Procesamiento de Lenguaje Natural o Minería de textos
Tema 5: Modelo de espacio vectorial.

Objetivo: El participante identificará cómo mejorar los resultados en una tarea de similitud semántica de textos o ser de gran ayudar para distintas tareas relacionadas con el procesamiento de textos en lenguaje natural, a partir de los modelos basados en la obtención de vectores de palabras y documentos.

Temario:

1. Vectores de palabras (*word embeddings*)
2. Similitud de palabras
3. Vectores de documentos
4. Similitud de documentos
5. Visualización y PCA (Análisis de Componentes Principales)

Lecturas:

García Ferrero, I. (2018). Estudio de *word embeddings* y métodos de generación de *meta embeddings*. Obtenido de https://addi.ehu.es/bitstream/handle/10810/29088/MemoriaTFG_IkerGarciaFerrero.pdf?isAllowed=y&sequence=3

Justicia de la T., M. d. (2017). Nuevas Técnicas de Minería de Textos: Aplicaciones. Universidad de Granada. Tesis Doctorales. Obtenido de http://hdl.handle.net/10481/46975

López Solaz, T., Troyano Jiménez, J. A., Ortega Rodríguez, F. J., & Enríquez de Salamanca Ros, F. (2016). Una aproximación al uso de *word embeddings* en una tarea de similitud de textos en español. Obtenido de http://rua.ua.es/dspace/handle/10045/57753

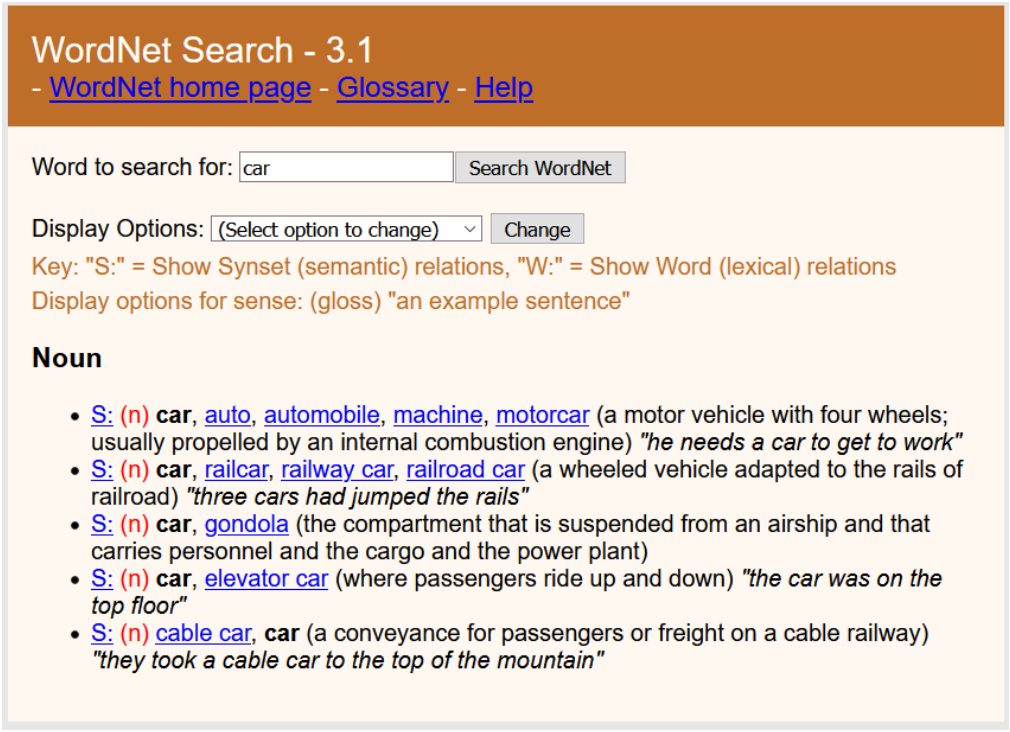
Introducción

La mayoría de los sistemas de reglas y sistemas estadísticos de PLN tradicionales, representan las palabras como unidades discretas y atómicas. Este tipo de codificación no aporta al sistema ningún tipo de información sobre como las diferentes palabras pueden estar relacionadas entre ellas. El sistema cuando procese información de animales no podrá saber que un humano se trata de un animal, que a su vez es un mamífero. Representar las palabras como variables únicas y discretas no nos aporta la suficiente información para ser capaces de entrenar modelos capaces de realizar tareas de procesamiento de lenguaje complejas de forma satisfactoria.

