**Trabajo Practico**

**Procesamiento del Lenguaje Natural**

**Trabajo realizado:** Detector de plagio entre documentos.

**Alumno:** Leonardo Guerberg.

**Profesor:** Hernán Borre.

**Repositorio código:** <https://github.com/leoguerberg/academic_plagiarism_detector>

**Índice**

1. Representación vectorial de las palabras manteniendo el contexto
   1. CBOW
   2. Skipgram
2. Entrenando un modelo utilizando Word2Vec
3. Desarrollo del detector de plagios.
   1. Usando un modelo ya entrenado – Fast Text.
   2. Utilizando una implementación propia con Wordnet.
4. Named Entity Recognition.
5. Clasificación.
6. Scraping.
7. Resultados.
8. Fuentes.

**Representación vectorial de las palabras manteniendo el contexto**

El primer problema con el que me enfrente a la hora de empezar con este proyecto fue encontrar alguna manera en que la computadora entienda el “significado” u “objetivo” de una palabra dentro de una oración en particular. Dicho de otra forma, que pueda asociar dos palabras si son usadas en contextos muy similares.

Para esta tarea me puse a revisar trabajos realizados por otros investigadores respecto a este tema.

Ferrero *et al. [1]* habla sobre la idea de “Word Embeddings”, cuyo objetivo es la representación vectorial de una palabra basada en las palabras que la rodean. Estas son proyectadas en un espacio continuo donde aquellas con un contexto similar deberían estar cerca.

Suleiman *et al. [2]* se centra en explicar cómo se podría calcular la similitud entre dos palabras si ya se tiene una representación de estas, usando técnicas como la similitud del Coseno, de Dice o de Jaccard.

Para unir estas dos ideas, usé el concepto de Word Embedding nombrado antes y di con lo que se conoce como Word2Vec [3]. Este se basa en el uso de una red neuronal donde los inputs son palabras de un alfabeto en formato One Hot [4] y el resultado es un vector de N dimensiones que para calcularse tendrá en cuenta el contexto que rodea a la palabra.

Dos algoritmos distintos para aplicar Word2Vec son Skipgram y CBOW. La diferencia principal que yace entre estos es que el CBOW intenta predecir la representación de una palabra dado el contexto y Skipgram busca representar el contexto dado una palabra [5].

Mikolov *et al. [6]* define particularmente Skipgram como un algoritmo cuyo objetivo es encontrar las palabras cuyas representaciones podrían ser útiles para predecir las palabras que la rodean.

Cuando se habla de predecir se sobre entiende que es buscar la representación vectorial de una palabra.

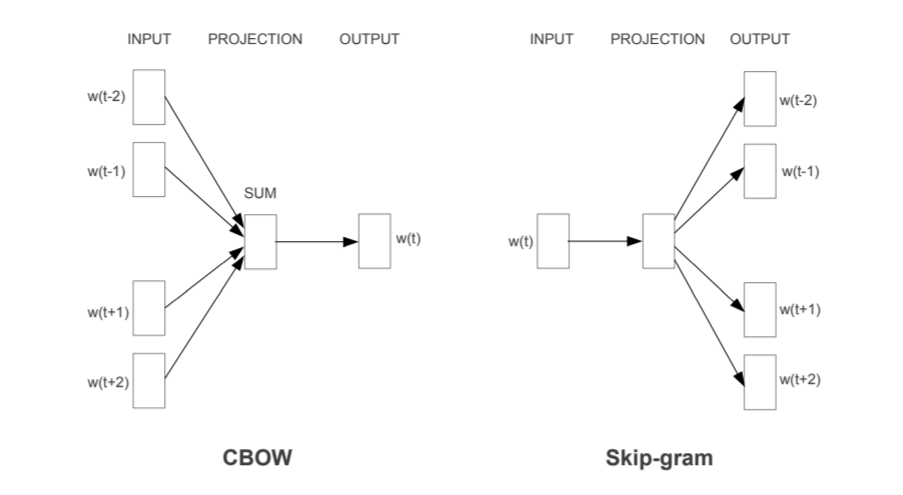
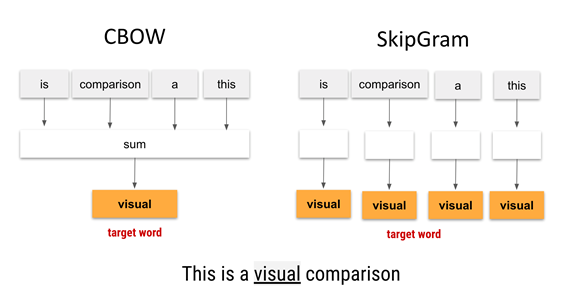


