Universidad Autónoma de Baja California

Facultad de Ingeniería, Arquitectura y Diseño

Proyecto Final

Análisis de sentimientos

**Mauricio Santiago Valdovinos Morales**

Materia: Cómputo Evolutivo

Profesor: Hugo Armando Guillen Ramírez

Ensenada, Baja California a; 5 de Diciembre del 2018.

Introducción

En la actualidad la inteligencia artificial es uno de los temas que más atención está captando en el mundo tecnológico y en el de los negocios. El análisis de sentimientos es una de las áreas que están causando gran impacto actualmente a pesar que es un tema relativamente viejo, ya que se estudia desde varios años atrás. Sin embargo en el área del idioma español no está muy explorado o desarrollado este tema, ya que existen muchas implementaciones del análisis de sentimientos principalmente en idioma inglés.

La razón por la que elegí este proyecto fue por querer retomar el mismo proyecto del cuso de inteligencia artificial pero ahora realizando un análisis a textos en idioma español clasificando opiniones de películas[1] extraídos de una página llamada MuchoCIne y TripAdvisor, clasificándolos por tres categorías: positivo, negativo y neutral. Utilizando métodos de procesamiento de lenguaje natural y un modelo de una red neuronal llamado Word2Vec programados en el lenguaje Python.

Procesamiento de lenguaje natural

El **procesamiento del lenguaje natural** (PLN) es el campo que combina las tecnologías de la ciencia computacional (como la inteligencia artificial, el aprendizaje automático o la inferencia estadística) con la lingüística aplicada, con el objetivo de hacer posible la comprensión y el procesamiento asistidos por ordenador de información expresada en lenguaje humano para determinadas tareas, como la traducción automática, los sistemas de diálogo interactivos, el análisis de opiniones, etc. [5]

Objetivo

Clasificar documentos de opinión de películas en idioma español en tres clases: positivo, negativo y neutral. Utilizando estrategias de procesamiento de lenguaje natural y la red neuronal Word2Vec para lograrlo.

Metodología

Para poder cumplir el objetivo de este proyecto se utilizaron varias técnicas pero la que mejor resulto fue utilizando una medida de distancias entre documentos. Esta distancia forma parte para la toma de decisión del clasificador utilizado llamado K-vecinos cercanos (K-Neirest Neighbors), este clasificador determina según el documento que se le otorgaotorga la clase a la que pertenece.

El proceso es posible utilizando cuatro pasos:

1. Preparación de los datos
2. Creación de embebidos
3. Búsqueda de vecinos
4. Clasificación

1.- Preparación de los datos

Utilizando un corpus de 3878 críticas de cine el primer paso que se necesita es concatenar los documentos en una sola lista. En este caso se utilizó una lista de diccionarios. Donde el diccionario almacenaba la crítica de cine llamado ‘review’, otro campo llamado ‘sentiment’ que según el número de estrellas con el que estaba calificada la película se determinaba la clase de la crítica utilizando las siguientes métricas: de una a dos estrellas era negativa, 3 estrellas era una crítica neutra y de 4 a 5 la crítica se calificaba como positiva.

Para poder analizar un texto este debe poder ser dividido entre sus oraciones o incluso entre palabras, en este caso cada critica se dividió en sus palabras y se realizó un filtrado de las llamadas *stop words* que estas son palabras que no aportan ningún tipo de valor a nuestro análisis. Esta review ya tokenizada y filtrada se guardó en un campo del diccionario llamado ‘review\_clean’.

De esta manera ya se tiene toda la informacion necesaria de cada critica en un diccionario y concatenados en una lista de 1x3878. Con el campo ‘review\_clean’ es con el que podremos entrenar nuestro modelo.

2.- Creación de embebidos

La creación de embebidos es prácticamente la transformación de palabras a vectores de números para que una computadora pueda entenderlos. Con estos vectores se puede realizar tareas como la búsqueda de similitudes entre palabras como se muestra en la figura (1).

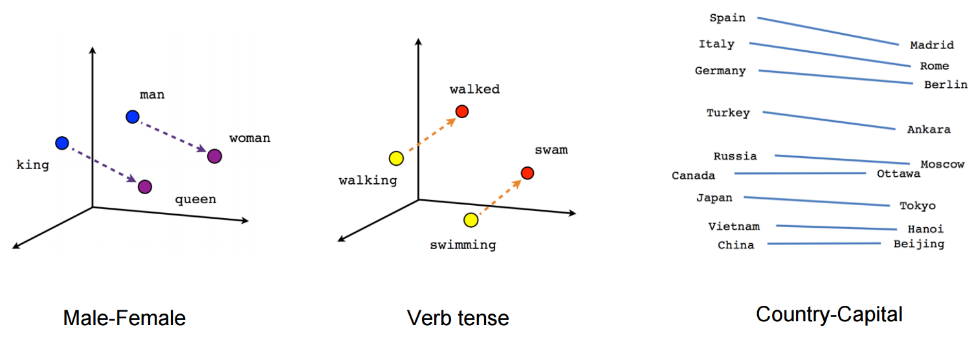


Figura (1). Ejemplos de similitud de palabras en inglés el espacio de embebidos.

En este proyecto se utilizó un modelo llamado skip gram[3]. Este modelo lo que hace es analizar las palabras de cada sentencia u oración y trata de usar cada palabra para predecir cuales serán sus palabras vecinas. Utilizando como datos de entrada un valor llamado *window* que este valor determina el número de palabras objetivo alrededor de la palabra contexto (mirar figura (3)) y un vector llamado *one-hot* este vector contiene puros unos y ceros y es de una dimensión de 1xN donde N es el número de palabras en el vocabulario. El vector one hot pone un uno en la palabra que se va a analizar y un cero en el resto de palabras. En la figura (2) se puede observar como el modelo recibe una palabra contexto y entrega N palabras objetivo que son los posibles vecinos de la palabra de entrada.

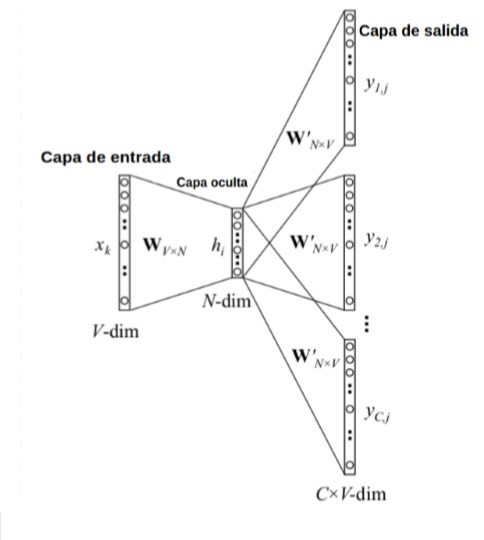


Figura (2). Diagrama del modelo Skip-Gram[1]

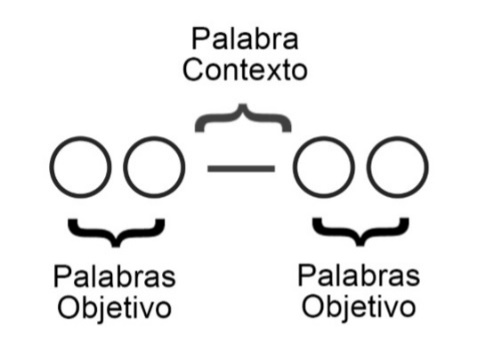


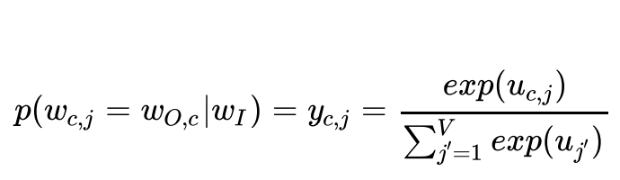
Figura (3). Palabras que ve la palabra contexto con un window = 2

La red neuronal utilizada en este proyecto se llama Word2Vec[3] y como su nombre lo indica es la conversión de una palabra a un vector, más específicamente un vector embebido. Con estos vectores se crea un espacio en el que pada palabra es un punto en él y asi podemos buscar similitudes entre palabras.

El modelo Word2Vec recibe de entrada el mismo vector que el skip gram, un vector one hot en la capa de entrada. En la capa oculta crea una matriz de NxM donde N es igual al número de palabras en el vocabulario y M es el número de dimensiones del espacio embebido. Y esta matriz es multiplicada por el vector one hot.

El vector resultante corresponde a la fila correspondiente al único 1 de one hot y utilizando una ecuación llamada softmax que entrega una distribución de probabilidades de la palabra de tener similitud con el resto de las palabras en el vocabulario.

La ecuación Softmax es la siguiente:



3.- Búsqueda de vecinos

Las búsquedas de vecinos cercanos se realizaron de las siguientes dos maneras:

* A nivel de palabras
* A nivel de documentos

En el espacio de embebidos cada palabra forma un ponto en él y utilizando distancia euclidiana se pueden encontrar las distancias entre las palabras y de esta manera buscar los vecinos cercanos de una palabra deseada. Esto con el sentido de que si unas palabras estas cercanas a otras significa que pueden compartir un significado, sentido y por ende se puede entender que pueden compartir un sentimiento.

De esta forma buscando los vecinos cercanos a una palabra podremos determinar a qué sentimiento tienden ese grupo y con esa informacion ayudarnos a clasificar.

Para la búsqueda de vecinos cercanos a nivel de documento se utilizó una métrica llamada *Word Movers Distance* (WMD) [4]. Esta métrica busca la similitud entre documentos entregando un valor que es la distancia. Entre más bajo es el valor más similares son los documentos y por ende pueden compartir el sentido y el sentimiento.

4.- Clasificación

El clasificador que se utilizo es uno llamado K-Vecinos cercanos (K-Nearest Neighbors)[1] este clasificador busca los k vecinos cercanos de un documento y por medio de un sistema de votación se realiza la clasificación. La clase más votada entre positivo, negativo y neutral será la clase con la que se clasificara al documento.