Las distintas aproximaciones de similitud a nivel de palabras se agrupan en dos categorías: **similitud léxica de palabras y similitud semántica de palabras**. Dos palabras son similares a nivel léxico si están compuestas por secuencias parecidas de caracteres. Para determinar la similitud léxica de palabras se

suelen utilizar distintas métricas basadas en comparación de cadenas de caracteres. La similitud semántica de palabras, por su parte, nos permite medir si dos palabras tienen significados parecidos o se usan en contextos parecidos. Hay dos grandes *grupos de técnicas para calcular la similitud semántica de palabras*: técnicas basadas en conocimiento y técnicas basadas en corpus.

Las técnicas de similitud de palabras basadas en conocimiento miden el grado de parecido entre palabras apoyándose en algún tipo de recurso lingüístico que proporcione información sobre el significado de las palabras. El recurso por excelencia es *WordNet* (Miller,1995[1]), una base de datos léxica organizada en torno a varias relaciones entre palabras. La relación de sinonimia es la más importante y en ella se apoya el concepto de synset (grupo de sinónimos) que permite definir de forma implícita el significado de las palabras a través del conjunto de synsets en los que aparece. Los synsets están interconectados mediante relaciones conceptual – semánticas y léxicas; es decir un synsets contiene una breve definición y, en la mayoría de los casos, una o más oraciones cortas que ilustran el uso de los miembros del synset. Las palabras con varios significados distintos se representan en tatos synsets distintos. Por lo tanto, cada par de forma – significado es WordNet es único.



WordNet online. Obtenido de: http://wordnetweb.princeton.edu/perl/webwn

WordNet distingue entre Tipos (sustantivos comunes) e Instancias (personas específicas, países y entidades geográficas). Por lo tanto, el sillón es un tipo de silla, Barack Obama es una instancia de un presidente. Las instancias son siempre nodos de hoja (terminal) en sus jerarquías.

[1] Miller, G. 1995. Wordnet: a lexical database for english. Communications of the ACM,38(11):39–41.

Meronimia (sustantivos), la relación entre la parte y el todo se mantiene entre los synsets. Las partes se heredan de sus superiores: si una silla tiene patas, entonces un sillón también tiene patas. Las piezas no se heredan "hacia arriba", ya que pueden ser características solo de tipos específicos de cosas en lugar de la clase en su conjunto: las sillas y los tipos de sillas tienen patas, pero no todos los tipos de muebles tienen patas.

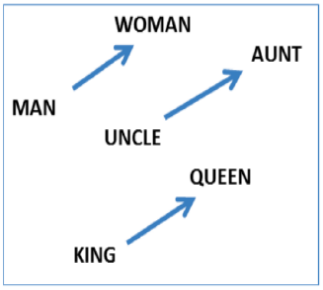
Verbos: Los synsets de verbos también se organizan en jerarquías; Los verbos hacia el fondo de los árboles (tropónimos) expresan maneras cada vez más específicas que caracterizan un evento, como en {comunicarse} - {hablar} - {susurro}. Volumen (como en el ejemplo anterior) es solo una dimensión a lo largo de la cual se pueden elaborar verbos. Otros son la velocidad (mover-trotar-correr) o la intensidad de la emoción (querer-amar-idealizar). Los verbos que describen eventos que necesaria y unidireccionalmente se relacionan entre sí están vinculados: {comprar} - {pagar}, {lograr} - {intentar}, etc.

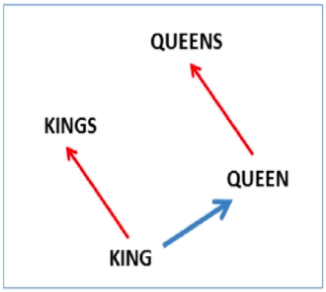
Adjetivos: Los adjetivos están organizados en términos de antonimia. Los pares de antónimos "directos" como húmedo-seco y joven-viejo, reflejan el fuerte contrato semántico de sus miembros. Cada uno de estos adjetivos polares, a su vez, está vinculado a una serie de "semánticamente similares". Seco está vinculado a: áridos, desecados y húmedo a empapados, etcétera. Los adjetivos semánticamente similares son "antónimos indirectos" del miembro del polo opuesto.

