'everyone', 'wedding', 'president_trump']

El *parser* hace el etiquetaje de categoría gramatical con el *PoS tagger* que clasifica *everyone* como un nombre. Ahora bien, hay *parsers* cuyo *PoS tagger* toma un criterio distinto y lo etiqueta como pronombre, con lo cual *everyone* no saldría en la lista de términos.

2.2.6. Software disponible

Hay una variedad de software de código abierto con el que se pueden listar los *tokens*, hacer el etiquetado de categorías gramaticales, buscar los n-gramas y las colocaciones y hacer un análisis sintáctico de un texto. Todos son igualmente recomendables. El analista puede utilizar el software en el lenguaje de programación con el que se sienta más cómodo o le sea más fácil de integrar en el entorno de análisis. A continuación presentamos una tabla con algunos de los más conocidos y utilizados.

Tabla 11. Librerías de procesamiento del lenguaje natural de código abierto

Librería	Desarrollador	Lenguaje
NLTK	Team NLTK	Python
Gensim	RaRe Technologies	Python
Spacy	Matthew Honnibal	Python y Cython
Pattern	CLiPs (University of Antwerp)	Python
OpenNLP	Apache Software Foundation	Java
GATE	Inicialmente desarrollado en la Universidad de Sheffield	Java
CoreNLP	Universidad de Stanford	Java
Tokenizers	Lincoln Mullen, Os Keyes, Dmitriy Selivanov, Jeffrey Arnold, Kenneth Benoit	R

Librería	Desarrollador	Lenguaje
MALLET	Andrew McCallum	Java

2.3. Resultados

Henry, Beth, Anne y Joseph se reúnen con Peter, el coordinador, para mostrar y discutir los resultados que han obtenido.

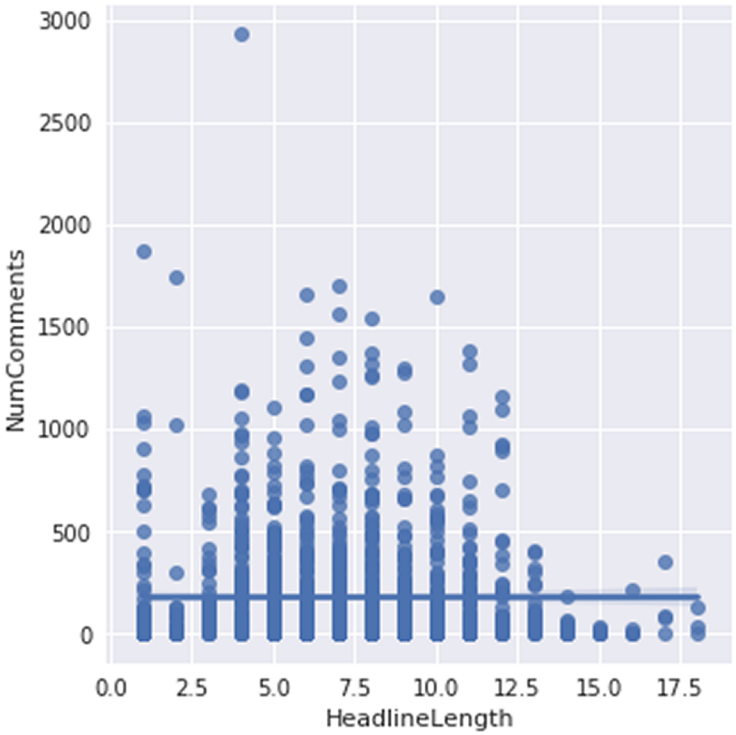
Joseph tenía que ver si existía una relación entre la longitud de los titulares y el número de comentarios. Su análisis no requería un preprocesado de los titulares, pues se limitaba a cortarlos (*split*) allí donde había un espacio en blanco y calculaba la longitud de la lista de elementos separados.

Los resultados obtenidos por Joseph muestran que la longitud de los titulares no es un dato relevante. El grado de correlación entre la longitud y el número de comentarios es muy bajo (figura 1) y la línea de regresión es prácticamente plana (figura 2).

Figura 1. Encabezamiento del *dataframe* que relaciona longitud de los titulares (*HeadlineLength*) con el número de comentarios (*NumComments*) y grado de correlación.

```
                  articleID  HeadlineLength  NumComments
0  5a974697410cf7000162e8a4               4           37
1  5a974be7410cf7000162e8af               7           22
2  5a9752a2410cf7000162e8ba               7          307
3  5a975310410cf7000162e8bd               9           44
4  5a975406410cf7000162e8c3              11          365
                HeadlineLength  NumComments
HeadlineLength        1.000000    -0.000054
NumComments          -0.000054     1.000000
```

Figura 2. Representación gráfica de la correlación entre la longitud de los titulares (*HeadlineLength*) y el número de comentarios (*NumComments*)



Por su parte, Anne, Beth y Henry explican en la reunión que han trabajado en el preprocesado de los titulares y en la elaboración de algoritmos para detectar los términos mono y multipalabra. Les queda por hacer la exploración de los términos y detectar los que son más relevantes para relacionar los titulares con el número de comentarios.

Vistos los resultados obtenidos con el no procesado de los titulares, Peter dice a todos los miembros del grupo que se involucren en la exploración de términos y quedan para la semana siguiente para discutir los resultados obtenidos.

3. Detección de términos relevantes

En este apartado presentaremos los aspectos que hay que tener en cuenta a la hora de detectar y procesar las palabras más relevantes para el análisis. En nuestro caso de uso, las palabras más relevantes son las palabras de los titulares que supuestamente influyen en el lector y le motivan a escribir comentarios. Como en el apartado anterior, expondremos estos aspectos y también los recursos disponibles explicando cómo los miembros de S&S se van enfrentando a los problemas más comunes.

3.1. Ley de Zipf

Anne, Beth, Joseph y Henry son conscientes de que tienen que enfrentarse a la llamada **Ley de Zipf**. Una ley que contradice el mito de que los datos relevantes están entre los datos más frecuentes. Este mito puede ser cierto para muchos tipos de datos, pero cuando los datos son lingüísticos, no funciona.

El filólogo y lingüista George Kingsley Zipf presentó, a mediados de los años cuarenta del siglo pasado, la ley que lleva su nombre y que puede resumirse de la siguiente manera.

Ley de Zipf

Si listamos las palabras según su orden de frecuencia, esto es, por su rango, la segunda palabra de la lista aparecerá aproximadamente la mitad de veces que la primera palabra, la tercera palabra aparecerá con una frecuencia aproximada de un tercio de veces y así sucesivamente.

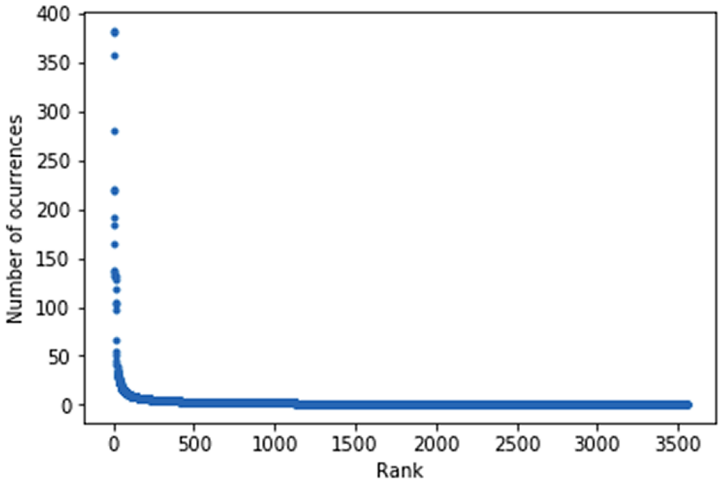
Según esta ley, las palabras que ocupan las primeras posiciones concentran el mayor porcentaje de las palabras de un texto. Lo relevante es que entre estas palabras están los artículos, las preposiciones o las conjunciones. Palabras que, a pesar de repetirse mucho en el texto, no aportan casi nada de contenido.

Zipf formuló su ley para demostrar la tendencia que tienen los humanos a hacer las cosas con el mínimo esfuerzo. Si el receptor es capaz de entender el mensaje cuando el emisor utiliza las palabras más importantes N veces, ¿por qué repetirlas más veces? Las palabras que se repiten mucho son aquellas cuya función es la de articular el mensaje de manera reconocible y comprensible para el receptor. Por ejemplo, que el receptor sepa en qué lengua le están hablando; esta es la función principal de los artículos, preposiciones, etc.

La ley se cumple ya sea con un libro, un artículo, una noticia, etc. Y los titulares del *NewYorkTimes* no son una excepción. La siguiente gráfica[3] muestra la relación entre la frecuencia y el rango de los términos monopalabra de los titulares del *New York Times*. El rango es la posición que un término ocupa en la lista de términos ordenados por frecuencia.

(3)En el *notebook* PLA-1. 3.1 está el *script* que genera la gráfica

Figura 3. Relación entre la frecuencia y el rango de los términos monopalabra en los titulares del *New York Times*



Las palabras que tienen más información siempre estarán en la parte más baja de la curva. Incluso pueden ser *hapax legomenon*, que significa 'dicho una sola vez', en griego, ya que, ¿por qué tengo que repetir una y otra vez una palabra que, dicha una sola vez, ya queda claro cuál es el mensaje que quiero transmitir?

3.2. Filtraje de *stop words*

Las palabras que concentran el mayor porcentaje de frecuencia de aparición, pero que no aportan nada de contenido, son las *stopwords* (c.f. 2.2.4.4). Por lo tanto, estas palabras deberían filtrarse antes de proceder a la exploración de los datos interesantes.