Paquetería necesaria para el programa

Primero que el programa funcione se debe realizar la instalación de las siguientes paqueterías: NLTK, sklearn, gensim, pickle con la instrucción en la ventana de comandos

*pip install nltk*

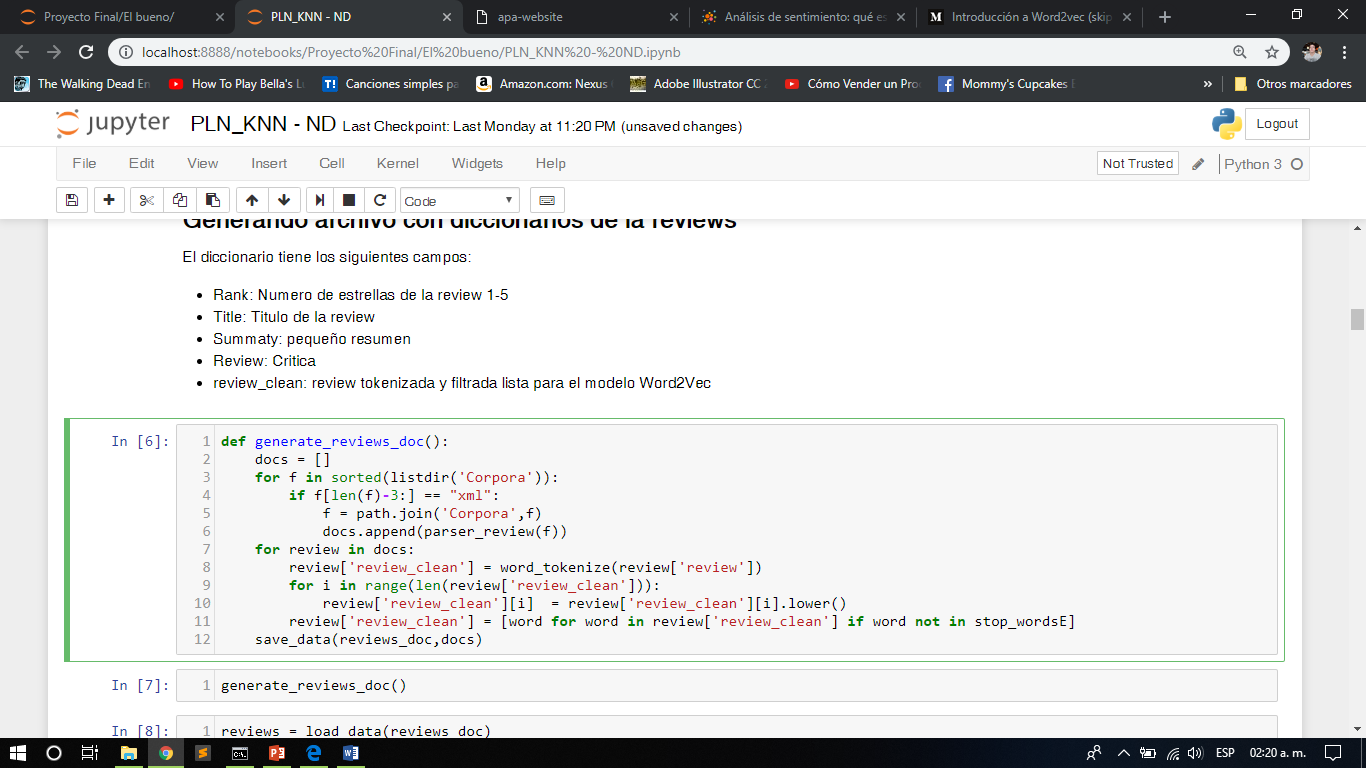
*pip install sklearn*

*pip install gensim*

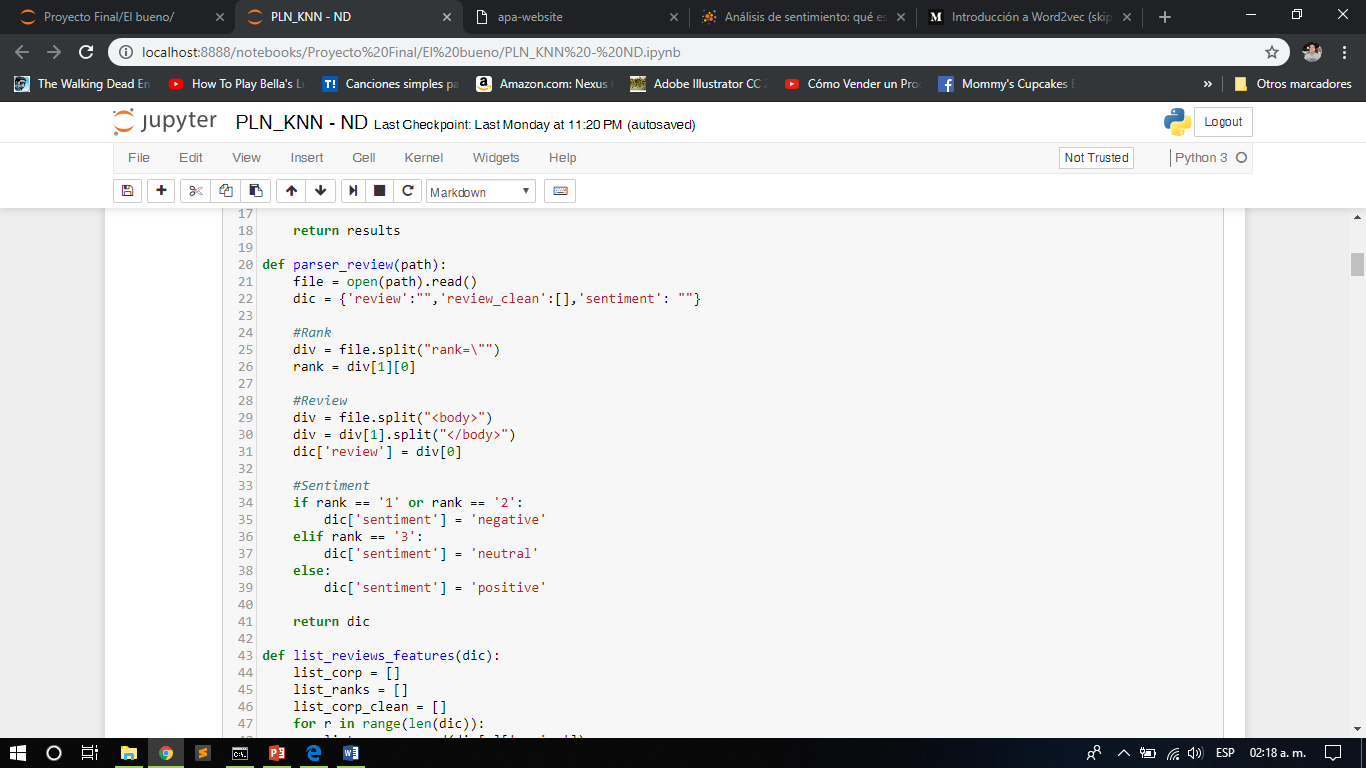
*pip install pickle*

Procedimiento del programa

1.- Primero se extrae las críticas y se preparan para ser utilizadas en el modelo Word2Vec

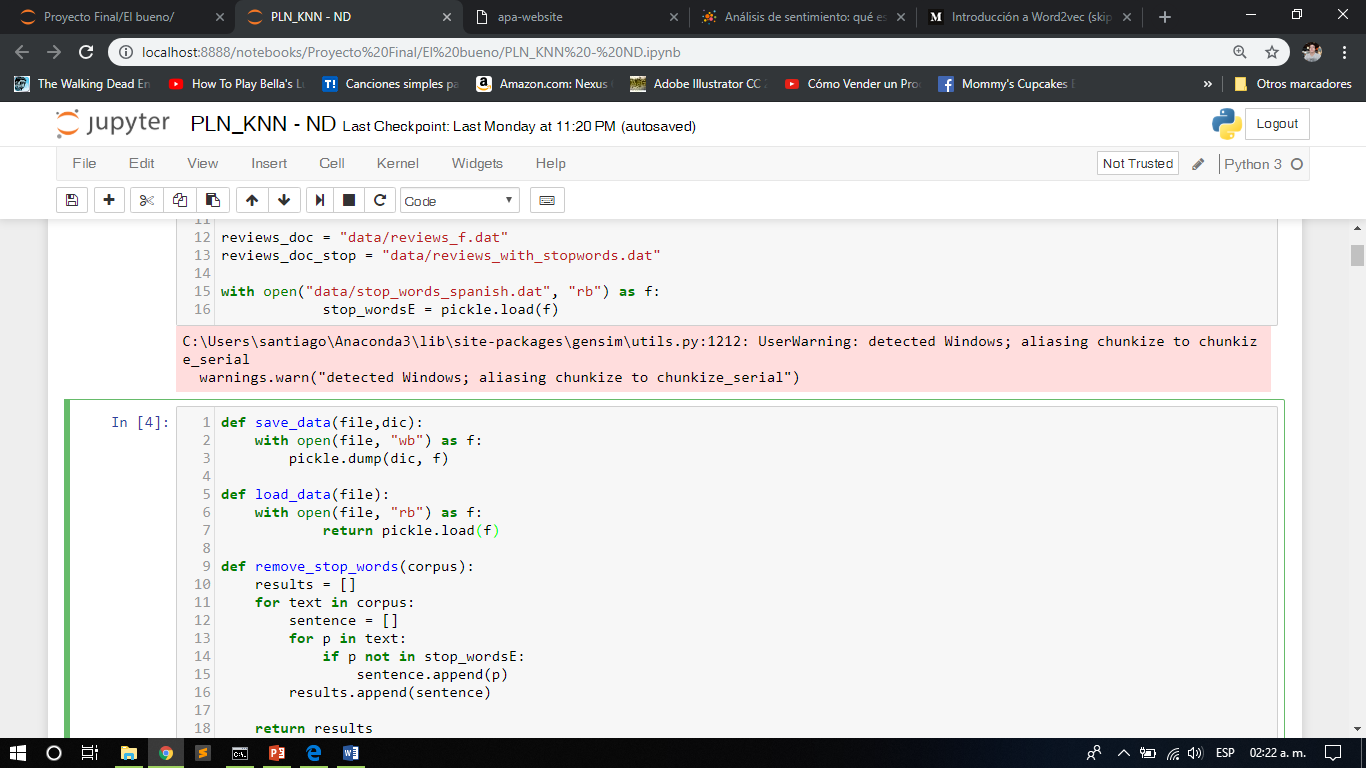


Función que extrae los documentos, los manda a parsear y les filtra las stop words