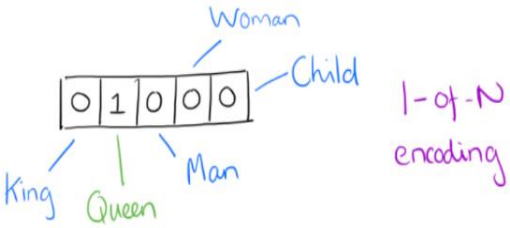
Adverbios: Hay pocos adverbios en WordNet (difícilmente, seguramente, realmente, etcétera.) ya que la mayoría de los adverbios en inglés se derivan directamente de los adjetivos a través de la fijación morfológica (sorprendentemente, extrañamente, etcétera.)

Ejercicio5(es)-Modelo de Espacio Vectorial.ipynb

Las técnicas de similitud de palabras basadas en corpus determinan el parecido semántico de dos palabras en función de los usos de esas palabras en una gran colección de textos. Por lo general, estas técnicas se basan en algún tipo de representación vectorial de las palabras en función de los distintos contextos en los que dichas palabras aparecen. Dentro de las técnicas basadas en corpus destacan las de *latent semantic analysis* (LSA) y las de *word embedding*. LSA analiza las relaciones entre un conjunto de documentos y los términos que contienen, estableciendo que dos palabras son similares si ocurren en fragmentos similares de textos. LSA parte de una matriz de palabras frente a documentos y aplica una técnica matemática denominada *singular value decomposition* que permite reducir el número de filas (documentos) preservando la similitud entre columnas (palabras). Por su parte, las técnicas de *word embedding* parten de representaciones BOW (*bag of words*) de los distintos contextos de las palabras para obtener representaciones vectoriales de las palabras de dimensiones mucho más reducidas que capturan el significado y las relaciones entre palabras.





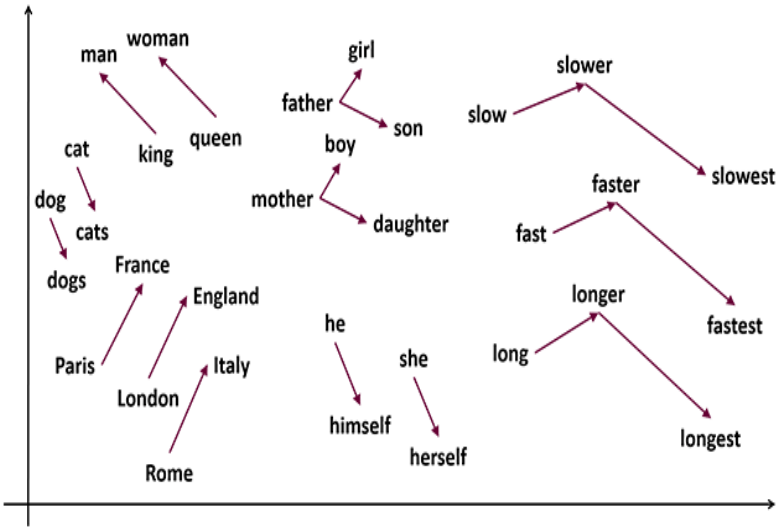


Vector de palabras: Es simplemente un vector de pesos. Ejemplo: Codificación 1 de N. Cada elemento del vector es asociado con una del vocabulario.

Los vectores son muy buenos para responder preguntas del tipo: *A es a B como C es a…*

La similaridad entre dos palabras viene en general dada por la distancia coseno entre los vectores. Los resultados se prueban con listas de similaridad[2] y relacionalidad[3] elaboradas por humanos. Ejemplo: *hombre es a mujer como tío es a…*