Diagrama simple de ambos algoritmos Word2Vec



Ejemplo de lo que usaría cada algoritmo para representar la palabra “visual”

**Entrenando modelo propio**

La idea principal de este proyecto iba a ser dedicarle el mayor tiempo a la implementación del detector, por lo que lo ideal iba a ser buscar un modelo ya entrenado con un vocabulario muy amplio de palabras y la representación vectorial de estas.

Sin embargo, quise probar entrenar un modelo con las pocas palabras que se podían obtener de los 300 documentos que tenía en mi poder, para simular como sería una verdadera implementación (en el caso que hubiese contado con muchos más trabajos prácticos para analizar).

Para esta tarea se utilizó Jupyter Notebooks.

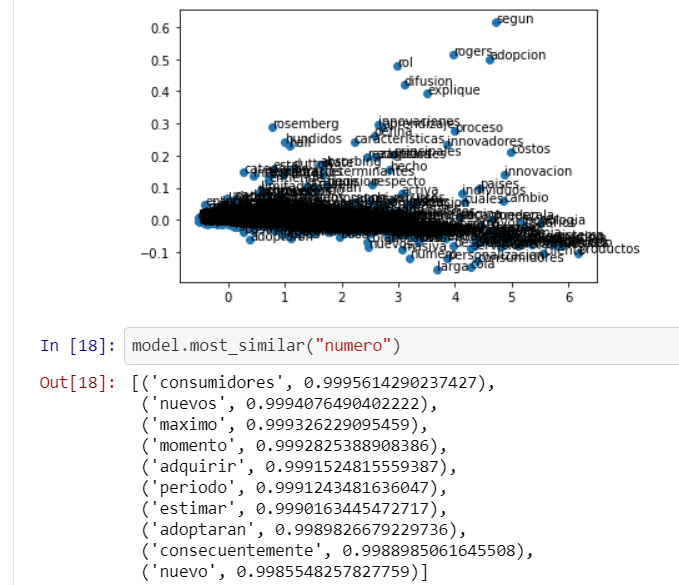
El proceso de esta fue muy simple. La idea era buscar librerías de Python que implementen toda la lógica anteriormente explicada (se utilizó el modelo CBOW).

El primer paso fue utilizar la librería PDFMiner para leer los documentos PDF que fueron dados. Luego de leerlos, se pre procesaron todos (se eliminaron stop words, números, caracteres inválidos, entre otras cosas) [7]

Una vez hecho esto, se creó un solo vector con tantos elementos como oraciones sumadas entre todos los documentos juntos. Este vector de oraciones es el que será utilizado para alimentar la red neuronal y entrenar el modelo.

Para la implementación propiamente dicha, se utilizó la librería Gensim. Para luego plottear los resultados, Matplotlib, y para traducir la representación vectorial de las palabras de un vector de 100 dimensiones a uno de 2 para verlo (lo que se conoce como Dimensionality Reduction), Sklearn [8].

El resultado de todo este proceso fue el siguiente:



El notebook entero se puede encontrar en el [repositorio](https://github.com/leoguerberg/academic_plagiarism_detector/blob/master/word2vec_training.ipynb) del trabajo.

**Desarrollo del detector de plagios**

Para la implementación del detector de plagios, se utilizó un modelo ya entrenado utilizando lo que se conoce como Fast Text, técnica muy similar al Word2Vec que fue explicado pero un poco más completo, ya que en la etapa de entrenamiento no solo utiliza la palabra y su contexto para calcular su representación vectorial, sino que también utiliza los n-gramas de esa palabra (si están en el data set) para calcular el error y recalcular su representación (proceso conocido como back-propagation). Esto lo hace un poco más lento de entrenar que el Word2Vec, pero generalmente da resultados más precisos.