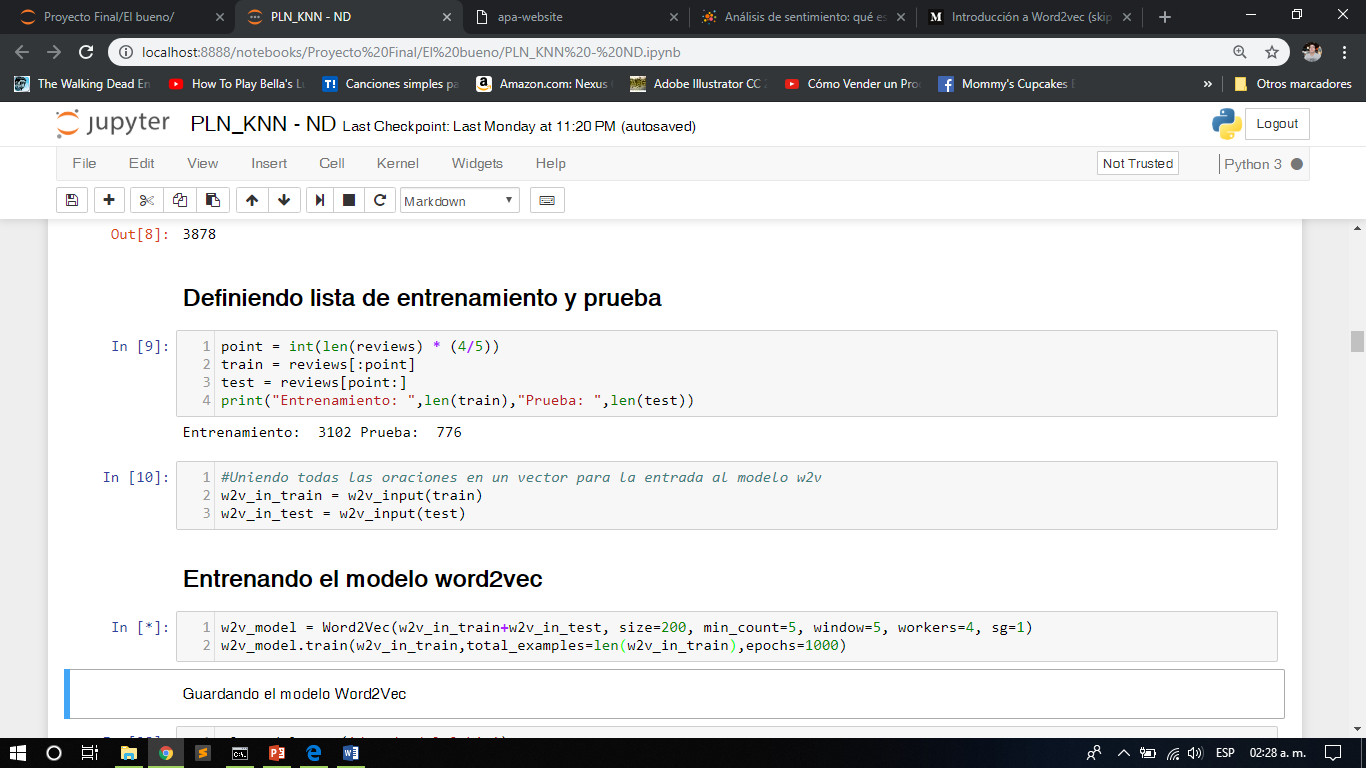
Funcion parser, que elimina las etiquetas HTML y extrae solo la informacion que se necesita.

2.- Para la agilidad en el proceso de realizar el proyecto se las criticas concatenadas se guardaron en un documento .dat para que posteriormente solo sea cargado.

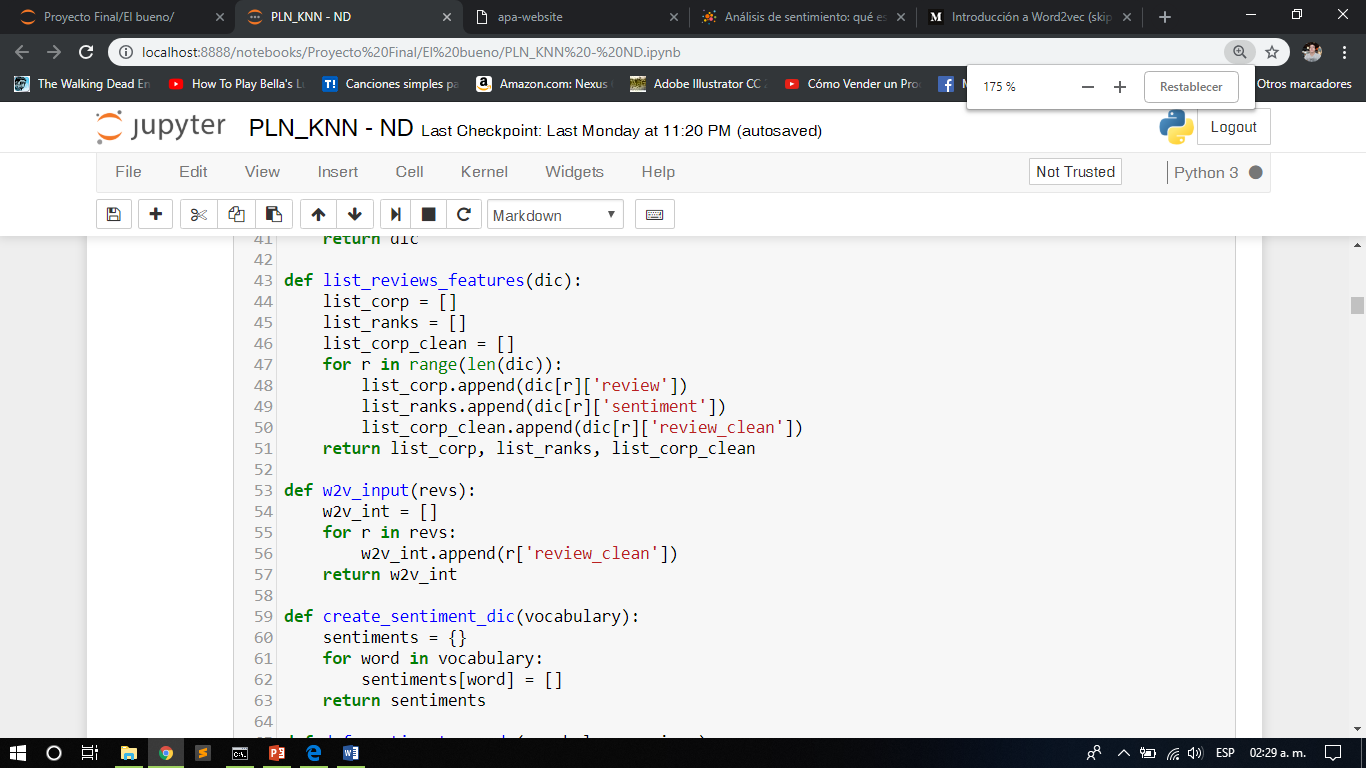


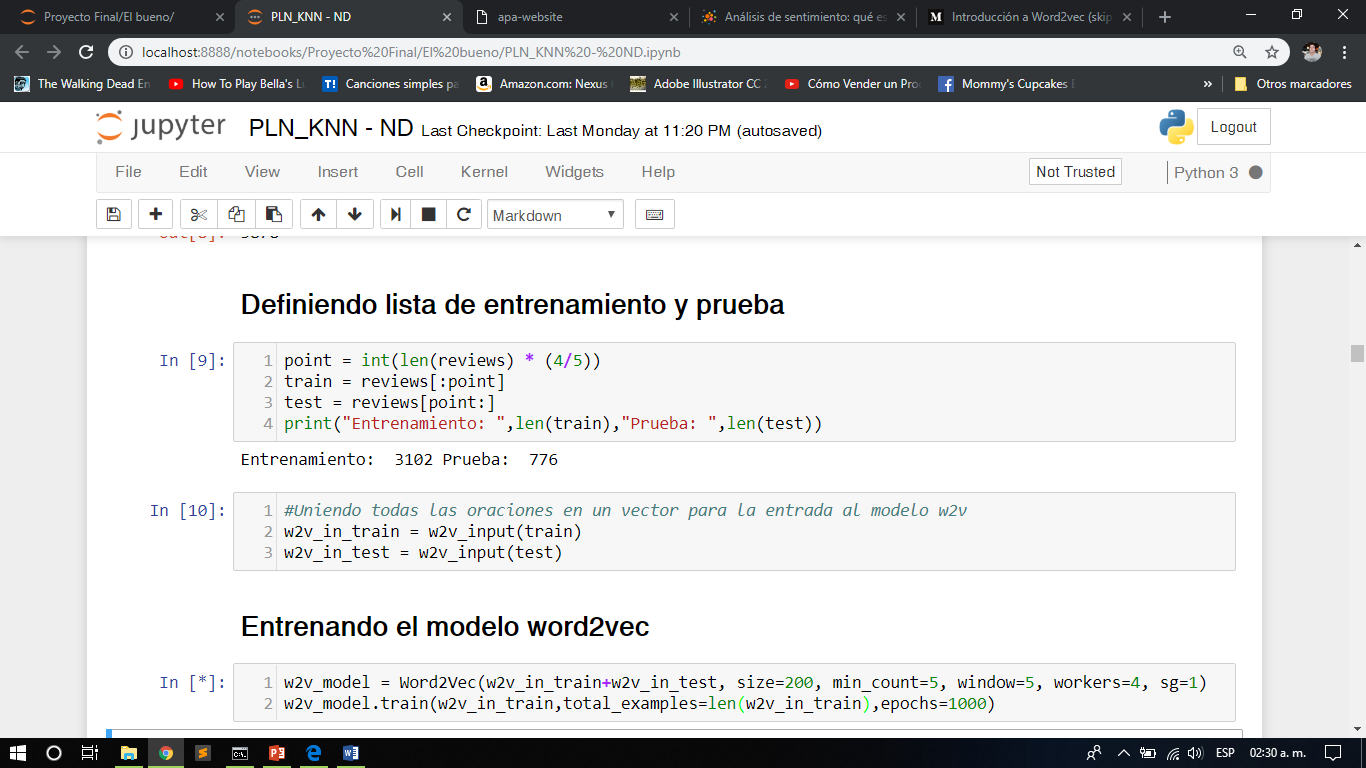
Funciones para guardar archivos y cargarlos

3.-Como paso siguiente se definen los datos de entrenamiento y de prueba. Utilizando un 80% del total de críticas del corpora para entrenar y un 20% para probar.