3.3. Normalización del texto

Una tarea importante del proceso de exploración de los datos es la **normalización** del texto. Los elementos de un texto son términos como *libro* y *libros*, que son variantes de una forma que aglutina las referencias al concepto de *libro*. Es importante, por tanto, que los datos que hay que explorar sean las formas aglutinadoras de las variantes.

3.3.1. Lemas y *stems*

La forma que aglutina las variantes suele ser el lema.

El **lema** es la forma de las entradas de un diccionario. Cuando se trata de un nombre, en lenguas como el inglés, el castellano, el catalán, etc., el lema coincide con la forma singular. Cuando se trata de un verbo, el lema coincide con el infinitivo.

Coloquialmente, cuando alguien dice la palabra *P* significa *x*, está identificando la palabra con su lema.

A partir de ahora, y para facilitar la comprensión, cuando digamos «la palabra *P* aparece *N* veces», lo hacemos en su sentido coloquial. Así, cuando digamos que la palabra *kill* aparece 200 veces, es porque hemos sumado las veces que aparecen las variantes del lema *kill* (*killed, kill, kills, killing*, etc.)

Otra forma normalizada es la **raíz** de un término (*stem*, en inglés).

La **raíz** de un sustantivo y también la raíz de un verbo es lo que queda después de haber sacado sus morfemas variables. El proceso de sacar los morfemas variantes es conocido como *stemming*.

Hay distintos algoritmos de *stemming*, por lo que los resultados pueden variar según el algoritmo elegido.

Mostramos a continuación unos ejemplos de lematización y *stemming* (véase *notebook* PLA-1 3.2). El algoritmo de *stemming* utilizado es el *Porter stemmer*.

Tabla 12. Ejemplos de lematización y *stemming*

Formas	Lema	Stem
feet	*foot*	*feet*
elephants	*elephant*	*eleph*
communities	*community*	*commun*

3.3.2. Sinonimia

Anne cree que la normalización de los datos puede ir más allá agrupando términos por su significado. El criterio más evidente de agrupación de términos según su significado es la relación de sinonimia. Es decir, Anne piensa en procesar expresiones variantes del mismo significado y pone como ejemplo contar formas variantes del concepto de matar. De esta manera, se contarían términos monopalabra y multipalabra como *kill* y *put_to_death* como variantes de este concepto, al igual que se contarían *President Trump*, *President Donald Trump* o *Donald Trump* como referencias al mismo personaje.

Anne sabe muy bien la importancia que tiene la sinonimia, pues recuerda su análisis con métodos de PLN de las ofertas laborales en inteligencia de negocio. Anne no podía imaginarse que la sinonimia le supondría un verdadero quebradero de cabeza. Unas ofertas de trabajo ponían *business intelligence*, en otras *BI* y en otras *Business Analytics*; también había quien ponía DW en vez de Data Warehouse, etc. Se le pusieron los pelos de punta cuando Peter le dijo que se estaba hablando de hacer el mismo análisis para una empresa española con ofertas publicadas en España y también con ofertas publicadas en países de Sudamérica, lo cual suponía unificar la referencia a los mismos conceptos en todas las variantes del español. El proyecto finalmente no se llevó a cabo.

3.3.3. Hiperonimia

Beth, siempre dispuesta a aplicar sus conocimientos de teoría lingüística, propone explorar otro criterio de normalización basado en el significado. Este criterio es la relación de hiperonimia.

En una ontología, la **relación de hiperonimia** es la que se establece entre las instancias de una clase y la clase. Esta relación es una relación de tipo *A* es un *B*.

Por ejemplo, las palabras *pistola*, *rifle*, *escopeta* comparten el mismo hiperónimo, ***arma***, con lo cual, se podrían agrupar estas palabras a un nivel más abstracto que el de la sinonimia. Las palabras que comparten un hiperónimo *H* se conocen como los **hipónimos** de *H*.

Beth cree que los datos relevantes para el proyecto no son tanto los términos como los hiperónimos. Considera que los hiperónimos son como los temas de los que habla un titular, y que habría que orientar el proyecto en poder hacer afirmaciones como esta: «Cuando se habla de armas o deportes hay más comentarios».

A continuación presentamos un ejemplo de normalización de términos que son *N* y *N+N*. Los términos *attacks* y *fire*[4] se normalizan en un solo término por compartir el mismo hiperónimo.

Tabla 13. Normalización de términos por hiperonimia

Texto: The "Daily Show" host said the attacks and fire of president Trump on fellow Republicans at a meeting reminded him of "a drunk uncle calling everyone out at a wedding".	
	Términos
Sin normalización por hiperonimia	['show', 'host', 'attacks', 'fire', 'president', 'trump', 'republicans', 'meeting', 'uncle', 'everyone', 'wedding', 'president_trump']

(4) *Fire* tienen el sentido de crítica en este ejemplo, un sentido que no es el que nos viene primero a la mente cuando la palabra está sin contexto.

Ved también

Sobre el sentido de las palabras, véase el apartado 3.3.4.

Normalización por hiperonimia (hiperónimo común: *criticism*)	['show', 'host', '**criticism**', 'president', 'trump', 'republicans', 'meeting', 'uncle', 'everyone', 'wedding', 'president_trump']

3.3.4. El problema de la ambigüedad

La normalización automática del texto puede producir resultados no deseados, debido en gran parte a la ambigüedad en la forma de las palabras.

Dos palabras pueden tener la misma forma, pero la categoría gramatical y el significado ser diferentes.

> Por ejemplo, en la frase *The inhabitants house the inmigrants in their own house*, la forma *house* tiene dos PoS diferentes. El primer *house* es un verbo y el segundo *house* es un sustantivo. El verbo *house* significa 'alojar' y el sustantivo significa 'casa'. Por lo tanto, el lema *house* no debería aglutinar estas dos formas.

Por otro lado, dos formas idénticas con el mismo PoS pueden tener significados diferentes.

> Por ejemplo, tenemos *run,* que es la forma de dos verbos distintos. El verbo *run* puede significar 'correr' y también puede significar 'ejecutar', como en *this software can run several tasks at the same time*.

La distinción de los PoS y de los sentidos de las palabras que tienen la misma forma se conoce como **desambiguación**, y en la jerga de la lingüística computacional, *Word Sense Disambiguation*, o WSD.

La normalización depende, por tanto, de buenas herramientas de desambiguación, una de las tareas más difíciles en el procesamiento del lenguaje natural.

3.3.5. Recursos

La lematización se puede hacer con el paquete NLTK, que busca en la base de datos léxica Wordnet el lema de un término (véase PLA-1. 3.2). También se puede hacer con la **librería Wordnet** en R. Si el término tiene la misma forma en más de una categoría gramatical, hay que especificarle al lematizador la categoría gramatical que el *PoS tagger* le ha asignado al término.

Wordnet, como base de datos léxica organizada como una ontología, es además una herramienta básica para desarrollar la normalización de los términos según las relaciones de sinonimia e hiperonimia.

En cuanto al *stemming*, hay que destacar el stemmer **Snowball**. Snowball hace *stemming* de palabras en inglés, catalán, español, francés, italiano, además de otras lenguas románicas y germánicas, y también hace stemming del árabe. Está integrado en la librería NLTK, aunque también está en Java, integrado en el buscador Lucene.

La desambiguación o WSD es una tarea que deben realizar los *parsers*, pues una correcta representación sintáctica depende de lo bien que desambigüen el PoS y el sentido de las palabras.

Actualmente, uno de los *parsers* de código abierto más utilizado es **FreeLIng**. A su vez, la mayoría de herramientas presentadas en 2.2.6 tienen un *parser*.

FreeLing

FreeLing, desarrollado en la UPC por Lluís Padró, además del inglés y español, analiza textos en catalán, euskera y gallego, entre otras lenguas. Véase **http://nlp.lsi.upc.edu/freeling** y **X. Carreras; I. Chao; L. Padró** y **M. Padró** (2004). *FreeLing: An Open-Source Suite of Language Analyzers*. Proceedings of the 4th International Conference on Language Resources and Evaluation (LREC'04).

Con la librerías en Python NLTK y SpaCy se analizan textos en inglés, aunque SpaCy puede aprender a analizar en otra lengua si dispone de un modelo de esta lengua. Pattern también analiza textos en español y GATE tiene su *parser* para el inglés en lenguaje Java.

3.3.6. Resultados

Llega el día en que tienen que mostrar a Peter los resultados obtenidos. Solamente han tenido tiempo de normalizar el texto vía lematización. También han unificado términos como *Mr. Trump* o *Donald Trump* con el término aglutinador *Trump*. Han preferido la lematización al *stemming* porque los lemas son más fácilmente reconocibles a la hora de verificar el proceso de normalización del texto que las raíces de las palabras, que pueden ser a menudo confusas. La normalización por medio de relaciones semánticas como la sinonimia, la hiperonimia, etc., propuesta por Beth, se presenta como una posibilidad que hay que considerar.

Bibliografía

Sobre las relaciones de sinonimia, hiperonimia y otras, véase el capítulo 6 del libro de **Jurafsky** y **Martin** (2018) *Speech and Language Processing*.

Anne, Beth y Henry presentan un *tag cloud* (figura 4) de la frecuencia de los términos mono y multipalabra lematizados —en el *notebook* PLA-1 3.3 se muestra cómo se hace. A Peter le llama la atención que *Trump* y *president* estén entre los términos más frecuentes. También considera que hay muchos términos que por sí solos no aportan nada (p. ej. *say, go, new*) porque pueden aparecer en todos los titulares, motivadores de comentarios o no, y hay otros términos que posiblemente son subalternos a términos que son más relevantes y tendrían que estar más destacados. Por ejemplo, le queda la duda de si *president* es un término vinculado a *Trump,* o bien es un término que indica que los presidentes, sean cuales sean, provocan comentarios en los lectores.