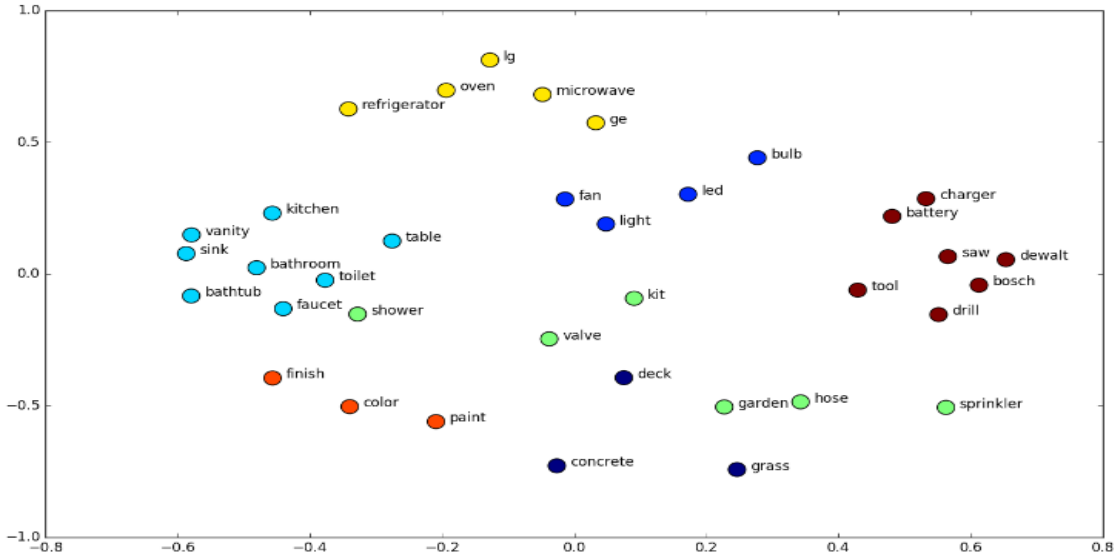
La idea detrás de estos métodos es que si podemos predecir en qué contexto aparece una palabra, entonces significa que entendemos el significado de la palabra en su contexto. Así, las palabras se representan en espacios vectoriales donde palabras semánticamente similares se encontrarán cerca entre ellas.



Representación en 2D de algunos word embeddings de palabras

Las técnicas de *word embedding* han demostrado ser muy útiles en múltiples tareas del Procesamiento del Lenguaje Natural aparte de la similitud de textos[4][5], y en la actualidad gozan de gran popularidad. Gracias a ello se han convertido en la base de sistemas de traducción automática neuronal[6], clasificación de textos[7], recuperación de información, sistemas de preguntas – respuestas, etc.

[2] Similaridad Semántica: significan lo mismo (chico/pequeño, coche/carro). Similaridad Fonética: suenan parecido (cuchara/cucaracha)
[3] Relacionalidad: dos palabras están relacionadas en la conciencia lingüística de los hablantes (gato/perro, animal, cola; frío/ caliente, abrigo; sopa/comer)
[4] Collobert, R., J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu y P. Kuksa. 2011.Natural language processing (almost) from scratch.J. Mach. Learn. Res.,12:2493–2537, Noviembre.
[5] Zou, W., R. Socher, D. Cer, y C. Manning. 2013. Bilingual word embeddings for phrase-based machine translation. En EMNLP, páginas 1393–1398.
[6] Mikel Artetxe, Gorka Labaka, Eneko Agirre, and Kyunghyun Cho. Unsupervised neural machine translation. In Proceedings of the Sixth International Conference on Learning Representations, April 2018.
[7] Xiang Zhang, Junbo Zhao, and Yann LeCun. Character-level convolutional networks for text classification. pages 649–657, 2015.



Métodos de generación de *word embeddings*:

En la tabla se muestran algunos de los métodos de generación de *word embeddings* vienen normalmente acompañados por vectores pre - calculados:

- ✓ Word2Vec[8]

https://code.google.com/archive/p/word2vec/

- ✓ FastText[10]

https://fasttext.cc/docs/en/english-vectors.html

- ✓ GLOVE[12]

https://nlp.stanford.edu/projects/glove/

- ✓ LEXVEC[14]

- ✓ PDC y HDC[9]

http://ofey.me/projects/wordrep/

- ✓ SketchEngine[11]

https://embeddings.sketchengine.eu/static/index.html

- ✓ UKB[13]

http://ixa2.si.ehu.eus/ukb/

- ✓ Numberbatch[15]