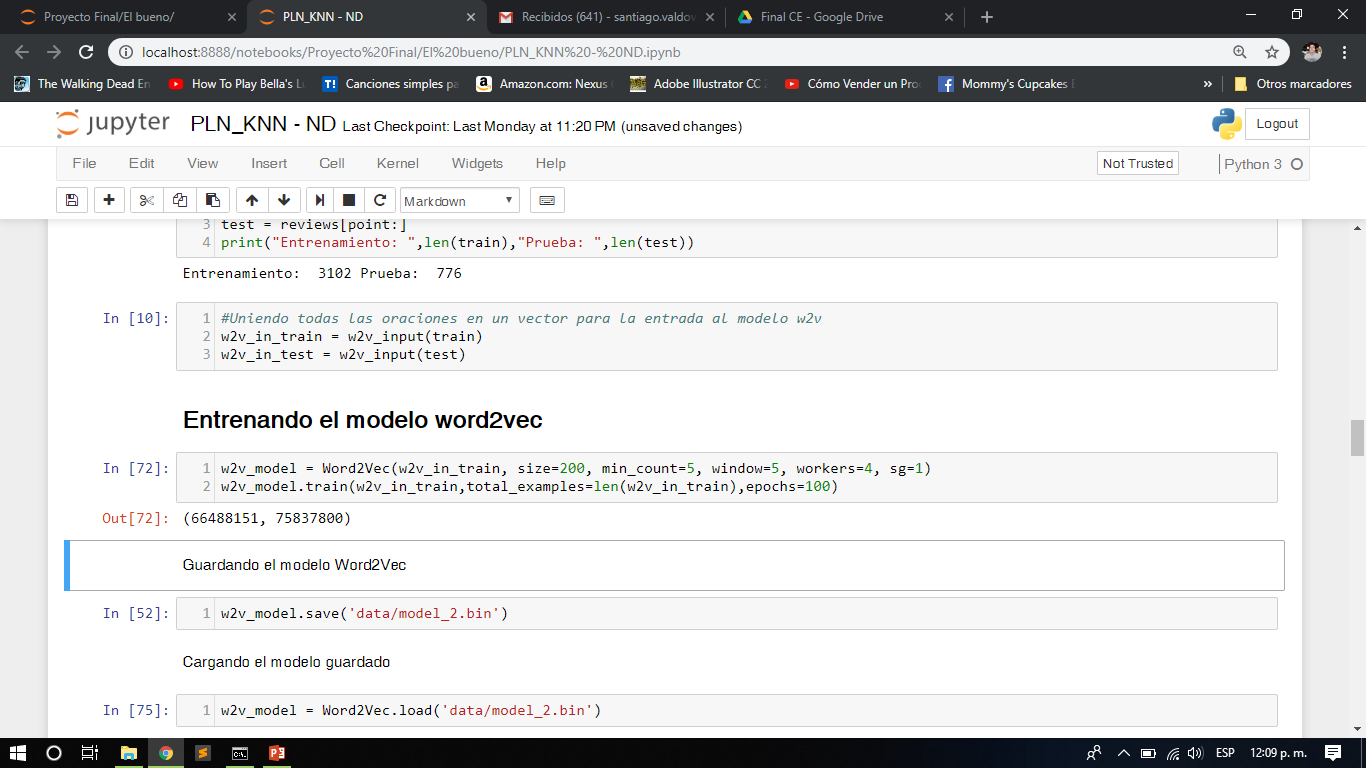
4.- Estructurando las listas que recibe como entrada el modelo Word2Vec utilizando el campo de los diccionarios ‘review\_clean’.



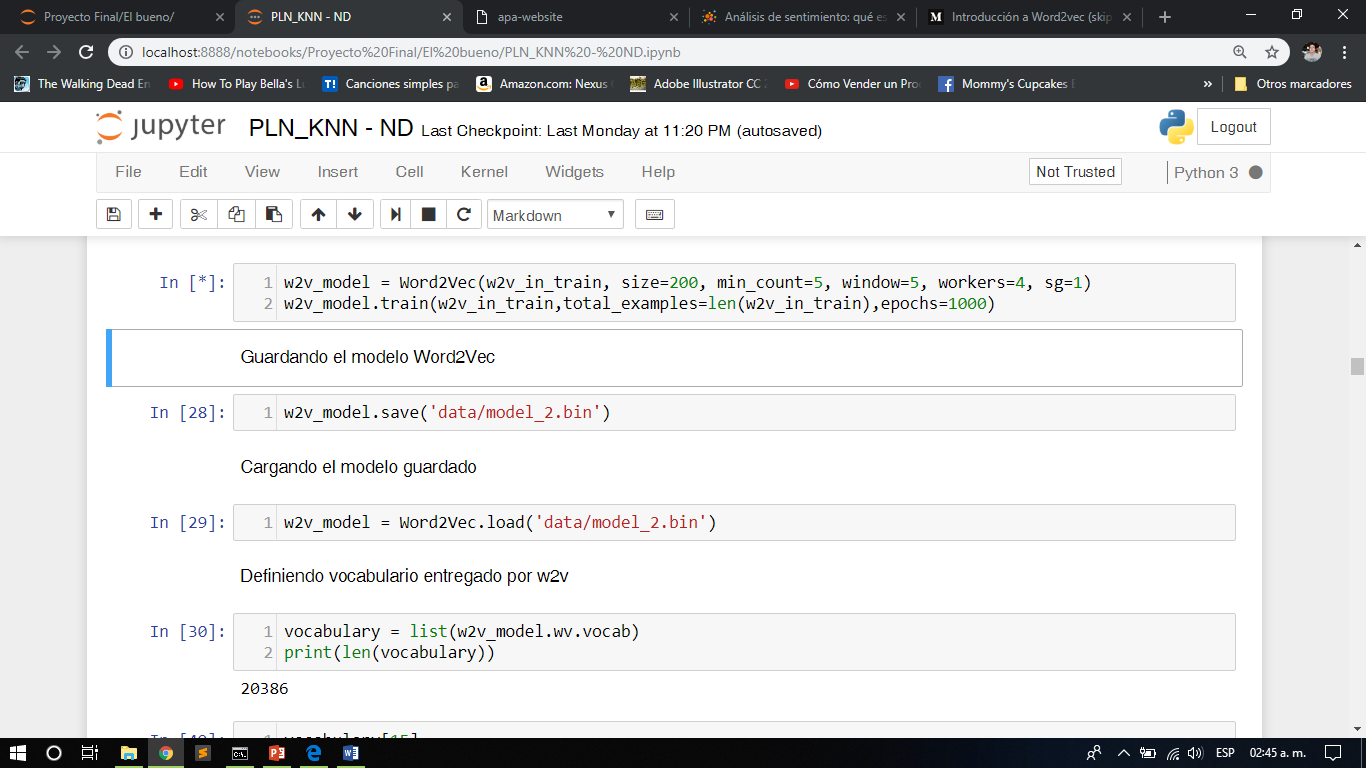


5.- Entrenar el modelo Word2Vec. Se utilizó el modelo de la paquetería gensim y se utilizaron los siguientes parámetros:

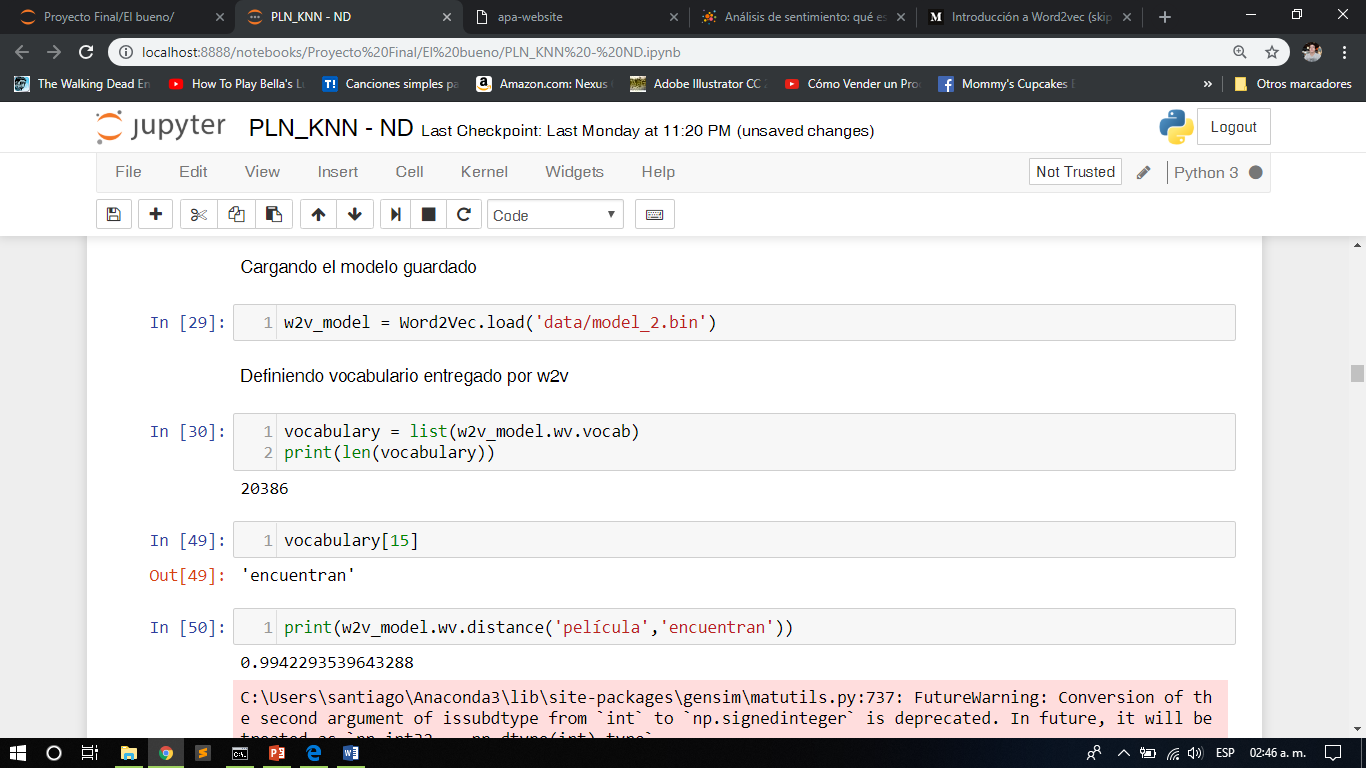
|  |  |
| --- | --- |
| size (número de dimensiones) | 200 |
| min\_count (Cuenta mínima de repetición de una palabra) | 5 |
| window | 5 |
| workers (números de hilos que usara) | 4 |
| sg (0= CBOW, 1 = Skit-gram) | 1 |
| epochs (Numero de iteraciones) | 100 |