Este detector es el que fue depurado y vistos los resultados, aunque también se implementó otra solución para un detector de plagios, utilizando lo que se conoce como Wordnet.

Esta no pudo terminar de ser depurada y testeada para evaluar los resultados. Esto se hizo para conocer un poco lo que se puede hacer con esta herramienta aplicado a esta tarea. Toda esta implementación, por más que no haya sido completada del todo, esta detallada en este documento.

**Modelo Fast Text**

Este fue entrenado utilizando el concepto Fast Text y el algoritmo ya explicado, Skipgram [9].

El corpus utilizado fue el Spanish Unannotated Corpora.

Algunos de los parámetros con los que se entrenó este fueron los siguientes:

* min subword-ngram = 3
  + Cantidad mínima de “sub – palabras” de la palabra principal que debe contener el data set para que se utilicen para recalcular la representación vectorial.
* max subword-ngram = 6
  + Cantidad máxima de “sub – palabras” de la palabra principal que debe contener el data set para que se utilicen para recalcular la representación vectorial.
* minCount = 5
  + Cantidad de veces mínima que debe aparecer la palabra en el training set para ser tenida en cuenta.
* dim = 300
  + Dimensión de los vectores resultado.

Para calcular y exportar las oraciones plagiadas entre un documento y otro lo que se hizo fue utilizar este modelo entrenado para obtener la representación vectorial de cada oración de estos documentos. Esta se calcula como la suma de la representación de las palabras de dicha oración dividido la cantidad.

Una vez obtenido esto, se calculó la distancia del coseno entre las mismas [10].

En el caso que la distancia esta sea menor a 0.1 (valor que, probando, encontré razonable, pero puede ser configurable) y la longitud de la oración sea mayor a 3 palabras, se considerará como plagio y se utilizara esta distancia como intervalo de confianza de la predicción.

Ej.: si la distancia entre dos vectores es de 0.04, el intervalo de confianza será de (1-0.04) \* 100 = 96%.

**Implementación personalizada usando Wordnet**

Como ya se explicó en la introducción de esta sección, el detector de plagios anterior no fue el único que se desarrolló. También intente aplicar lo que se conoce como Wordnet a esta tarea.

El resultado no fue muy bueno, además de que tampoco pudo terminar de ser probado, pero sirvió para mi verdadero objetivo, conocer los usos que se le podrían dar a esta herramienta.

Una Wordnet es un corpus gigante que contiene muchísima información sobre muchísimas palabras. Entre esta podemos encontrar, significados, sinónimos, antónimos, el uso de estas palabras en diferentes oraciones, etc.

Al ver esto, se me ocurrió una forma de usar los sinónimos [11] de una palabra para una implementación personalizada de un detector de plagios.

El procedimiento general para este método de detección entre dos documentos fue:

1. Dividir en oraciones los documentos en cuestión.

Para una mejor explicación de lo que sigue, vamos a analizar el plagio de dos oraciones solamente (obviamente, al igual que con el modelo Fast Text, se chequean todas las oraciones de cada documento entre ellas).

1. Se divide la primera oración en palabras.
2. Para la primera palabra de la oración, se buscan los sinónimos utilizando Wordnet.
3. Se chequea, para cada sinónimo, si este forma parte de la segunda oración. En caso afirmativo, se marca como una similitud entre la primera oración y la segunda.
4. Se repite este proceso para todas las palabras de la primera oración contra la segunda.
5. En el caso que la cantidad de similitudes entre una oración y otra sea mayor al 30% (configurable) de la cantidad de palabras de una de estas, se marca como plagio.

Como ejemplo, si una oración tiene 15 palabras y se chequea el plagio de esta contra otra y se encuentra una cantidad de similitudes mayor a 5 (15 \* 0.3) (véase definición de similitud en el paso 3), se marca como plagio.

**NER**

Además de la detección de plagio como tal, otro de los puntos para tener en cuenta era poder encontrar el nombre del creador del documento al que se le están analizando los plagios.

Para esto se usó la funcionalidad que ofrece la librería NLTK para hacer NER.