[8] Greg Corrado Tomas Mikolov, Kai Chen and Jeffrey Dean. Efficient estimation of word representations in vector space.ar Xiv preprint arXiv:1301.3781v3, 2013.
[9] Fei Sun, Jiafeng Guo, Yanyan Lan, Jun Xu, and Xueqi Cheng. Learning word re-presentations by jointly modeling syntagmatic and paradigmatic relations. pages136–145, 2015.
[10] Tomas Mikolov, Edouard Grave, Piotr Bojanowski, Christian Puhrsch, and Armand Joulin. Advances in pre-training distributed word representations. In Proceedings of the International Conference on Language Resources and Evaluation (LREC 2018),2018.
[11] Sketch engine. https://www.sketchengine.eu/
[12] Jeffrey Pennington and Richard Socher and Christopher D. Manning. GloVe: Global Vectors for Word Representation. Empirical Methods in Natural Language Processing (EMNLP). pages 1532–1543, 2014.
[13] Josu Goikoetxea, Aitor Soroa, and Eneko Agirre. Random walks and neural net-work language models on knowledge bases. In Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pages 1434–1439. Association for Computational Linguistics, 2015.

https://github.com/alexandres/lexvec

- ✓ jointcHYB[16]

https://github.com/commonsense/conceptnet-numberbatch

- ✓ Context2Vec[17]

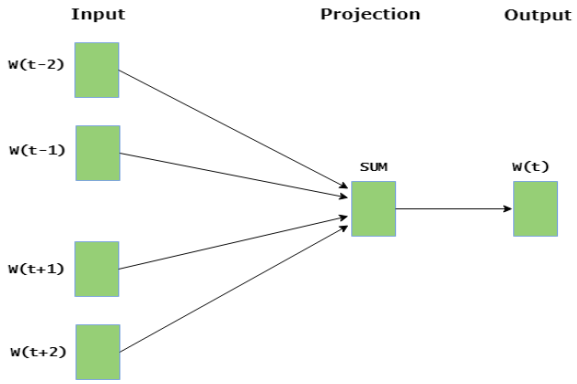
http://ixa2.si.ehu.eus/ukb/bilingual_embeddings.html

https://u.cs.biu.ac.il/~nlp/resources/downloads/context2vec/

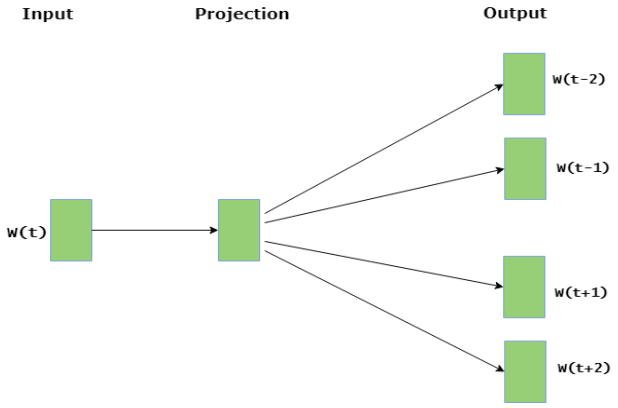
Word2Vec: Se trata de un modelo predictivo de generación de *word embeddings*. Word2Vec implementa dos modelos neuronales: CBOW y Skip-gram. En el primero, dado el contexto de la palabra objetivo, intenta predecirla. En el segundo, dada la palabra intenta predecir el contexto. Las capas internas de la red neuronal codifican la representación de la palabra objetivo, es decir, los *word embeddings*.

Vectores entrenados en el *dataset* Google News, disponible en: https://code.google.com/archive/p/word2vec/

- Cuenta con alrededor de 100 mil millones de palabras y contiene vectores de 300 dimensiones para 3 millones de palabras y frases.



El modelo CBOW predice la palabra actual dada las palabras de contexto dentro de una ventana específica. La capa de entrada contiene las palabras de contexto y la capa de salida contiene la palabra actual. La capa oculta contiene el número de dimensiones en las que queremos representar la palabra actual presente en la capa de salida.



Skip gram predice las palabras de contexto circundantes dentro de una ventana específica dada la palabra actual. La capa de entrada contiene la palabra actual y la capa de salida contiene las palabras de contexto. La capa oculta contiene el número de dimensiones en las que queremos representar la palabra actual presente en la capa de entrada.