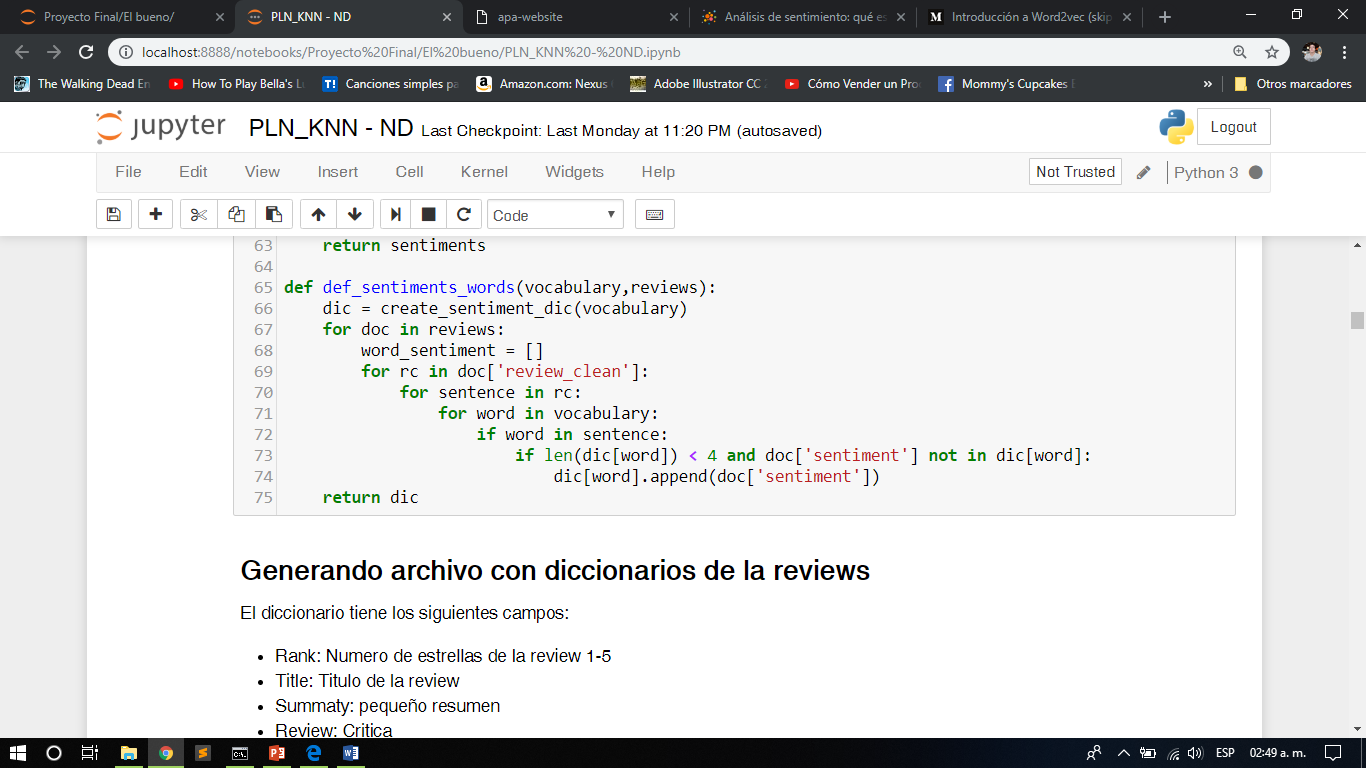
6.- Se guardó el modelo para poder utilizarlo en el futuro sin tener que volver a entrenar.

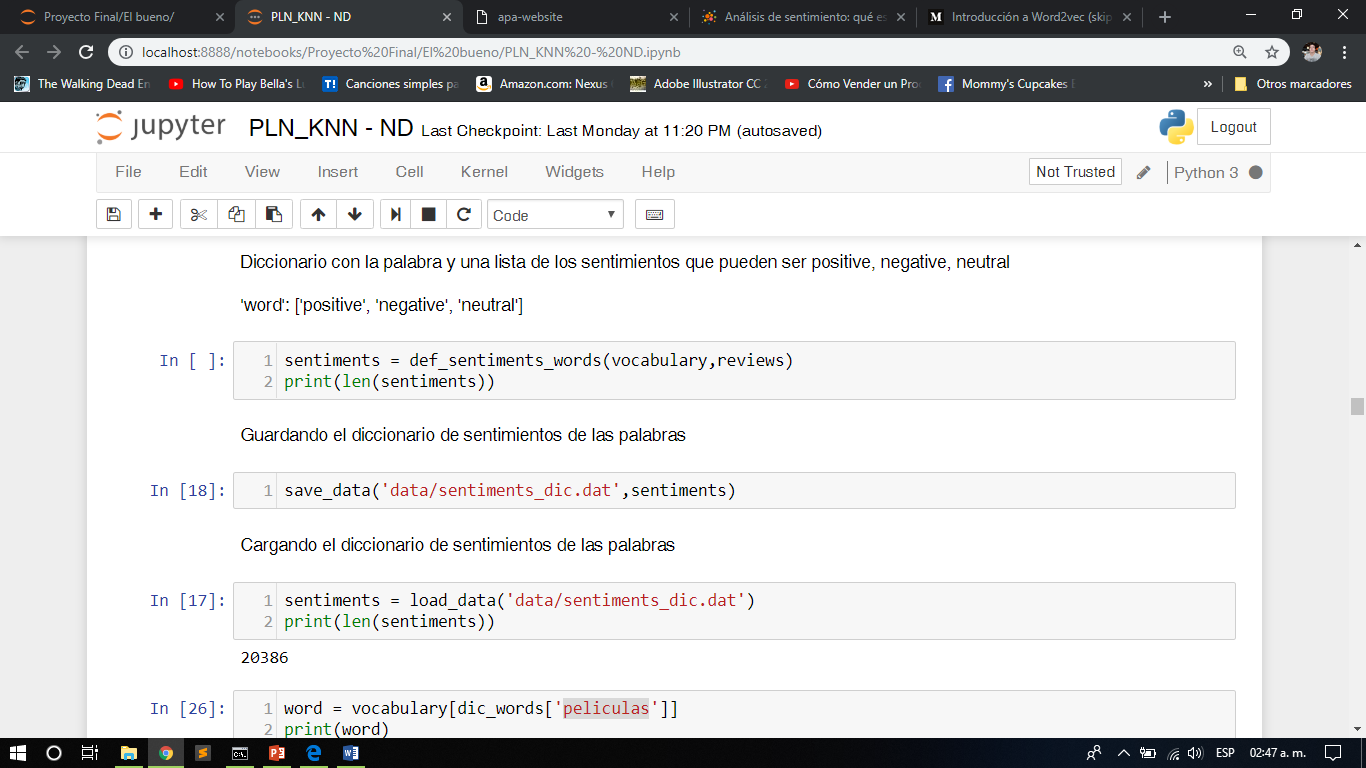


7.- Definiendo el vocabulario entregado por el wrod2vec:

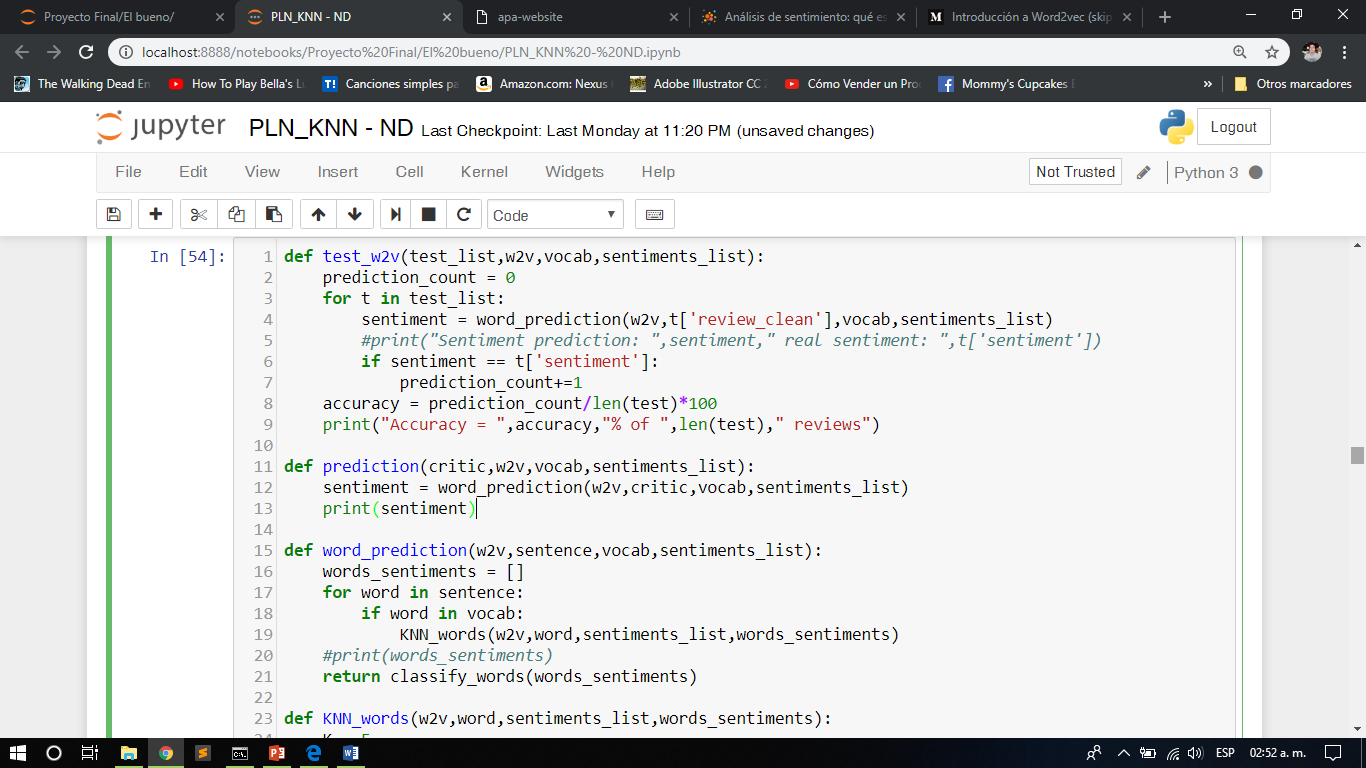


8.- Se creó un diccionario con las palabras del vocabulario y se almacenaba las clases a las que pueden pertenecer según a la critica en la que aparecen. De igual manera, se guardó el diccionario en un documento.