NER (Named Entity Recognition) consiste básicamente en “etiquetar” las palabras de una oración según diferentes categorías en un contexto en particular, como podrían ser sustantivo, verbo, adjetivo, o aún más puntual si se quiere, organización, persona, país, etc.

El uso que se le dio a esta herramienta para el TP fue muy simple e intuitivo.

Primero se etiquetó todas las palabras del nombre del documento (algunos alumnos pusieron su nombre en este). En el caso que alguna de estas se etiquete como PERSON, se supone que este será el nombre del dueño [12].

Caso que no se encuentre ninguno, se pasa a buscar en todo el documento de arriba hacia abajo (para darle prioridad a la carátula).

Casos en los que esto puede fallar sería, por ejemplo, si un documento tiene en la carátula el nombre de su profesor (se podría detectar el nombre del profesor como el dueño del documento). Se podría aplicar un poco de heurística para resolver estos casos aislados.

**Clasificación**

Otro de los puntos para tener en cuenta era intentar clasificar el tópico del documento en cuestión.

Para esta tarea se analizaron muchas posibilidades. Entre ellas estaba la de buscar un data set con diferentes documentos y oraciones ya clasificadas para entrenar un modelo utilizando una regresión logística, un SVM o una red neuronal profunda.

Al presentarse dificultades para encontrar este data set, pase a buscar modelos ya entrenados con tópicos que puedan servirme según los documentos que quiero clasificar. Luego de mucho buscar, encontré una API llamada Dandelion, donde se le enviaba una petición con una oración (entre otros parámetros) y esta te la clasificaba según distintas categorías que encajaban bastante bien con lo que buscaba [13]. Estas eran:

* Economía
* Tecnología
* Salud
* Ciencia
* Negocios
* Política
* Entretenimiento
* Deporte

Para cada predicción que hacia esta API se le daba un puntaje para cada categoría.

Esto fue muy útil, ya que implemente un algoritmo que envié a través del método GET 5 oraciones del documento a clasificar a esta API y vaya acumulando en un vector los puntajes que se le van dando a cada categoría para cada una de estas.

El vector tiene tantos elementos como categorías, en este caso 8. Cada elemento es un acumulador de puntajes como ya fue explicado.

Al finalizar, se toma el mayor de estos 8 elementos y se convierte la posición de este puntaje a la categoría correspondiente. Esta es elegida como la categoría del documento.

Ejemplo, si el mayor número del vector está en la posición 3, el documento será categorizado como Salud.

Aclaración: como esta API está en inglés, se utilizó una API de Google que se le envía un texto en español y lo devuelve en inglés.

**Scraping**

Para esta última sección técnica, se va a detallar un poco el scraping de la web que se tuvo que hacer para, no solo chequear plagios contra los documentos que uno ya tiene en el local, sino que también chequear contra documentos online.

Scraping consiste en hacer peticiones web a una página en particular y utilizar la respuesta (que no siempre es un lindo JSON, sino que podría ser HTML o JavaScript puro) para obtener cierta información.

Como en este caso la respuesta estaba en formato HTML, se utilizó la librería Beautiful Soup para el parseo y la obtención de lo necesario.

Lo que se hizo para obtener documentos relacionados al documento al cual se está chequeando plagios fue primero categorizar el documento como ya vimos en la sección anterior. Una vez hecho esto, se usó BS para hacer la búsqueda “Trabajo Practico **\*TOPICO DEL DOCUMENTO\***” en Google y traer los primeros 25 (configurable) resultados de esta.

También usando Beautiful Soup, se parseo la respuesta de Google para quedarse solamente con los links y no con toda la página en formato HTML [14].

Ya con las URL de los resultados a trabajos prácticos sobre el tema en cuestión, se filtró solamente por las que terminen en “.pdf” o “.docx”. Esto porque, para aprovechar el detector ya desarrollado, se buscará descargar directamente documentos en estos formatos y chequear plagios de la misma forma que entre documentos que no vienen de la web.