Por lo tanto, Peter les dice que trabajen para encontrar las palabras realmente relevantes y les cita para la semana siguiente.

Figura 4. *Tag cloud* de los términos lematizados en los titulares del *New York Times*

3.4. Vectorización del texto

Cuando se reúne el equipo, Henry apunta que el encargo de Peter supone ir más allá de contar términos. Los términos tienen que ponerse en el contexto de dos tipos de titulares. El primer tipo de titular es el que provoca muchos comentarios, y el segundo tipo de titular es el que provoca pocos comentarios. Por lo tanto, hay que encontrar los términos que son relevantes en los titulares del primer tipo, y esto significa que hay que representarlos de una forma computacionalmente tratable. La representación de los textos como vectores es la más utilizada.

3.4.1. Vectorizador

El vectorizador, más conocido por su término en inglés, *vectorizer*, es la herramienta que transforma cada texto de una colección de documentos en un vector. La vectorización se hace con una matriz de términos y documentos (*term-document matrix*) y con el conjunto de términos distintos que aparecen en todos los documentos de la colección (vocabulario).

Cada columna de la matriz representa un documento de la colección, y cada fila es la representación de un término del vocabulario. Cuando el vectorizador se basa en la frecuencia de los términos (*count vectorizer*), cada celda de esta fila representa las veces que un término del vocabulario aparece en el documento de la columna.

La tabla 14 muestra el *term-document-matrix* de cuatro obras de Shakespeare suponiendo que el vocabulario de estas obras es de cuatro palabras[5].

[5] El ejemplo es la figura 6.2 del libro de Jurafsky y Martin (2018).

Tabla 14. Ejemplo de *term-document-matrix*

	As you like it	Twelfth Night	Julius Caesar	Henry V
battle	1	1	8	15
soldier	2	2	12	36
fool	37	58	1	5

	As you like it	Twelfth Night	Julius Caesar	Henry V
clown	5	117	0	0

La vectorización de cada obra se hace recogiendo los valores numéricos de su columna correspondiente. O sea, *Henry V* se representaría con el vector [15, 36, 5, 0] y *Julius Caesar* con el vector [8, 12, 1, 0]

El *term-document-matrix* permite describir el documento como un vector, pero además permite describir un término del vocabulario también como un vector. El vector [5, 117, 0, 0], que representa *clown*, describe la contribución de este término en las cuatro obras.

Veamos ahora cómo se vectorizan cuatro titulares del *New York Times* con un *countvectorizer*. Los titulares son los siguientes:

T1: Trump proclaims tariffs on steel, and a tariff on aluminium and stocks sag in reply

T2: Trump embraces a trade war, which could undermine growth

T3: Mr. Trump and destructive trade war

T4: The macroeconomics of trade war

El *term-document matrix* es el que aparece a continuación. Los términos considerados son de la forma *N* y *N+N*, y han sido lematizados.

Tabla 15. *Term-document-matrix* de cuatro titulares del *New York Times*

	T1	T2	T3	T4
Trump	1	1	1	0
tariff	2	0	0	0
steel	1	0	0	0
stock	1	0	0	0
reply	1	0	0	0
trade	0	1	1	1
war	0	1	1	1
growth	0	1	0	0
trade_war	0	1	1	1
aluminium	1	0	0	0
macroeconomics	0	0	0	1

Los vectores que representan los titulares son los siguientes:

T1: [1, 2, 1, 1, 1, 0, 0, 1, 0, 1, 0]

T2: [1, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0]

T3: [1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0]

T4: [0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1]

3.4.2. Vectorizador tf.idf

A partir de este momento, los titulares de las noticias con más comentarios se llamarán titulares TOP. Los titulares restantes serán titulares NOTOP. Los titulares de tipo TOP son los titulares de las noticias con un número de comentarios que superan la media.

Para Henry, el vectorizador no debería tener en cuenta solo la presencia de los términos del vocabulario en los titulares TOP; también debería tener en cuenta la presencia de los términos del vocabulario en los titulares NOTOP. Pone como ejemplo a un doctorando que hace la tesis comparando los rasgos estilísticos de *Moby Dick* de Herman Melville con los de *Sense and Sensibility* de Jane Austen. Es evidente que tiene que buscar términos que aparecen *en Moby Dick* como *white whale* o *wildest watery spaces*, pero también tiene que comprobar que estos son términos característicos de esta novela, porque no aparecen en *Sense and Sensibility*. Peter ya apuntaba una cuestión parecida cuando decía que *president* y *new* no parecían ser términos específicos de los titulares TOP, porque pueden aparecer también en los titulares NOTOP.

Henry propone vectorizar los titulares con el llamado *vectorizador tf.idf*. El vectorizador tf.idf se aplicaría a partir de un *term-document-matrix* con dos columnas: una para los titulares TOP y otra para los titulares NOTOP. Las filas describirían el vocabulario de términos de todos los titulares y cada celda contendría un valor numérico indicativo de la contribución de un término a los titulares TOP y NOTOP. La contribución no es la frecuencia del término en un tipo de titular, sino el valor de una métrica llamada tf.idf.

La idea detrás de la métrica tf.idf es la siguiente: dado un documento *d* en una colección de documentos, un término *t* es característico del documento *d* si *t* es frecuente en *d*, pero, en cambio, no se encuentra o aparece muy poco en el resto de documentos de la colección. Volviendo al análisis de *Moby Dick*, el término *white whale* es representativo de la novela de Melville, porque es muy frecuente en *MobyDick*, pero no lo es en *Sense and Sensibility*.

La métrica es la combinación de otras dos métricas. El tf, o *Term frequency*, que es la frecuencia total de *t* en *d*; y el idf, *Inverse document frequency*, que es la inversa del número de documentos de la colección en los que aparece.

Bibliografía

Sobre el peso y la relevancia de los términos según su tf.idf, véase el apartado 15.2.2 del libro de Manning y Schütze (2000) y también los apartados 6.5 y 6.6 del libro de Jurafsky y Martin (2018). En la entrada *tf.idf* de la Wikipedia hay una explicación extensa de la fórmula y un ejemplo concreto. (https://en.wikipedia.org/wiki/Tf–idf)

La fórmula es la siguiente:

$$\text{tf.idf}(t, d, D) = \text{tf}(t, d) * \text{idf}(t, D)$$

t: término; *d*: documento; *D*: colección de documentos

Para calcular el TF, a menudo se normaliza la frecuencia de un término en el documento con el número total de palabras del documento.

El IDF es el logaritmo de la división entre el número de documentos de la colección y el número de documentos de la colección que tienen el término. El resultado es un número del 0 al 1. 0 significa que el término no es nada relevante en el documento, mientras que el 1 significa que la relevancia del término en el documento es máxima.

Nota

Puesto que los términos que aparecen en todos los documentos tendrían un valor de IDF igual a 0, porque el logaritmo de 1 es 0, para evitar casos de multiplicación por 0, podemos encontrar una versión de la fórmula en la que el IDF se describe como $1 + \log(N/n)$, donde N es el número total de documentos y n es el número de documentos que contienen el término *t*.

3.4.3. Cuestiones relativas a la vectorización

Un vectorizador tiene un componente llamado *analizador* (*analyzer*) que se ocupa de establecer las unidades sobre las cuales se calculará un valor numérico (frecuencia o tf.idf). Es importante señalar que el analizador establece las unidades de una colección de textos. Las unidades generalmente son n-gramas que cumplen unas condiciones (por ejemplo, que no sean *stopwords*), aunque pueden ser otras unidades definidas por el analista. Por ejemplo, términos con el *pattern* «N y N + N», como hemos visto en la vectorización de titulares en 3.4.1.

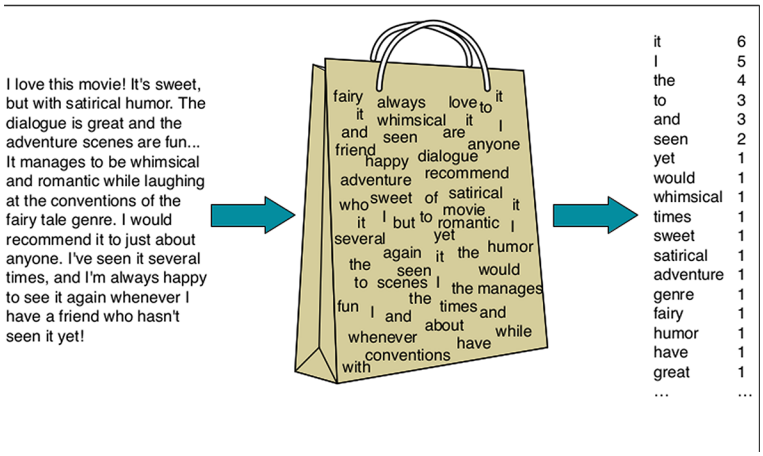
La lista de unidades establecidas por el analizador es el vocabulario de la colección de textos. Cada unidad o término ocupa una posición en la lista. La posición o índice del término no está preestablecido porque el vocabulario sirve para representar los textos según un modelo conocido como *bag of words* o BOW, donde el orden en que aparecen las unidades no es importante. El *bag of words* es una metáfora muy gráfica para entender los textos como un conjunto de términos sin un orden establecido (figura 5).

Bibliografía

Sobre la representación BOW, consúltese el apartado 4.1 del libro de **Jurafsky** y **Martin** (2018).

Una vez obtenido el vocabulario, el vectorizador construye para cada texto de la colección un vector con tantas dimensiones como índices hay en el vocabulario. El vectorizador calcula para cada término del vocabulario un valor numérico que lo representa. Este valor numérico puede ser, como ya se ha dicho, la frecuencia del término en el texto que se vectoriza o su valor de tf.idf. El valor obtenido se pone en el vector en la misma posición que ocupaba el término en el vocabulario, es decir, su índice. El valor numérico es la *feature* y el término que la *feature* describe es la *feature name*.