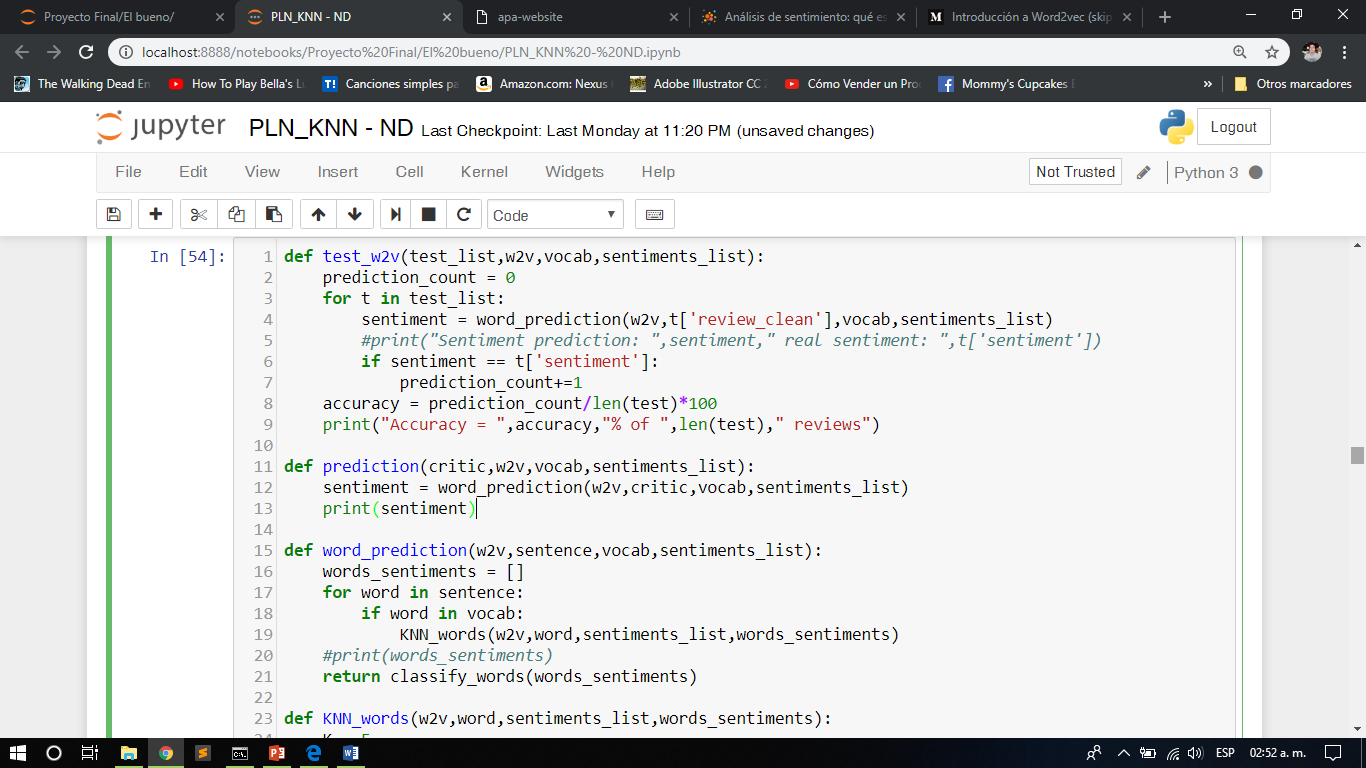




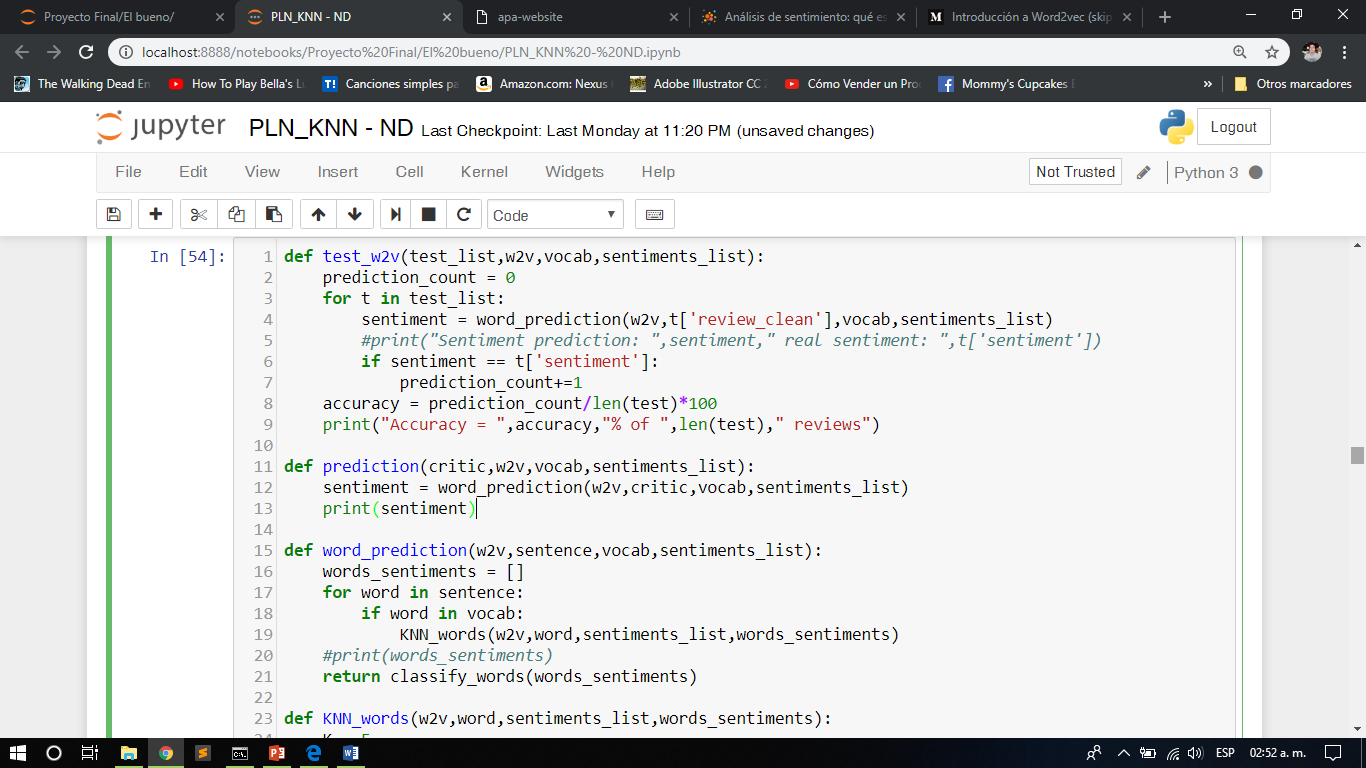
9.- Funciones para probar el modelo

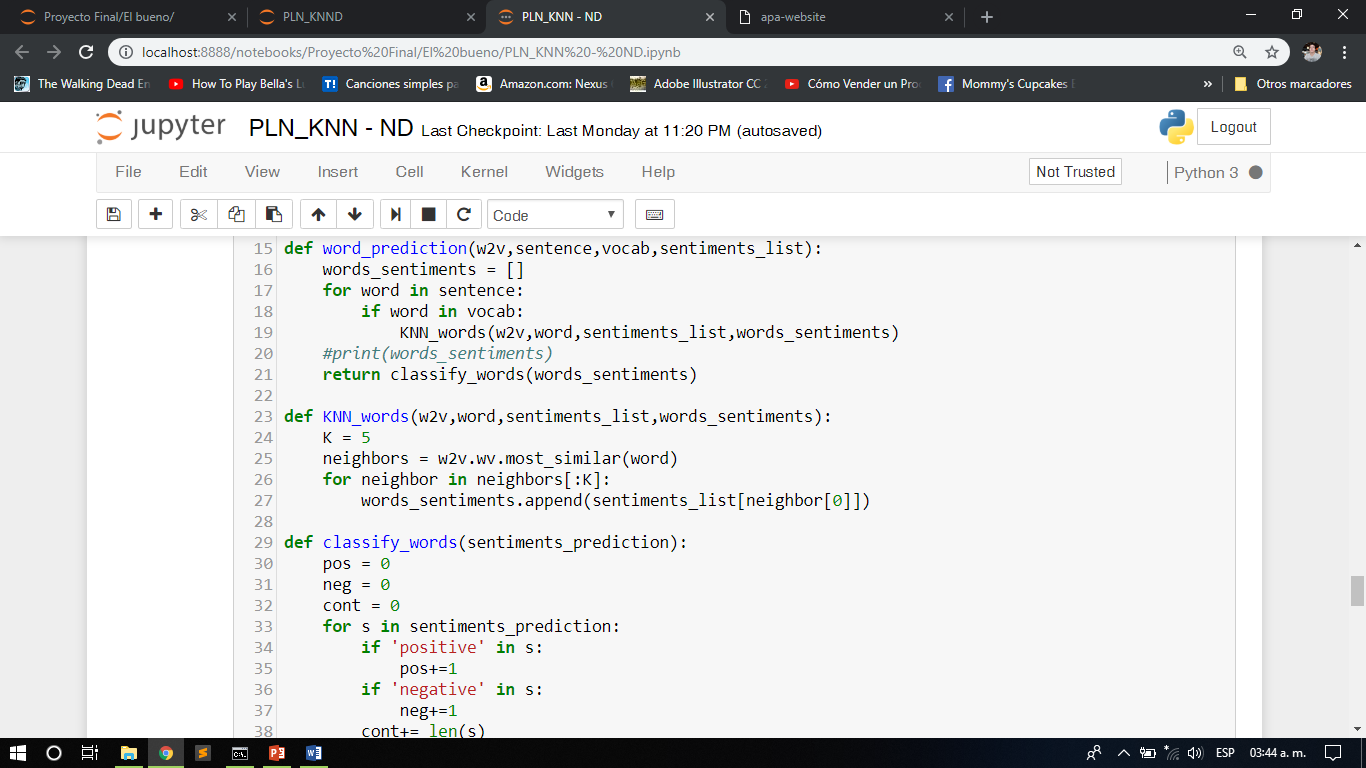


Función para determinar la exactitud de predecir del clasificador con la lista de prueba.