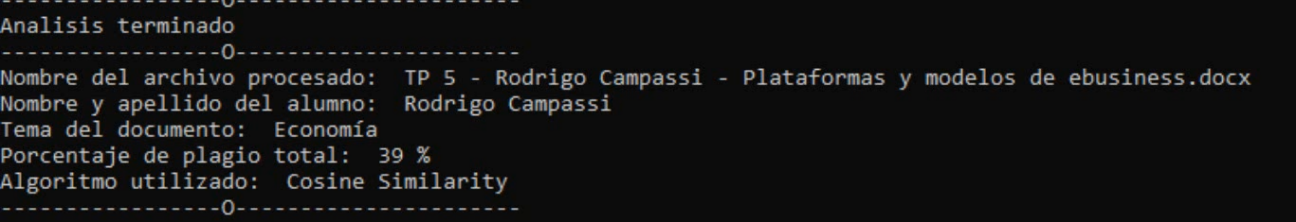
Una vez descargados y guardados en una subcarpeta llamada “Scraped” (esta se limpia completamente al hacer un análisis nuevo), se eliminan aquellos que pesen menos de 50kb, ya que, durante el desarrollo de esto, se vio que muchos de los archivos descargados de la web en estos formatos que pesaban excesivamente poco estaban corruptos.

Una vez terminado todo este proceso, ya solo se necesita pasar estos documentos descargados por el mismo detector que pasan los documentos que ya estaban de antes.

**Resultados**

Para evaluar los resultados de todo el proceso explicado, se analizaron los plagios de un documento contra los 300 documentos que teníamos en nuestro poder.

La información obtenida del documento que se esta analizando fue la siguiente:



Dentro de las oraciones que dio como plagiadas (el 39% de las oraciones totales del documento), se pueden encontrar las siguientes:

|  |  |  |  |
| --- | --- | --- | --- |
| **Doc. Plagiado** | **Oración Plagiada** | **Oración Chequeada** | **Predicción** |
| TP 2 Larga Cola - Ezequiel Ogando .docx | por otro lado, también facilita la búsqueda filtro de productos a la vez que actúa como una red de recomendaciones comentarios logrando que los consumidores tengan más opciones de elección puedan descubrir artículos según sus necesidades. | en lo que nos respecta de marketing los sitios web que le recomiendan artículos que podrían gustarle con base en compras anteriores utilizan el machine learning para analizar su historial de compras promocionar otros artículos que podrían interesarle. | 90.05% |
| PREGUNTAS TP Machine, Platform, Crowd.docx | ahora los productos se venden a través de plataformas digitales a su vez se utilizan las plataformas para ofrecer servicios experiencias diferenciales por utilizar la plataforma. | son plataformas que no ofrecen cosas digitales como música software o servicios financieros, sino que incluye bienes servicios físicos a través de una plataforma online. | 90.29% |
| Marketing en Internet y Nueva Economía - TP0.docx | fuente tercer definición se entiende como economía digital al conjunto de estrategias acciones para contribuir al impulso de la competitividad del sector productivo de una organización mediante su inserción en el mundo digital es decir migrar a una economía donde se generaliza el uso de las tecnologías de la información en todas las actividades económicas culturales sociales. | comunidades de consumo Facebook el conocimiento de las masas la tecnología aplicada a la información permite dirigir la publicidad a un siguiente nivel. | 92.02% |
| Marketing en Internet y Nueva Economía - TP0.docx | su desarrollo supone una mejora en el bienestar de los ciudadanos ya que la economía digital busca automatizar los procesos a través de internet desde el proceso de venta hasta el mantenimiento sin necesidad de que el cliente se desplace a diferentes puntos físicos para adquirir un determinado producto o servicio ahorrando en costo tiempo. | a su vez este avance abarata el costo de los productos la tecnología al estar al alcance de todos permite descentralizar el conocimiento. | 91.6% |
| MKTG\_TP2 - Suchecki - Marketing 4.0.pdf | aprovecha la conectividad de maquina a máquina la inteligencia artificial para mejorar la productividad de marketing mientras aprovecha la conectividad de persona a persona para fortalecer el compromiso del cliente. | como transforma a la inteligencia humana los productos al conocimiento centralizado Core el incremento rápido expansivo de las capacidades de las maquinas se contrapone a la inteligencia humana permite obtener mucha información para la toma de decisiones. | 90.03% |
| MKTG\_TP2 - Suchecki - Marketing 4.0.pdf | en la economía digital los clientes ahora cuentan con la capacidad para evaluar e incluso analizar la promesa de posicionamiento de marca de cualquier empresa. | esta capacidad de capturar datos analizarlos usarlos para personalizar una experiencia de compra o implementar una campaña de marketing es el futuro del comercio detallista. | 90.84% |
| PREGUNTAS TP Machine, Platform, Crowd Gabriela Gonzalez.docx | las tres tendencias son incremento rápido expansivo de las capacidades de las maquinas la inteligencia artificial se contrapone a la inteligencia humana. | como transforma a la inteligencia humana los productos al conocimiento centralizado Core el incremento rápido expansivo de las capacidades de las maquinas se contrapone a la inteligencia humana permite obtener mucha información para la toma de decisiones. | 93.98% |
| Marketing - TP 1.docx | en los mercados de larga cola la regla cambia en lo siguiente se pueden ofrecer más productos hay una mejor distribución de las ventas entre los artículos más populares los de nicho en esto influyen las recomendaciones las críticas se pueden obtener beneficios tanto de productos populares como de productos de nicho por que las tiendas online que basan su estrategia en larga cola igualmente necesitan ofrecer hits a sus clientes las tiendas online que basan su estrategia en larga cola necesitan ofrecer hits porque esos productos son los que atraen al público a la tienda para que luego el mismo descubra otros productos de nicho. | en lo que nos respecta de marketing los sitios web que le recomiendan artículos que podrían gustarle con base en compras anteriores utilizan el machine learning para analizar su historial de compras promocionar otros artículos que podrían interesarle. | 91.27% |
| Marketing - TP 0.docx | email resolución brindar tres definiciones de economía digital conjunto de estrategias acciones para contribuir al impulso de la competitividad del sector productivo de una organización mediante su inserción en el mundo digital es decir migrar a una economía donde se generaliza el uso de las tecnologías de la información en todas las actividades económicas culturales sociales. | comunidades de consumo Facebook el conocimiento de las masas la tecnología aplicada a la información permite dirigir la publicidad a un siguiente nivel. | 92.32% |