Figura 5. Ilustración de un texto como un *bag of words* con la frecuencia de cada unidad, en este caso palabra



Fuente: Jurafsky y Martin (2018), pág 65

La siguiente tabla ilustra la vectorización de una colección con dos titulares, que son T1 y T2, que hemos visto anteriormente (c.f. 3.4.1). El vectorizador es un vectorizador tf.idf (véase ejemplo en *notebook* PLA-1 3.4)

Tabla 16. Vectorización de una colección de dos documentos con un *tf.idf vectorizer*

Unidades detectadas por el *analyzer*	*pattern* N, N+N		
Titulares	T1: *Trump proclaims tariffs on steel, and a tariff on aluminium and stocks sag in reply* T2: *Trump embraces a trade war, which could undermine growth*		
Vocabulario	'aluminium', 'growth', 'proclaims*', 'reply', 'steel', 'stock', 'tariff', 'trade', 'trade_war', 'trump', 'war'		
Texto	**Índice**	***Feature name***	**Vector de *features***
T1	0 1 2 3 4 5 6 7 8 9 10	['aluminium', ‘growth’ 'proclaims' 'reply' 'steel' 'stocks' 'tariff', 'trade' ‘trade_war’ 'trump' ‘war’	[0.32 0 0.32 0.32 0.32 0.32 0.64 0 0 0.23 0]

* Problema de WSD. El *parser* ha tomado *proclaims* como un nombre y no como un verbo.

T2	0 1 2 3 4 5 67 8910	'aluminium', ‘growth’ 'proclaims' 'reply' 'steel' 'stocks' 'tariff', 'trade' ‘trade_war’ 'trump' ‘war’	[0.0 0.47 0.0 0.0 0.0 0.0 0.0 0.47 0.47 0.33 0.47]

* Problema de WSD. El *parser* ha tomado *proclaims* como un nombre y no como un verbo.

Es importante tener en cuenta que un vector que representa un documento tiene muchos ceros, porque un gran porcentaje de las palabras del vocabulario no están presentes. Teniendo en cuenta que la longitud de un vector es el número de términos del vocabulario, las longitudes de los vectores pueden ser muy grandes, sobre todo cuando el vocabulario es ingente, como el de la Wikipedia entera o todos los artículos del *New York Times*. Por esta razón, se suele poner un límite de *features* (*max_features*), para reducir el vector a las *features* que tienen un peso más grande.

3.4.4. Recursos

Generalmente, las librerías especializadas en *machine learning* tienen vectorizadores. En Python, una de las librerías de referencia es Sklearn.

En Sklearn, el *term-document-matrix* es distinto del mostrado en las tablas 1.14 y 1.15. El corpus de documentos se representa como una matriz con una fila para cada documento y una columna para cada n-grama, *pattern*, etc. del vocabulario de documentos.

3.4.5. Resultados

En la reunión con Peter, Henry, Anne, Beth y Joseph presentan los resultados de haber vectorizado un documento que recoge todos los titulares TOP con un vectorizador tf.idf. La siguiente tabla muestra los diez primeros términos de los titulares TOP ordenados por su valor de tf.idf. Los resultados muestran que *Trump* es el término más significativo y característico de los titulares TOP.

Figura 6. Diez primeros términos de los titulares TOP ordenados por su tf.idf

	Term	TfIdf
0	trump	0.837212
1	president	0.123119
2	school	0.098496
3	tariff	0.098496
4	say	0.086184
5	trade	0.086184
6	ex	0.073872
7	facebook	0.073872
8	get	0.073872
9	new	0.073872

Peter comenta que le sorprende que la relación entre un titular y el número de comentarios sea tan clara solo con un término. Cree que debe haber algo más, tal vez relacionado con la presencia de palabras como *trade* o *tariff*. De momento, le es difícil saber si los lectores del *New York Times* se animan a comentar por el solo hecho de ver la palabra *Trump* en el titular, o bien hay otros factores que las demás palabras sin contexto no tienen la capacidad de revelar de forma clara. Peter quiere que le despejen esta duda para la la próxima reunión.

Antes de salir del despacho, Peter les dice que quizá la clave sea conocer las connotaciones que tiene la palabra *Trump* en los titulares de *New York Times*. Conocer una palabra no se limita a su denotación. Una palabra como *Trump* se debe conocer en el contexto del ideario del periódico. Al cabo de un rato de estar hablando sobre cómo se conoce una palabra, Peter les dice: «No sé si viene a cuento, pero recuerdo que una vez el lingüista J. R. Firth dijo que una palabra se la conoce por sus compañías, una idea que se remonta a Ludwig Wittgenstein». Sin darle mucha importancia a lo que piensan, que es una más de las citas a las que Peter les tiene acostumbrados, los miembros del equipo deciden irse a comer.

3.5. Vectorización de los términos

Cuando se reúne el equipo, Joseph les recuerda lo que Peter les dijo sobre que una palabra se la conoce por sus compañías. Compañía implica proximidad, por lo cual, si hay que conocer la palabra *Trump* por los titulares del *New York Times*, deben saber qué palabras le son próximas.

La proximidad supone situar elementos en el espacio y, en realidad, tal como reconoce Henry, es lo que han hecho cuando han vectorizado los titulares. Han puesto estos vectores en un espacio, un espacio llamado *espacio vectorial*. Lo mismo debería hacerse con el término *Trump* y con los términos de los

titulares de noticia que provocan más comentarios, habría que vectorizar los términos y situarlos en un espacio vectorial donde poder ver los términos próximos. La pregunta es: ¿cómo convertir un término en un vector?

3.5.1. *One-hot vector*

Una forma directa de convertir un término en un vector es convertirlo en un *one-hot vector*.

> Un *one-hot vector* es un vector de *N* dimensiones, con una sola dimensión con el valor de 1, el resto son 0.

Un término *t* se puede representar como un one-hot vector con tantas dimensiones como términos en el vocabulario. El vector tiene ceros en todas las dimensiones, excepto en la dimensión correspondiente al índice de *t* en el vocabulario.

Imaginemos un vocabulario de diez términos y que el término *Trump* tiene el índice número tres en el vocabulario. La figura 7 muestra el *one-hot vector* que representa *Trump*.

Figura 7. *One-hot vector* que representa el término *Trump* en un vocabulario de diez términos

0	0	1	0	0	0	0	0	0	0

Imaginemos ahora cómo serían los *one-hot vectors* de los términos de los titulares del *New York Times* y de los términos de la Wikipedia. Vectores de centenares de miles, incluso millones de dimensiones, todas con cero y un solo 1.

3.5.2. Word embeddings

> Convertir un término en un vector y ponerlo en un espacio vectorial se llama *word embedding*. En el vector que representa un documento, en el índice correspondiente a un término se insiere (*embed*) el vector que representa la palabra.

Cuando los vectores tienen una dimensionalidad muy grande porque el vocabulario es muy extenso, procesar *one-hot vectors* tiene un coste computacional que no corresponde a los datos que se pueden obtener.

Por ejemplo, sería de esperar que los vectores de dos términos sinónimos fueran iguales o, como mínimo, muy cercanos. Sin embargo, los *one-hot vectors* de dos términos sinónimos pueden ser muy diferentes, porque la única dimensión que no es cero puede ocupar índices dispares en los vectores que representan los dos términos.

3.5.3. *Dense vectors*

El objetivo es representar los términos con vectores de dimensionalidad más reducida que, además, sirvan para capturar las relaciones entre los términos del vocabulario.

Los vectores de dimensionalidad reducida se llaman *dense vectors* y se construyen según la relación de coocurrencia del término con los términos del vocabulario en un contexto.

Un método clásico de creación de *dense vectors* es el SVD (*singularvalue decomposition*). Este método rota los ejes del espacio donde están los vectores. El resultado de la rotación es un nuevo espacio, de manera que la primera dimensión del nuevo espacio muestra la variación mayor. La generación de nuevos espacios y posteriores rotaciones hacen que con un número más reducido de dimensiones se obtenga una representación significativa de los datos a pesar de perder información. Una de las aplicaciones más conocidas del SVD es el Latent Semantic Analysis (LSA). La frecuencia de coaparición de un término con los demás términos del vocabulario en un contexto es el criterio de generación de nuevos espacios con mayor variabilidad.

Bibliografía

En el capítulo 6 del libro de **Jurafsik** y **Martin** se puede encontrar más información sobre *word embeddings* y *dense vectors*. La figura 16.1 del mismo libro, en su edición de 2015, ilustra muy bien la creación de *dense vectors*.

El apartado 6.8 del libro de **Jurafsik** y **Martin** (2018) explica el método Latent Semantic Analysis (LSA) de manera más detallada.

3.5.4. Word2Vec

Enlace de interés

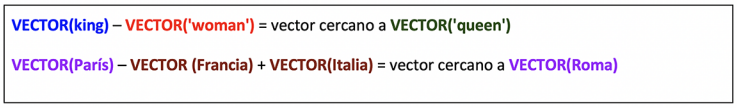
Para ver cómo se implementa el Word2Vec con una red neuronal: https://towardsdatascience.com/learn-word2vec-by-implementing-it-in-tensorflow-45641adaf2ac

Word2Vec es un método que recientemente ha tenido un gran impacto. Es un método que, por un lado, reduce la dimensión de los vectores que representan los términos y, a la vez, captura la relación semántica entre los términos del vocabulario. Además, es un método muy sugerente, porque se basa en un proceso cognitivo que se ha modelado mediante una red neuronal y es representante de una corriente del *machine learning*, llamada *deep learning*, con proyección a la inteligencia artificial.