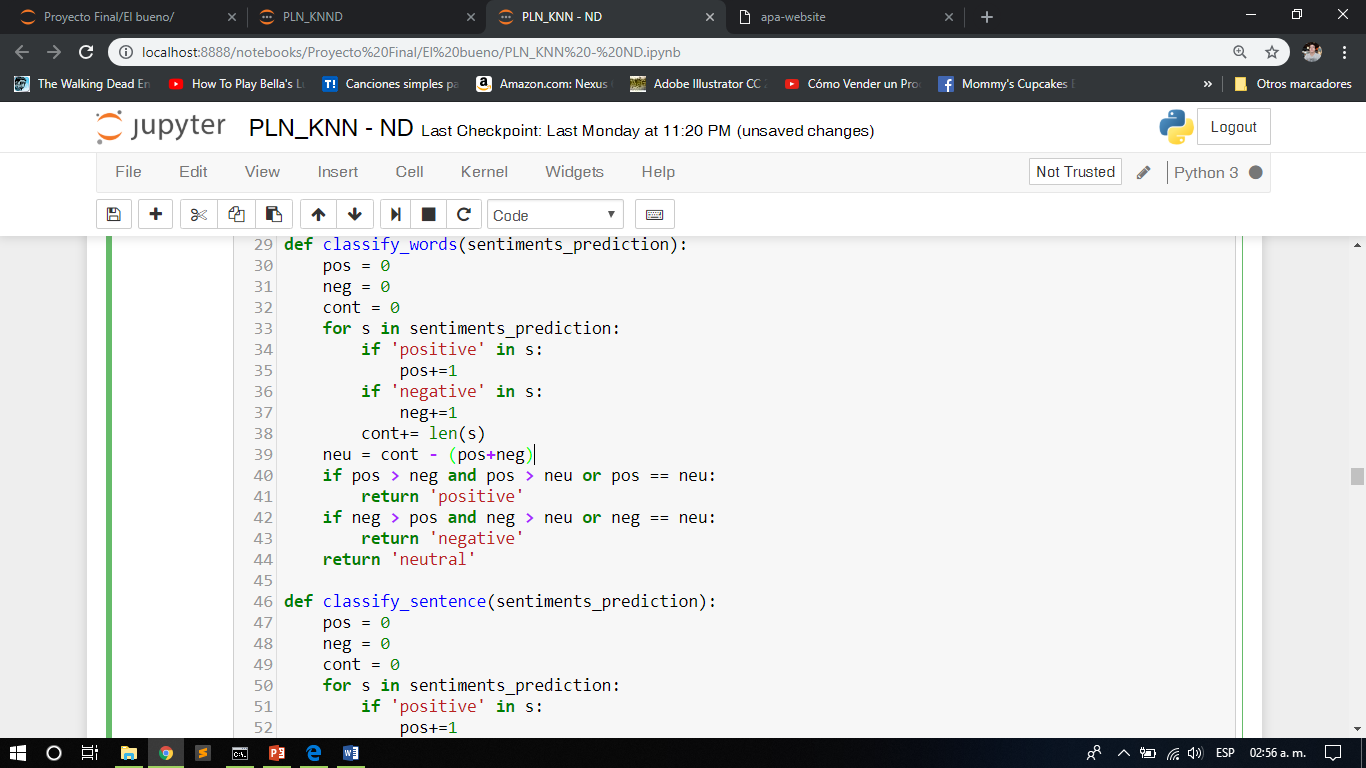


Función para predecir la clase de una sola critica.

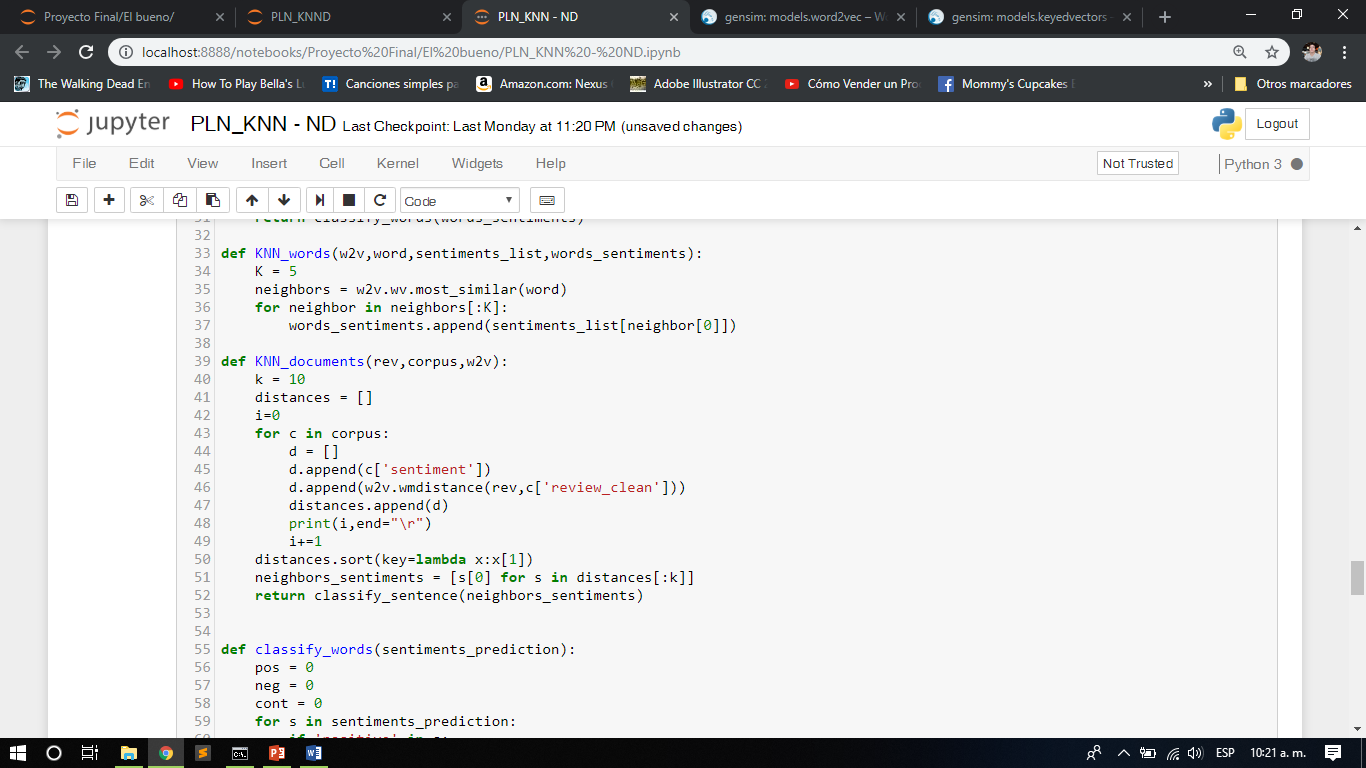




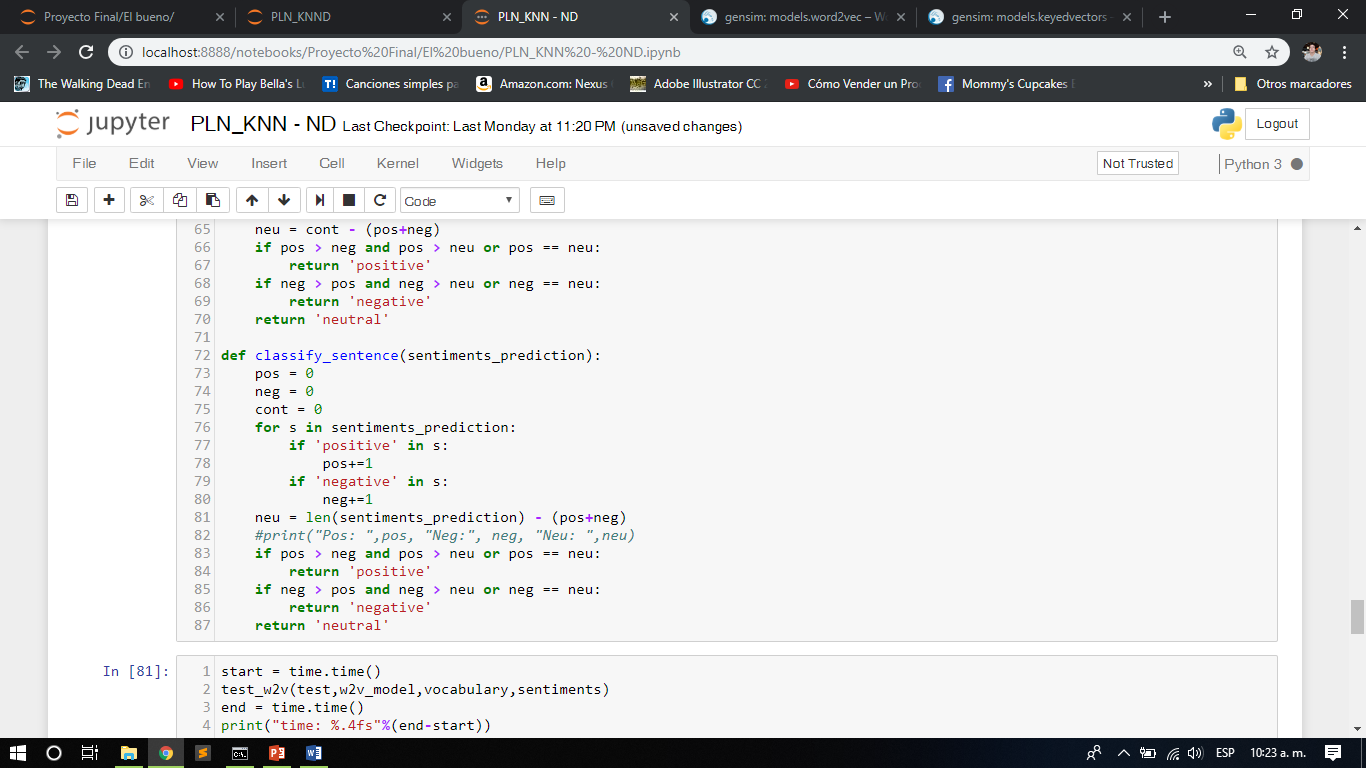
Búsqueda de los K vecinos a nivel de palabras