[14] Marco Idiart Alexandre Salle and Aline Villavicencio. Matrix factorization using window sampling and negative sampling for improved word representations.ACL,2016.
[15] Robert Speer and Joanna Lowry-Duda. Concept net at semeval-2017 task 2: Extending word embeddings with multilingual relational knowledge.CoRR, ab-s/1704.03560, 2017.
[16] Josu Goikoetxea, Aitor Soroa, and Eneko Agirre. Bilingual embeddings with random walks over multilingual wordnets. Know.-Based Syst., 150(C):218–230, June2018.
[17] Ido Dagan Oren Melamud, Jacob Goldberger. context2vec: Learning generic context embedding with bidirectional lstm. 2016.

FastText: *FastText* es una extensión del modelo Word2Vec. Cada palabra es tratada como la suma de sus composiciones de caracteres llamados ngrams. El vector para una palabra está compuesto por la suma de sus ngrams. Por ejemplo, el vector para la palabra "apple" está compuesto por la suma los vectores para los ngrams *"<ap, app, appl, apple, apple>, ppl, pple, pple>, ple, ple>, le>*". De esta forma se espera obtener mejores representaciones para palabras "raras", las cuales cuentan con muy pocas apariciones en corpus de textos, y así poder generar vectores para palabras que no se encuentran en el vocabulario de los *word embeddings*. →

Los vectores disponibles en: https://fasttext.cc/docs/en/english-vectors.html

- FastText calculados en el corpus *common crawl* usando 600 mil millones de tokens; cuentan con representaciones de 300 dimensiones para 2 millones de palabras.
- FastText calculados usando los corpus wikipedia, UMBC y statmt.org; cuentan con representaciones de 300 dimensiones para 1 millón de palabras. Han sido entrenados usando 16 mil millones de tokens.

GLOVE: GLOVE a diferencia de Word2Vec, es un modelo basado en conteo. GloVe genera una matriz de gran tamaño donde se almacena la información de la concurrencia entre palabras y contextos; es decir, para cada palabra contamos cuantas veces aparece dicha palabra en algún contexto. El objetivo de entrenamiento de dicha matriz es aprender vectores de forma que el producto escalar entre las palabras sea igual al logaritmo de la probabilidad de coocurrencia entre las palabras. El número de contextos es muy alto. Por lo tanto, se realiza una factorización de dicha matriz para obtener una de menores dimensiones, obteniendo así un vector que representa a cada una de las palabras. La ventaja de GLOVE sobre Word2Vec es que es más sencillo paralelizar el entrenamiento. Por lo tanto, es posible usar más información durante el entrenamiento, por lo que es posible usar una mayor cantidad de datos durante el entrenamiento.

Los vectores disponibles en: https://nlp.stanford.edu/projects/glove/

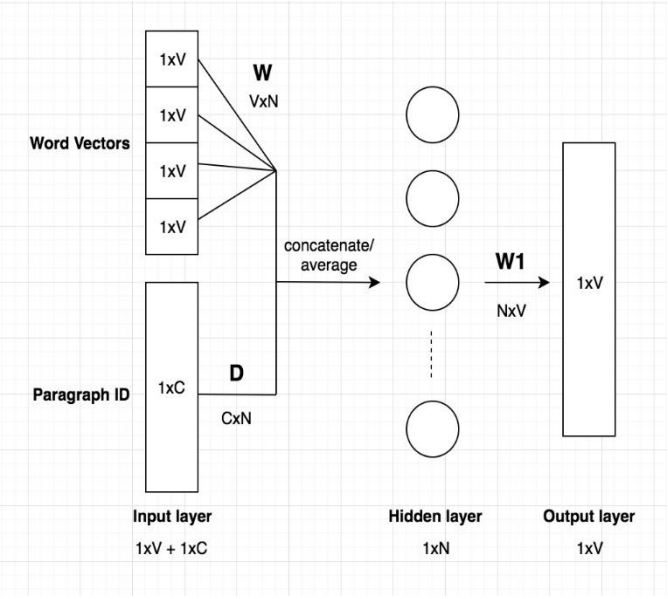
- Con representación de 300 dimensiones calculados en el corpus *common crawl*, usando 840 mil millones de tokens y cuenta con representación para 2,2 millones de palabras.
- Con representación de 300 dimensiones calculados en el corpus *common crawl*, usando 42 mil millones de tokens y cuenta con representación para 1,9 millones de palabras.
- Existen versiones con 50, 100, 200 y 300 dimensiones, usando los corpus Wikipedia (2014) y Gigaword5, con 6 mil millones de tokens y cuentan con representaciones para 400 mil palabras.
- Un corpus formado por 2 millones de tweets extraídos de la red social Twitter, usando 27 mil millones de tokens y cuentan con un vocabulario de 1,2 millones de palabras; disponibles en 25, 50, 100 y 200 dimensiones.