Si leemos *United States of,* inconscientemente completamos la frase con la palabra *America*. La herramienta que aplica el Word2Vec, también. Es decir, como nosotros, después de haber sido entrenada con muchísimos documentos, la herramienta aprende a generar unos *dense vectors* basados en la probabilidad de que, dada la presencia de *United States of*, aparezca el término del vocabulario *America*. Así aprende a modelar los textos del documento. Concretamente, este método de modelado se llama *Continuous Bag of Words* (CBOW). También aplica el método conocido como *skip-gram*, que calcula la probabilidad de que, dada una palabra, aparezca una combinación determinada de términos del vocabulario. Por ejemplo, dada la palabra *Supremo*, calcula la probabilidad de que le preceda *El Tribunal*.

Lo que ha provocado más impacto ha sido el descubrimiento de las virtudes que tienen los *dense vectors* calculados con estos dos métodos. Situados en un espacio vectorial, los vectores representan muy bien el significado de los términos y las relaciones semánticas entre ellos. Por ejemplo, los vectores de los términos sinónimos (casi) se superponen y los términos semánticamente alejados también lo están en el espacio vectorial. Por su parte, aplicando operaciones de suma o diferencia entre vectores se obtienen vectores semánticamente cercanos a otros términos, con lo cual, se pueden hacer inferencias que relacionan *king* y *queen*, o *Roma* con *París* y *Francia* con *Italia*. Esta capacidad de hacer inferencias con simples operaciones sobre vectores hace que Word2Vec se haya convertido en un método estándar para descubrir relaciones que no son evidentes.

Estas operaciones representan términos en el método Word2Vec:



Los resultados, sobre todo cuando el corpus es grande, confirman el principio de que un término es parecido a otro si los dos comparten contextos similares.

> En la siguiente frase: «el hijacok tiene un sabor dulce», aunque no sepamos qué es el *hijacok*, intuimos que es un alimento, porque comparte contexto con palabras como manzana u otros términos de comida.

Se han hecho estudios para ver si en otros dominios también se cumple este principio, definiendo *contexto* como 'la distribución de la población en una zona geográfica', por ejemplo. De este modo, se puede aplicar a distintos dominios el potencial predictivo y de descubrimiento de relaciones no evidentes que tiene este método (recomendadores, traducción automática, descubrimiento de casuísticas «escondidas» en informes médicos, etc.)

3.5.5. Herramientas para hacer *word embeddings*

Word2Vec es el nombre de la herramienta que Google desarrolló en C++ para aplicar su método. Hay una versión en Python que está en la librería **Gensim**.

Otra aplicación del *word embeddings* es **Glove** (*Global Vectores for Word Representation*), de la Universidad de Stanford, también disponible en Python y en Java (JGloVe). La diferencia respecto a Word2Vec es que Glove cuenta los contextos coocurrentes (*context-counting vectors*) mientras **Word2Vec** los predice (*context-predicting vectors*).

La plataforma **H20** permite también ejecutar Word2Vec en R.

Facebook también ha creado una librería para hacer *word embeddings*. Se llama **Fasttext**, y tiene la particularidad de que toma combinaciones de caracteres (n-gramas de caracteres) como una unidad mínima para hacer los *word embeddings*. También está disponible en Python.

Para abordar retos del procesamiento del lenguaje natural aplicando *deep learning*, es de destacar la librería *open source* TensorFlow, distribuida por Google en lenguaje Python. TensorFlow no solo es útil para explotar las posibilidades de los ***word embeddings***; también se aplica a la traducción automática, los asistentes tipo Siri, Google Assistant o Alexa, el *sentiment analysis*, la generación de resúmenes o la combinación del PLN con el reconocimiento automático de imágenes.

Bibliografía

T. Ganegedara (2018). *Natural Language Processing with TensorFlow*. Packt Publishing.

3.5.6. Detección de términos relevantes con Word2Vec

Henry, Beth, Anne y Joseph deciden hacer un modelo de los titulares TOP con el método Word2Vec. Gracias a este modelo, podrán conocer el término *Trump* y sus compañías. Creen que si sacaran el término *Trump* como si estuviera en un cesto de cerezas, este se llevaría consigo los términos más significativos de los titulares por tener contextos parecidos. Asumen que los vectores *embedded* de estos términos están cerca del de *Trump*.

En el *notebook* PLA-1. 3.5 se muestra la creación del modelo de los titulares de tipo TOP, cómo se entrena con él y la visualización de la distancia entre los vectores de los términos[6]. La representación de los vectores de múltiples dimensiones es difícil de visualizar.

[6]En el *notebook* PLA-1 3.5 se utiliza el algoritmo de la librería Gensim para detectar términos.

Sin embargo, la visualización es posible gracias a una técnica llamada t-SNE (*t-Distributed Stochastic Neighbor Embedding*). La t-SNE reduce las dimensionalidades, de manera que es posible visualizar los datos en un espacio de dos dimensiones sin perder significativamente la información contenida en el espacio de múltiples dimensiones originario. Otros métodos de reducción de dimensiones para hacer visualizaciones es el PCA (*Principal Component Analysis*).

La figura 7 muestra la lista de tuplas con los términos más próximos a *Trump*[7], ordenados según su valor de proximidad. El modelo Word2Vec se ha hecho con términos que aparecen como mínimo tres veces.

[7]Las variantes sinónimas *Donald Trump* y *Mr.Trump* se han aglutinado en el término *Trump*.

Figura 8. Términos más próximos a *Trump* según el modelo Word2Vec de los titulares POP

[('hold', 0.4118765470918369), ('c.e.o', 0.39993705465232043), ('deputy', 0.3538065093214791),
('win', 0.348140536254552), ('america', 0.34293782846949183), ('school',
0.32951093072057625), ('u.s', 0.3212339552410002), ('sex', 0.320808788112185), ('show',
0.30992592790911333), ('team', 0.30042379098722555), ('bolton', 0.2957814267552699),
('woman', 0.295501082936443), ('face', 0.2901126359692892), ('left', 0.2900806517515107),
('debate', 0.28658889897992407), ('ask', 0.2862524160673383), ('get', 0.2801972471163876),
('picture', 0.276551496695057), ('good', 0.2734423779977755), ('charge',
0.26934251410840426), ('lawyer', 0.2679939352346472), ('control', 0.25379695221250415),
('state', 0.24705133456763104), ('trade', 0.2460686002225894), ('race', 0.24421519481754128),
('chief', 0.24408938658380933), ('job', 0.24179929360434121), ('go', 0.239144779273626), ('gun',
0.2365317337489964), ('voter', 0.2348461031520803), ('gun control', 0.23393609899021056),
('facebook', 0.23159944668923804), ('fire', 0.2253085323505727), ('data',
0.21968546009527112), ('join', 0.21903136354333852), ('teacher', 0.21456841760683254), ('say',
0.2120758062710082), ('kushner', 0.2114884766218698), ('fear', 0.21105424485075885), ('stop',
0.21101526085876493), ('take', 0.20955965022835402), ('end', 0.2081539031478748), ('sue',
0.20813092218350437), ('power', 0.20645384934277566), ('plan', 0.20359681167356578),
('problem', 0.2034988668147314), ('college', 0.19522967924544055), ('silence',
0.19045767711048026), ('play', 0.18604973916620057), ('war', 0.18526196017560254), ('target',
0.18514794611908938), ('man', 0.18208618865480622), ('democrat', 0.18025277757698877),
('class', 0.17508857757215443), ('next', 0.1741021257106088), ('call', 0.1613580067020598),
('sign', 0.155942888042117), ('lead', 0.14944439981420052), ('void', 0.1492531471166409),
('new', 0.14164171749431204), ('aide', 0.1414166930866753), ('russia', 0.13515511290878007),
('age', 0.13507114225105687), ('house', 0.13422658386597502), ('porn star',
0.13036370042078124), ('abuse', 0.1236985572061892), ('president', 0.11380626723687277),
('make', 0.11347772784779273), ('trade war', 0.11213267626704823), ('tariff',
0.1032503878981752), ('prison', 0.09837882059707678), ('knew', 0.0881065073912421), ('talk',
0.08054224554315702), ('security', 0.05333730796407229), ('grow', 0.0531161619974365),
('mueller', 0.04004909702423757), ('g.o.p', 0.04004879000162311), ('firm',
0.03684471214451898), ('point', 0.005291083964170051), ('chaos', 0.0018369926847929818)]

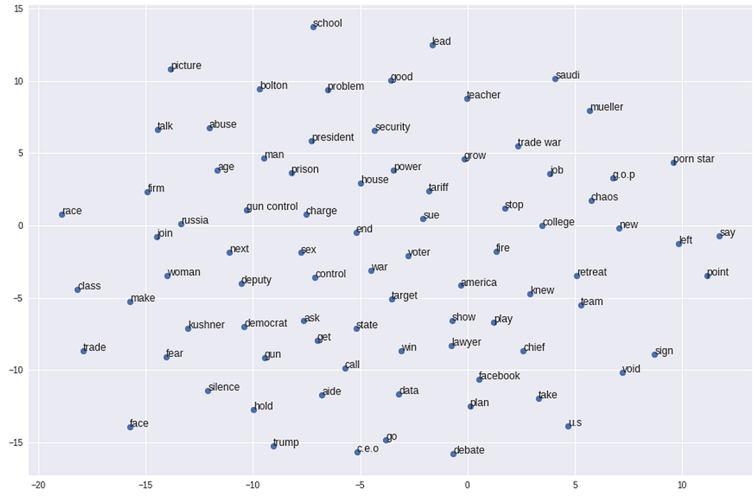
La figura 8 muestra la distancia entre los términos según el modelo Word2Vec de los titulares TOP.