Votación de los sentimientos encontrados por los k vecinos a nivel palabra



Búsqueda de los k vecinos a nivel de documento

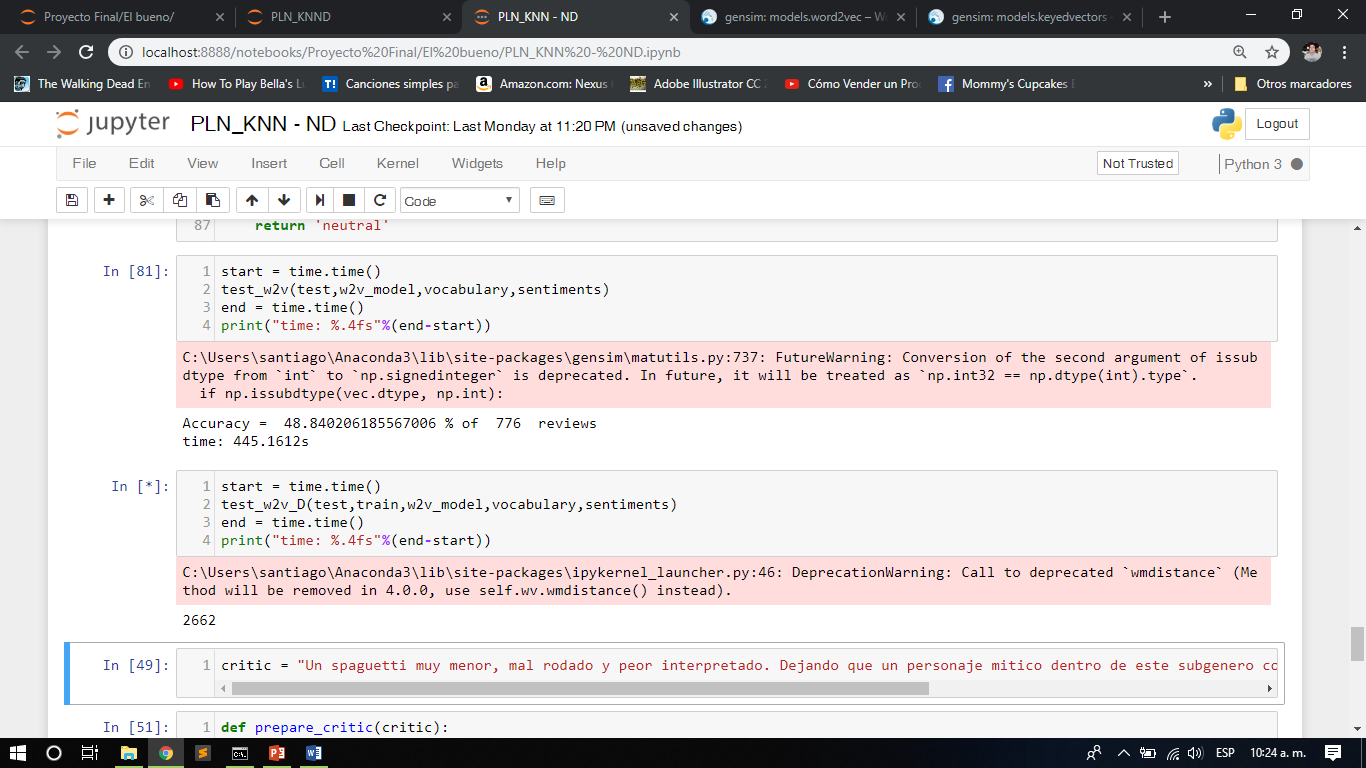


Votación de los sentimientos a nivel de documento

Resultados

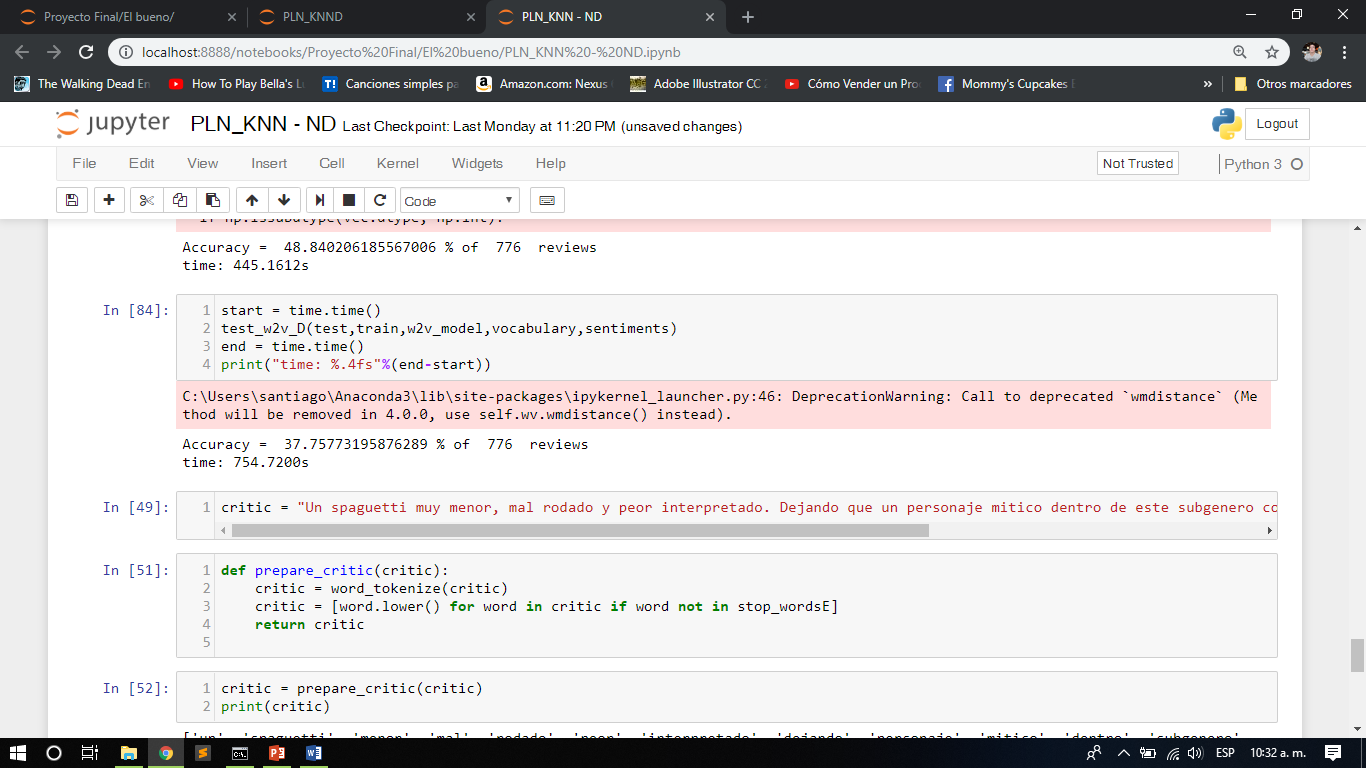
Resultados después de probar el modelo con nuestro clasificador KNN.

Los resultados no son los esperados, entregándonos un porcentaje de predicción del sentimiento muy bajo.



Este resultado es el entregado por la clasificación de las críticas realizando una votación a nivel de las palabras del documento.

Se realizó una segunda clasificación utilizando los vecinos más cercanos a nivel de documento, pero este nos entregó un porcentaje más bajo.



Problemas durante la práctica

1. La memoria RAM. Durante la elaboración de pruebas y utilizando modelos pre-entrenados del word2vec que eran demasiados pesados, causaban que la computadora comenzara a trabarse y no permitía continuar.
2. Las stop words en español. NLTK cuenta con un corpus de stop words en el idioma de español, sin embargo no cuenta con las suficientes palabras ya que en español existen más que en inglés, por lo cual se utilizó la lista que provee nltk y se le agregaron más palabras pero sin embargo aún le falta agregar más palabras.
3. El porcentaje de exactitud que entrega el clasificador. Se utilizaron diferentes estrategias para la clasificación teniendo en un inicio un porcentaje del 32% y se logró aumentarlo a un 48%.

Conclusión

El análisis de sentimientos tiene ya su tiempo en estudio y aun asi su implementación en análisis de textos en el idioma español es un sector muy poco explorado o estudiado por lo que crea un camino amplio para trabajar en el e implementar distintos tipos de aplicaciones y no solo analizando textos, ya que el análisis de los sentimientos se pueden implementar en distintas cosas como analizando el tono de voz, los gestos de la cara, etc.

Después de este proyecto aprendí a cómo analizar de una forma más eficiente y más actual los textos, logrando no solo clasificar por medio de una votación si no que utilizando un modelo que puede entender la semántica de un texto y el sentido de él, que utilizando eso se puede realizar distintas cosas, como generar más clases y determinar si texto es alegre, triste, agresivo, etc. A pesar de que no se logró del todo cumplir con el objetivo del proyecto debido a tiempo y sobre todo conocimiento es un área que me gustaría explorar más.

Referencias

1. Antonio de Jesús García Chávez. (2018). Análisis de documentos de opinión usando la representación word2vec. Ensenada, B.C: CICESE.
2. Kristian Bannister. (10 febrero 2015). Entendiendo el análisis de sentimiento: qué es y para qué se usa. 04/12/2018, de Brandwatch Sitio web: <https://www.brandwatch.com/es/blog/analisis-de-sentimiento/>
3. Gonzalo Ruiz de Villa. (2018). Introducción a Word2vec (skip gram model). 04/12/2018, de Medium Sitio web: <https://medium.com/@gruizdevilla/introducci%C3%B3n-a-word2vec-skip-gram-model-4800f72c871f>
4. markroxor. (2017). Finding similar documents with Word2Vec and WMD. 15/11/2018, de github Sitio web: https://markroxor.github.io/gensim/static/notebooks/WMD\_tutorial.html
5. Vicomtech. (2015). Procesamiento del Lenguaje Natural. 30/05/2018, de Vicomtech Sitio web: http://www.vicomtech.org/t4/e11/procesamiento-del-lenguaje-natural