Para más información, se muestra en (García, 2018[18]), una tabla resumen de las principales características de cada *word embedding.*

Ejercicio5(es)-Modelo de Espacio Vectorial.ipynb

Vector de documentos

El concepto de **Doc2Vec** es bastante simple, si ya está familiarizado con el modelo de Word2vec. El modelo Doc2vec se basa en Word2Vec, con solo agregar otro vector (ID de párrafo) a la entrada. La arquitectura del modelo Doc2Vec se muestra en la figura.



El diagrama se basa en el modelo CBOW, pero en lugar de usar solo palabras cercanas para predecir la palabra, también agregamos otro vector de características, que es único en el documento. Por lo tanto, al entrenar los vectores de palabras W, el vector de documento D también se entrena y, al final del entrenamiento, contiene una representación numérica del documento.

Las entradas consisten en vectores de palabra y vectores de identificación de documento. La palabra vector es un vector *one-hot* con una dimensión $1xV$. El vector Id del documento tiene una dimensión de $1xC$, donde C es el número total de documentos. La dimensión de la matriz de peso W de la capa oculta es VxN. La dimensión de la matriz de peso D de la capa oculta es CxN.

El modelo anterior se llama *Distributed Memory version of Paragraph Vector* (PV-DM). Otro algoritmo de Doc2Vec que se basa en Skip-Gram se llama *Distributed Bag of Words version of de Paragraph Vector* (PV-DBOW).

Ejercicio5(es)-Modelo de Espacio Vectorial.ipynb

PCA (Análisis de Componentes Principales)[19][20][21]

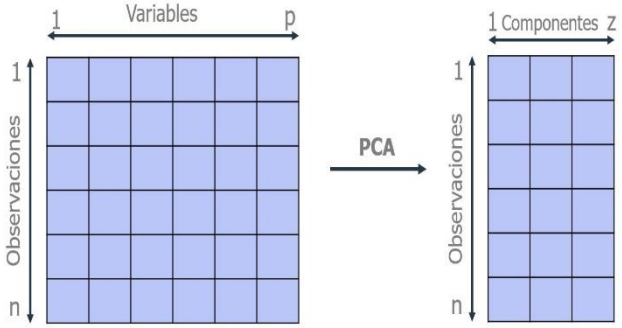
[18] García Ferrero, I. (2018). Estudio de word embeddings y métodos de generación de meta embeddings. Pág. 17. Obtenido de https://addi.ehu.es/bitstream/handle/10810/29088/MemoriaTFG_IkerGarciaFerrero.pdf?isAllowed=y&sequence=3

[19] Análisis de componentes principales PCA con Python by Joaquín Amat Rodrigo, available under a Attribution 4.0 International (CC BY 4.0) at https://www.cienciadedatos.net/documentos/py19-pca-python.html
Análisis de Componentes Principales (Principal Component Analysis, PCA) y t-SNE by Joaquín Amat Rodrigo, available under a Attribution 4.0 International (CC BY 4.0) at https://www.cienciadedatos.net/documentos/35_principal_component_analysis

[20] Visualice video explicativo: Procesamiento de datos (PCA), disponible en: https://www.youtube.com/watch?v=b1NGM3IbRcI

[21] Comprende Principal Component Analysis en Aprende Machine Learning. Disponible en: https://www.aprendemachinelearning.com/comprende-principal-component-analysis/

El análisis de componentes principales (*Principal Component Analysis* PCA) es un método de reducción de dimensionalidad que permite simplificar la complejidad de espacios con múltiples dimensiones a la vez que conserva su información.