Los resultados obtenidos se pueden interpretar de distintas maneras. Henry, mirando los términos cercanos a Trump, destaca estas relaciones en los titulares TOP:

- Las armas y su control en los centros educativos (‘gun control’, ‘school’, ‘college’)
- El comercio y la guerra comercial (‘trade’, ‘tariff’, ‘trade war’)
- Escándalos como el de la relación con una actriz pornográfica (‘sex’, ‘porn star’) y el uso fraudulento de datos para su campaña electoral (‘data’, ‘facebook’)

Beth, sin embargo, mirando la gráfica de la figura 9, se fija más en el hecho de que los términos del modelo de los titulares TOP están bastante relacionados con la violencia (‘gun’, ‘war’, ‘trade war’) y otros aspectos negativos como ‘problem’, ‘chaos’, ‘fear’, ‘fire’, ‘abuse’ y acciones judiciales, que implican la presencia de conflictos (‘sue’).

Figura 9. Distancia de los términos con una frecuencia mínima de 3 en los titulares TOP según Word2Vec



Ahora bien, Beth considera que el modelo Word2Vec de los titulares TOP todavía es disperso y confuso por la presencia de términos como *go*, *say* o *get*, que aportan muy poco como términos motivadores de comentarios.

4. Detección de temas (*topic detection*)

Beth quiere tener una visión más global y coherente de los términos agrupándolos por temas (*topics*). Según ella, es de esperar que la agrupación por *topics* permita identificar los titulares TOP con un criterio más general y objetivo.

4.1. Detección de temas con Wordnet

Para la detección de los temas, Beth piensa en aprovechar la base de datos léxica Wordnet del inglés. Wordnet recoge el léxico de la lengua inglesa según una ontología en la que los sentidos de las palabras se relacionan con otros sentidos por relaciones de hiperonimia y sinonimia (véanse apartados 3.3.2 y 3.3.3).

La ontología de Wordnet es un grafo acíclico y dirigido (DAG). Cada nodo del grafo es el identificador de un sentido, o *synset*, representado por un conjunto de palabras sinónimas; y cada arco v → w representa una relación que se verbaliza como: «el *synset w* es un hiperónimo del *synset v* o como el *synset v* es un hipónimo del *synset w*».

Beth piensa que, precisamente, esta relación de hiperonimia es adecuada para agrupar palabras según un criterio más lógico y abstracto. Algo que ya le rondaba por la cabeza cuando quería aprovechar la relación de hiperonimia para normalizar términos (c.f. 3.3.3). Si, por ejemplo, se habla de *Corea*, y también de *Rusia*, se pueden agrupar las dos palabras por su hiperónimo, *país*. Así se podrá decir que los temas de los titulares no hablan tanto de Corea y Rusia como de países en general.

En un grafo se puede calcular la distancia semántica entre sentidos según los nodos que hay que atravesar recorriendo los arcos que conducen de un sentido S1 a un sentido S2. Las palabras que son hipónimos directos de un *synset* son semánticamente más cercanas que las palabras que tienen un hiperónimo común muy alejado en el grafo.

> Por ejemplo, *car*, *van*, etc. son más cercanas porque son hipónimos más directos de *vehicle,* que no, en cambio, *car* y *knife* teniendo *artifact* como su hiperónimo común.

La libreria NLTK permite calcular la distancia entre dos sentidos de Wordnet aplicando distintas métricas. Una de ellas es la *Wu-Palmer Similarity*, con un valor que va del 0 al 1. Cuanto más cercano al 1, más cercanos son los sentidos y cuanto más se acerca al 0, más alejados son los sentidos.

Bibliografía

Véase apéndice C.3 del libro de **Jurafsik** y **Martin** (2018) y http://bit.ly/2x6TpLc y http://www.nltk.org/howto/wordnet.html y http://www.nltk.org/howto/wordnet.html

En el *notebook* PLA-1 4.1 enseñamos la aplicación de la Wu-Palmer similarity para calcular la distancia entre *dog* y *cat* y entre *United States* y *Spain*, en sus sentidos más habituales (indicados como .01), cuando su PoS es un nombre (n).

Ejemplo de cálculo de distancia entre dos *synsets* de Wordnet

LA DISTANCIA SEMÁNTICA ENTRE 'DOG' Y 'CAT' ES 0.8571

LA DISTANCIA SEMÁNTICA ENTRE 'UNITED STATES' Y 'SPAIN' ES 0.8

Beth piensa hacer el *topic detection* creando un vector para cada término del vocabulario de los titulares. Las dimensiones del vector recogerán las distancias de los *synsets* del término *t* respecto a los *synsets* de todos los términos t_i del vocabulario de titulares. Una vez se sitúan los vectores de cada término en un espacio vectorial, un algoritmo de *clustering* agrupará los términos semánticamente cercanos y se podrán identificar los temas.

Beth aplica este método para justificar, por su proximidad semántica, la unificación de *fire* y *attacks* en el siguiente titular sobre Trump (véase PLA-1 4), lo que justificaría asimismo que los titulares toman como tema la intolerancia del presidente.

> The "Daily Show" host said the attacks and fire of president Trump on fellow Republicans at a meeting reminded him of "a drunk uncle calling everyone out at a wedding".

Los *synsets* de *fire* y *attack*, cuando son un nombre, forman el vocabulario de *synsets*, que es el siguiente:

[Synset('attack.n.01'), Synset('attack.n.02'), Synset('fire.n.09'), Synset('approach.n.01'), Synset('attack.n.05'), Synset('attack.n.06'), Synset('attack.n.07'), Synset('attack.n.08'), Synset('attack.n.09'), Synset('fire.n.01'), Synset('fire.n.02'), Synset('fire.n.03'), Synset('fire.n.04'), Synset('fire.n.05'), Synset('ardor.n.03'), Synset('fire.n.07'), Synset('fire.n.08'), Synset('fire.n.09')]

Se genera el vector para cada *synset* del vocabulario. Por ejemplo, para crear el vector de Synset('attack.n.01') se crea una matriz que recoge la distancia entre este *synset* y cada uno de los *synsets* del vocabulario (tabla 17).

Tabla 17. Matriz sobre la que se crea el vector de *attack.n.01*

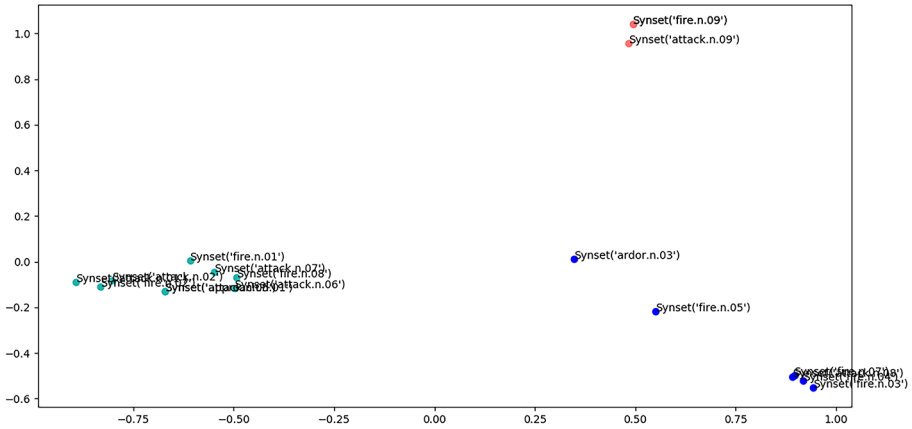
attack. n.01	attack. n.02	fire. n.09'	approach. n.01	attack. n.05	attack. n.06'	attack. n.07	attack. n.08	attack. n.09	'fire. n.01'	fire. n.02	fire. n.03'	fire. n.04'	fire. n.05	ardor. n.03	fire. n.07	fire. n.08	fire. n.09
1.0	0.75	0.26	0.66	0.66	0.55	0.5	0.15	0.26	0.57	0.94	0.11	0.10	0.26	0.26	0.14	0.47	0.26

El vector creado a partir de esta matriz, con los vectores de los demás *synsets*, se ponen en un espacio vectorial y se agrupan en tres *clusters* con el método K-means, como se muestra en la siguiente figura. La figura evidencia que la

(8)Las definiciones de los *synsets* de Wordnet se denominan *glosas*.

hipótesis de Beth es correcta, puesto que fire.n.09, que Wordnet define como *intense adverse criticism*, forma un *cluster* con attack.n.09, que se define como *strong criticism*[8].

Figura 10. Agrupación en *clusters* de los *synsets* de *fire* y *attack*



Henry expone sus dudas sobre la conveniencia de este método de *topic detection* para el objetivo que persiguen. Wordnet es un recurso que se tiene que actualizar con nuevos términos.

Por ejemplo, Trump, con el *synset* de *presidente de un país* no está. En cambio, Kennedy, sí.

4.2. Wordnet, DBpedia y ConceptNet

Beth sugiere añadir la **DBpedia** como un recurso léxico para su método. DBpedia organiza la Wikipedia como una base de datos abierta, gratuita, con una gran comunidad que la mantiene actualizada. La finalidad de la base de datos es poder hacer preguntas sobre hechos y entidades y desarrollar proyectos de procesamiento del lenguaje natural.

Consultando la DBpedia se puede tener la relación entre una entrada de la Wikipedia y un *synset* de Wordnet. Esto es posible gracias a que la DBpedia incorpora información de otra ontología, YAGO, que es el acrónimo de *Yet Another Great Onthology*. YAGO es una ontología desarrollada por el Max Planck Institute for Computer Science en Saarbrücken y uno de sus objetivos es relacionar la Wikipedia con otras ontologías, entre ellas, Wordnet.