Si se quieren ver los resultados completos están acá:



**Fuentes**

1. Ferrero, Jérémy & Agnès, Frédéric & Besacier, Laurent & Schwab, Didier. (2017). Using Word Embedding for Cross-Language Plagiarism Detection. 10.18653/v1/E17-2066.
2. Suleiman, D., Awajan, A., & Al-Madi, N. (2017, October). Deep Learning Based Technique for Plagiarism Detection in Arabic Texts. 2017 International Conference on New Trends in Computing Sciences (ICTCS). 10.1109/ictcs.2017.42.
3. <https://stackoverflow.com/questions/44786888/will-word2vec-be-more-efficient-in-text-based-plagiarism-detection-than-wordnet>
4. <https://machinelearningmastery.com/why-one-hot-encode-data-in-machine-learning/>
5. <https://stackoverflow.com/questions/38287772/cbow-v-s-skip-gram-why-invert-context-and-target-words>
6. Mikolov, Tomas & Sutskever, Ilya & Chen, Kai & Corrado, G.s & Dean, Jeffrey. (2013). Distributed Representations of Words and Phrases and their Compositionality. Advances in Neural Information Processing Systems. 26.
7. Pre-Procesado
   1. <https://es.stackoverflow.com/questions/135707/c%C3%B3mo-puedo-reemplazar-las-letras-con-tildes-por-las-mismas-sin-tilde-pero-no-l>
   2. <https://stackoverflow.com/questions/5541745/get-rid-of-stopwords-and-punctuation>
8. <https://machinelearningmastery.com/develop-word-embeddings-python-gensim/>
9. <https://github.com/dccuchile/spanish-word-embeddings#fasttext-embeddings-from-suc>
10. <https://radimrehurek.com/gensim/models/keyedvectors.html>
11. <https://pythonprogramming.net/wordnet-nltk-tutorial/>
12. <https://stackoverflow.com/questions/20290870/improving-the-extraction-of-human-names-with-nltk>
13. <https://dandelion.eu/docs/api/datatxt/cl/v1/>
14. <https://medium.com/@ajaypanthagani/heres-how-you-can-scrape-google-search-results-with-python-fa45d09a95a2>