Supóngase que existe una muestra con n individuos cada uno con p variables $(X_1, X_2, \ldots, X_p)$, es decir, el espacio muestral tiene p dimensiones. PCA permite encontrar un número de factores subyacentes $(z < p)$ que explican aproximadamente lo mismo que las p variables originales. Donde antes se necesitaban p valores para caracterizar a cada individuo, ahora bastan z valores. Cada una de estas z nuevas variables recibe el nombre de componente principal.

El método de PCA permite por lo tanto "condensar" la información aportada por múltiples variables en solo unas pocas componentes. Aun así, no hay que olvidar que sigue siendo necesario disponer del valor de las variables originales para calcular las componentes. Dos de las principales aplicaciones del PCA son la visualización y el preprocesado de predictores previo ajuste de modelos supervisados.

La librería *scikitlearn* contiene la clase *sklearn.decomposition.PCA,* que implementa la mayoría de las funcionalidades necesarias para crear y utilizar modelos PCA.

Con PCA se obtiene:

1. Una medida de como cada variable se asocia con las otras (matriz de covarianza)
2. La dirección en las que nuestros datos están dispersos (autovectores)
3. La relativa importancia de esas distintas direcciones (autovalores)

PCA combina los predictores y permite deshacernos de los autovectores de menor importancia relativa.

Como contras, el algoritmo de PCA es muy influenciado por los *outliers*[22] en los datos. Por esta razón, surgieron variantes de PCA para minimizar esta debilidad. Entre otros se encuentran: *RandomizedPCA*, *SparcePCA* y *KernelPCA*.

Por último, citar que PCA fue creado en 1933 y ha surgido una buena alternativa en 2008 llamada *t-SNE* con un enfoque distinto.

Ejercicio5(es)-Modelo de Espacio Vectorial.ipynb

[22] En estadística, tales como muestras estratificadas, un valor atípico (en inglés *outlier*) es una observación que es numéricamente distante del resto de los datos. Los valores atípicos pueden ser indicativos de datos que pertenecen a una población diferente del resto de las muestras establecidas.

Conclusiones

Muchas de las tareas de recuperación de información como la búsqueda, agrupamiento o categorización de textos tienen como primer objetivo procesar documentos en lenguaje natural. El problema que surge es que los algoritmos que pretenden resolver estas tareas, necesitan representaciones internas explícitas de los documentos. En el área de recuperación de información normalmente se usa una expresión vectorial, donde las dimensiones del vector representan términos, frases o conceptos que aparecen en el documento. En este aspecto la representación más adoptada es la conocida como bolsa de palabras: una colección de documentos compuesta por n documentos indexados y m términos representados por una matriz documento-término de $n\,x\,m$. Donde los n vectores renglón representan los n documentos; y el valor asignado a cada componente refleja la importancia o frecuencia ponderada que produce el término, frase o concepto t_i en la representación semántica del documento j.

En esta representación vectorial de documentos el éxito o fracaso se basa en la ponderación o peso de los términos. Aunque ha habido mucha investigación sobre técnicas de ponderación de términos, en realidad no hay un consenso sobre cuál método es el mejor. También hay que destacar que el espacio de renglones de la matriz documento-término determinan el contenido semántico de la colección de documentos. Sin embargo, una combinación lineal de dos vectores-documento no representa necesariamente un documento viable de la colección. Más importante aún, mediante el modelo espacio vectorial se pueden explotar las relaciones geométricas entre dos vectores documento (y términos) a fin de expresar las similitudes y diferencias entre términos.

Si bien el rendimiento de un sistema de recuperación de información depende en gran medida de las medidas de similitud entre documentos, la ponderación de términos desempeña un papel fundamental para que esa similitud entre documentos sea más confiable. Así, por ejemplo, mientras que una representación de documentos basada solo en las frecuencias o apariciones de términos no es capaz de representar adecuadamente el contenido semántico de los documentos, la representación de términos ponderados (Aplicación de métodos de normalización a la matriz documento-término) hace frente a errores o incertidumbres asociadas a la representación simple de documentos.

El modelo de espacio vectorial tiene las siguientes limitaciones:

1. Los documentos largos quedan poco representados ya que contienen pocos valores en común (un producto escalar menor y una gran dimensionalidad)
2. Las palabras de búsqueda deben coincidir con las palabras del documento; partes de una palabra pueden dar en falsos positivos.
3. Sensibilidad semántica, documentos con contextos similares, pero con diferente vocabulario no serán asociados, resultando en falsos negativos