La relación entre la Wikipedia y Wordnet declarada en la DBpedia nos permite conocer el *synset* de Wordnet que, según YAGO, le corresponde a la entrada *Donald Trump* de la Wikipedia. Curiosamente, Donald Trump es considerado como una *celebrity* (véase PLA-1 4).

Sin embargo, Henry no acaba de verlo claro. Los temas de muchos titulares relacionan términos ontológicamente muy distantes, como por ejemplo, *Trump*, *scorn*, *FBI*. Él tiene la intuición de que, aunque se pueda decir que Trump pro-

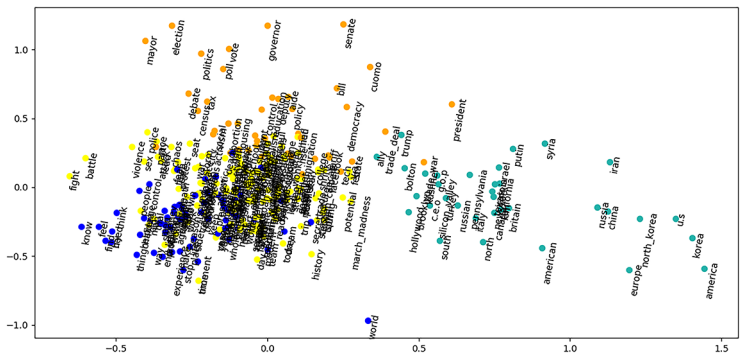
voca muchos comentarios porque es una *celebrity*, una característica de los titulares de las noticias más comentadas es la relación de palabras ontológicamente distantes.

Henry y Beth sí están de acuerdo en considerar que los titulares, como cualquier otro texto, se interpretan gracias a unas relaciones entre palabras que se consideran de sentido común, pero que escapan de la formalización taxonómica de Wordnet. Henry le habla de **ConceptNet**, una red semántica que es el fruto del proyecto colaborativo *Open Mind Common Sense*, que se inició el año 1999 en el MIT Media Lab. Esta red semántica pretende representar las relaciones entre palabras según el sentido común de los hablantes. Para ello, además de fuentes de conocimiento léxico (Wordnet, Wiktionary, DBpedia), añaden conocimiento de relaciones de sentido común que van más allá de estas fuentes y que se han obtenido con juegos de asociaciones de palabras como Verbosity, dentro del **GWAP** project.

Las diferencias entre Wordnet y ConceptNet se manifiestan en el cálculo de la relación semántica entre términos. En el *notebook* PLA-1 4 podemos ver el cálculo de la relación semántica de *United States* y *White House* según Wordnet y ConceptNet. Según Wordnet, cuando *White House* tiene el sentido del departamento ejecutivo del Gobierno de los Estados Unidos, el cálculo basado en las relaciones taxonómicas entre las dos entidades arroja un valor muy pequeño (0.086), mientras que el resultado del cálculo de proximidad según ConceptNet es 0.336, que es más consecuente con el sentido común.

Beth se anima por la posibilidad de que ConceptNet ofrezca resultados de distancia semántica más próximos al sentido común. Por eso decide hacer *clustering* de los términos vectorizando cada término con el cálculo de su distancia semántica respecto al resto de términos. El recurso empleado para calcular las distancias es la API de ConceptNet. En la siguiente figura se puede ver el resultado.

Figura 11. Agrupación en *clusters* de los términos de los titulares con ConceptNet



La gráfica muestra cosas interesantes. Por un lado, se distingue bastante bien un *cluster* que recoge los términos de países y otros términos relativos a la política exterior (*Syria, China, North Korea, Putin*, etc.). Por otro lado, se perfila bastante bien un *cluster* de términos más relacionados con la política y el funcionamiento democrático (*democracy, senate, poll, vote, governor*, etc.), y en la frontera entre ambos está *Trump*.

Las demás clases están más confusas, aunque a Beth no se le escapa el hecho de que términos como *fight*, *battle*, *sex* o *violence* se destaquen de su grupo.

4.3. LDA

Henry prefiere saber qué temas se tratan mediante los mismos artículos sin utilizar ninguna ontología ni recurso externo. Por eso aplica uno de los métodos no supervisados de detección de temas más utilizados: el ***Latent Dirichlet Allocation*** o LDA.

El método LDA se basa en la idea de que en un documento se tratan temas (*topics*) diferentes. El LDA hace el modelo de un documento como una distribución de temas. Para Henry este modelo es más realista porque en los titulares se relacionan temas heterogéneos, la política con el mundo del espectáculo, etc.

Presentamos cómo se hace la *topic detection*:

1) Decidir un número *K* de temas. Después se puede ir afinando el número según los resultados.

2) Para cada documento, se le asigna a cada palabra uno de los *K* temas de forma aleatoria. De esta forma, tenemos la representación de los temas de todos los documentos y la distribución de las palabras en estos temas (aunque no es la distribución óptima).

3) Para mejorarlo, para cada documento *d*.

a) Ir a cada término para actualizar la asignación de un tema según dos criterios:

- Cuánto de prevalente es el término en todos los temas.
- Cuánto de prevalentes son los temas en el documento *d*.

Los dos criterios

Los dos criterios son más extensamente explicados en: http://bit.ly/2x6KNUM

4) Este proceso de mejora se va repitiendo para cada palabra en todos los documentos pasando por toda la colección de documentos muchas veces. Esta actualización iterativa y múltiple asegura que al final se obtenga una distinción coherente de temas.

4.4. Recursos para hacer *topic detection*

Ldatuning es una librería en R que aplica el LDA para hacer *topic detection*. En Python es posible hacerlo también con la librería **Sklearn**, aunque es recomendable **pyLDAvis** si se quiere hacer una presentación clara y atractiva de los *topics* y la contribución de los términos en la detección de estos *topics*.

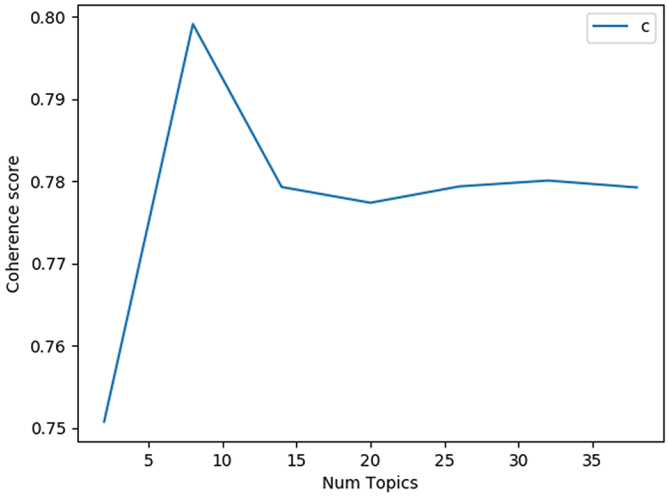
4.5. Resultados

A Henry y a Joseph les interesa aplicar el LDA. A Henry le interesa por las razones que ya hemos explicado, y a Joseph le interesa porque cree que la contribución de un término al *topic* puede ser útil para entrenar un predictor de noticias provocadoras de comentarios. Movidos por este interés, calculan la distribución de los términos[9] de los titulares TOP por temas. Por sugerencia de Beth, se concentran en términos nominales, puesto que los temas de un texto son más fáciles de identificar por medio de sus sintagmas nominales.

[9]Los términos tienen una frecuencia mínima de 3, como en el modelo Word2Vec.

Antes de empezar, deciden que cada titular será un documento, pero tienen que decidir el número de *topics*, esto es, la *K*. Como no se puede saber *a priori* el número de temas sobre los que tratan los titulares, se hace un cálculo del número de *K* aproximado a partir del cual aumentar su número no supone obtener resultados coherentes con el contenido de los titulares. En la figura 12 se muestra que el número de *K* más consecuente con los titulares está alrededor de 7.

Figura 12. Resultado del cálculo de la *K* más coherente para hacer un LDA de los titulares TOP



Las columnas de la figura 13 muestran las diez palabras más probables de pertenecer a un *topic* ordenadas por valor de probabilidad.

Figura 13. Términos de los titulares TOP ordenados según su probabilidad de pertenecer a un *topic*

	1	2	3	4	5	6	7
0	"trump"	"facebook"	"team"	"school"	"get"	"u.s"	"tariff"
1	"trade"	"state"	"aide"	"picture"	"trade war"	"america"	"say"
2	"talk"	"college"	"bolton"	"war"	"problem"	"lawyer"	"plan"
3	"chaos"	"sue"	"target"	"gun"	"control"	"security"	"russia"
4	"point"	"go"	"race"	"democrat"	"face"	"sex"	"gun control"
5	"g.o.p"	"abuse"	"mueller"	"good"	"house"	"lead"	"teacher"
6	"voter"	"call"	"saudi"	"win"	"c.e.o"	"play"	"void"
7	"fear"	"data"	"president"	"age"	"charge"	"power"	"sign"
8	"job"	"firm"	"class"	"left"	"job"	"woman"	"join"
9	"show"	"stop"	"fire"			"president"	

Vemos que las palabras más probables de pertenecer a un *topic* no definen temas distintivos. Más bien volvemos a ver los temas que habíamos visto relacionados con Trump, como el control de armas, la guerra comercial, los escándalos de Facebook y los sexuales. Son términos repartidos por los *topics* sin perfilar un campo temático reconocible que no sea el de los asuntos de Trump. Parece ser, por tanto, que los *topics* gravitan en torno a la figura del presidente.

De todos modos, a Beth le sigue llamando la atención que, entre los términos que por sí mismos no apuntan a un tema concreto, dominen palabras relacionadas con la violencia y el poder (*war*, *power*), o bien tengan connotaciones negativas (*problem, fear, chaos, abuse*).

5. Predicción

El grupo se reúne con Peter, a quien le sorprenden los resultados obtenidos con la detección de *topics*. Ve que *Trump* es el término principal en los titulares tipo TOP y le sorprenden ciertas agrupaciones que se han encontrado en los *topics* (p. ej.: *war* y *pornstar* o *control* y *sex*).

Hacen recopilación de los datos que han ido obteniendo:

1) Lema y PoS de los términos de los titulares

2) El tf.idf de los términos en los titulares TOP

3) La proximidad semántica entre los términos de los titulares TOP

4) Los topics a los que pertenecen los términos de los titulares TOP

Peter pregunta si ya con estos datos sería posible hacer un predictor de noticias motivadoras de comentarios, que es, de hecho, el objetivo último del proyecto. Joseph dice que los titulares ya se han etiquetado como TOP y NOTOP (c. f. 3.4.2), con lo cual, es posible abordar la predicción aplicando *machine learning*. El objetivo sería que el predictor aprendiera la tarea de clasificar titulares TOP y NOTOP.

Peter les dice que se pongan manos a la obra.

5.1. Pasos a realizar

El método para crear el predictor tiene los siguientes pasos:

1) Preprocesado de los titulares hasta tener los datos con los que el clasificador aprenderá a clasificar los titulares.

2) Entrenamiento del predictor.

3) Una vez entrenado, el predictor etiqueta titulares nuevos.

5.1.1. Preprocesado

El preprocesado consiste en:

1) Asociar los documentos con la clase que el clasificador debe distinguir.
2) Limpiar los documentos de caracteres extraños y de *tokens* que no interesan para hacer el análisis (URLS, nombres propios, emoticonos, etc.).

3) Preparar los datos que el clasificador debe aprender.
4) Preparar un corpus de entrenamiento y de test.

1) Asociación de los documentos con su clase. Los documentos deben estar etiquetados según la clase a la que pertenecen y que el clasificador debe aprender a identificar. En el caso que nos ocupa, las clases son dos: TOP y NOTOP. Los documentos TOP son los titulares de noticias que han motivado un número de comentarios que superan un umbral. Los documentos NOTOP son el resto de titulares. La asociación se realiza asignando a los titulares una etiqueta (*data_labelling*) que denota la clase a la que pertenecen.

2) Limpieza del corpus. La limpieza del corpus no supone mucho esfuerzo en el caso de los titulares del *NewYork Times*. Los titulares son tal como aparecieron publicados en el periódico, por lo tanto, no hay que eliminar caracteres especiales, ni emoticonos, ni corregir faltas de ortografía, ni eliminar dobles espacios, etc. El caso sería muy distinto si se tuvieran que preprocesar tuits, *e-mails*, foros y otros documentos de escritura informal, descuidada, con errores de tecleo y con una mezcla de caracteres especiales en los *emojis* con caracteres alfanuméricos.

3) Preparación de los datos. La preparación de los datos se hace con un vectorizador. Para el caso que nos ocupa, Beth, Henry, Joseph y Anne deciden utilizar las *features* de un *tf.idf vectorizer*. El vectorizador obtiene los *features* de los titulares con los procedimientos ya explicados en el apartado 3.4.

4) Preparar un corpus de entrenamiento y de test. El grupo decide preparar el 80 % del total de titulares TOP y NOTOP para el entrenamiento y el 20 % restante lo prepara para evaluar el clasificador (corpus de test). Se distinguen, tanto para el corpus de entrenamiento como el de test, los llamados datos *X*, que son los vectores que describen los documentos, y los datos *Y* (o *target*), que son las etiquetas asociadas a los vectores, esto es, TOP o NOTOP. A continuación, vemos un ejemplo de datos *X* e *Y*.

Tabla 18. Ejemplos de datos *X* y datos *Y*

Texto	Datos *X* (Vectorización del texto con un vectorizador tf.idf)	Datos *Y*
Titular 1	[0, 0, 0.78, 0.23,....]	TOP
Titular 2	[0, 0.45, 0.01, 0.26,....]	NOTOP

5.1.2. Entrenamiento

El entrenamiento consiste en configurar un modelo de los titulares. Hay un buen número de algoritmos para configurar este modelo, algunos son más adecuados que otros para clasificar ciertos fenómenos. Es normal evaluar los resultados de la predicción según un modelo y comparar los resultados obtenidos con los que se obtienen con modelos hechos con otros algoritmos.

De momento, el grupo de Beth, Henry, Joseph y Anne configuran un modelo basado en la *logistic regression model*, también conocido com *logit regression, maximum-entropy classification* o *log-linear classifier*. Este modelo es usado normalmente para clasificaciones con dos valores posibles, como es el caso de TOP y NOTOP. Es similar a la *linear regression*, con la diferencia de que los datos *Y* no son continuos, sino discretos, referidos a una clase. Además, la *regression model*, al contrario que la *linear regression*, permite estimar la probabilidad de que un titular pertenezca a una de las clases.

El clasificador se entrena ajustando los datos (*fit*) del corpus de entrenamiento al modelo establecido.

5.1.3. Predicción

Una vez entrenado, el clasificador predice las etiquetas de los datos *X* del corpus de test. Esto es, a partir de los datos *X* predice sus datos *Y*. A partir de estas predicciones, se evalúa el clasificador en función de cómo se alejan las predicciones respecto a las asignaciones de etiqueta que están en el corpus de test.

También se pueden saber las *features* que han sido más informativas a la hora de clasificar los titulares. Las *features* más informativas nos dan pistas sobre la calidad del clasificador y nos pueden ofrecer información interesante.

5.1.4. Recursos para hacer la predicción

La predicción se puede hacer con la librería de Python NLTK, que importa los clasificadores de la librería Scikit-learn. R tiene librerías ya cargadas que permiten hacer modelos con, por ejemplo, *linear regression* y *logistic regression*. Por otro lado, Apache Spark ofrece API en Java, Scala, Python y R para realizar modelos de clasificación.

5.2. Resultados

De momento, presentamos los resultados obtenidos por Beth, Henry, Joseph y Anne sobre los *features* más informativos para realizar la clasificación (véase *notebook* PLA-1, 5).

Figura 14. Informatividad de los términos en la clasificación de titulares TOP y NOTOP

NONTOP	-0.8058405273727582	episode
NONTOP	-0.7534346524087436	black
NONTOP	-0.7073540060974302	season
...		
TOP	**3.6327081228639**	**trump**
TOP	1.2363090722630365	picture
TOP	1.1256915185912726	president
TOP	1.0921666744283134	job
TOP	0.9698316413604575	voter
TOP	0.9176627731358946	school
TOP	0.9136615761705841	bolton
TOP	0.9098591448319565	chaos
TOP	0.8632258255927439	college
TOP 0.8446625962106591 trade		

Según estos resultados, se evidencia una vez más la importancia de *Trump* a la hora de predecir los titulares con más comentarios. Por otro lado, al comparar los valores de informatividad de los titulares etiquetados como TOP con los valores muy inferiores de los titulares NONTOP, se confirma que los *features* más informativos caracterizan sobre todo los titulares de noticias que motivan más comentarios.

De repente Peter siente un escalofrío. Por un momento ha visto pasar el temible espectro del *overfitting*. El hecho de que el término más informativo sea *Trump*, que es una referencia muy puntual a una persona, puede ser un aviso de que hayan hecho un clasificador muy bueno para predecir las etiquetas de los titulares con los que se ha entrenado, pero sea malo para predecir las etiquetas de titulares nuevos. En la evaluación del sistema habrá que comprobar que no se produce *overfitting*.

Sin embargo, Peter y su equipo reconocen que el sistema seguramente será un clasificador *one-shot* dependiente de los cambios en el tiempo, las tendencias y los temas de actualidad. Peter se pregunta: «y cuando no esté Trump, ¿servirá nuestro clasificador?» Los miembros de su equipo, excepto Beth, se encogen de hombros: «Habrá que actualizarlo cada *X* tiempo», dicen. Sin embargo, Beth, en su fuero interno, piensa que es posible hacer un clasificador donde no le afecte qué o quién es noticia en un momento concreto.

Resumen

En este módulo hemos introducido los métodos de procesamiento de lenguaje natural más comunes para procesar información textual y obtener datos para tomar una decisión. La presentación de estos métodos ha tenido un hilo argumental, que ha sido un hipotético proyecto para el *New York Times* que quiere automatizar la inclusión de publicidad en las noticias que provocan muchos comentarios. El periódico subcontrata un equipo de lingüistas computacionales para desarrollar un prototipo de predictor de noticias motivadoras de comentarios.

Hemos presentado métodos considerados básicos, como la detección de términos y colocaciones, el etiquetaje según la categoría gramatical, la lematización y el etiquetado semántico. Luego hemos explicado métodos de transformación de los datos lingüísticos a representaciones que facilitan el procesamiento de estos datos y la obtención de información relevante. Por un lado, hemos explicado la transformación de los documentos y de los términos en vectores y, por otro lado, hemos explorado métodos de cálculo para saber la relevancia de los términos en un documento y la relación semántica entre los términos de un documento. Lo hemos hecho comparando el cálculo de la relación semántica en un espacio vectorial con la relación semántica en una ontología como Wordnet. También hemos presentado un método de agrupación de palabras por temas.

Finalmente, hemos presentado un método de *machine learning* para predecir datos vinculados a las opiniones, como ha sido predecir si una noticia motivará muchos comentarios. La predicción se ha hecho con un clasificador de textos. El método de predicción ha motivado cuestiones importantes como el peligro del *overfitting* y también la consideración del clasificador como un sistema que debe ir actualizándose con nuevos datos cada cierto tiempo.

De momento, hemos enseñado un método de clasificación. Ahora bien, hay muchos métodos alternativos y habrá que evaluar los resultados obtenidos con varios de ellos para elegir el más adecuado.