Tarea final

Jon Ander Ortiz Durántez

17 de mayo de 2014

Resumen

El creciente volumen de datos existente en internet y la necesidad de las organizaciones de ser visibles y evaluar su impacto en internet hacen que sea necesario determinado tipo de herramientas que criben la información superflua de la importante y ofrezcan una visión del posicionamiento en internet.

Los modelos de extracción de información, permiten llevar a cabo la tarea asignada de caracterizar tweets en base a un conjunto de información anterior o corpus. El presente documento plasma un estudio de las técnicas existentes para realizar la tarea de generación del corpus, generación del modelo, evaluación de información objetivo (tweets), y la evaluación de dicho modelo, lo que permite automatizar este proceso de evaluación del impacto online.

Sistema desarrollado

El sistema desarrollado para la presente actividad, aglutina todos los pasos del proceso de extracción de información:

- Creación de un corpus (a partir de los textos devueltos por un buscador generalista de internet).
- Generación de un modelo del lenguaje (basado en N-Gramas).
- Caracterización de la información objetivo (tweets).
- Evaluación de la caracterización realizada.

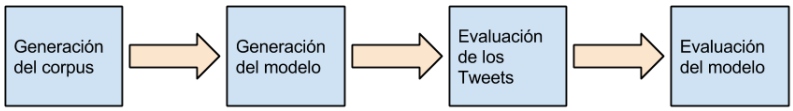


Figura 1: Esquema general del funcionamiento del programa

El sistema ha sido desarrollado en Python, con varias librerías de terceros, entre ellas las mas importantes: NLTK[1] y MITLM [3].

Utilizando NLTK se realizan la mayoría de las operaciones relacionadas con el procesamiento de lenguaje natural en el programa desarrollado. Adicionalmente y debido a problemas de implementación en NLTK[1], se ha generado una nueva versión del desarrollo que utiliza los binarios del proyecto MITLM para caracterizar los tweets utilizando un algoritmo de suavizado avanzado como Knesser-Ney.

Generación del corpus

Para generar el corpus se recorren todos los documentos de referencia de cada uno de los objetivos a analizar. El primer paso realizado es determinar el tipo de fichero a analizar, dado que también se encuentran ficheros en formato PDF a los que hay que hacer un tratamiento especial.

Para determinar el tipo de fichero que tenemos que analizar, utilizamos la librería *python-magic*[2], que se basa en la librería libmagic. Esta librería determina el formato de los ficheros mediante un conjunto de patrones (típicamente de la primera parte del fichero)[3].

[1] https://github.com/nltk/nltk/issues/602

[2] Este nombre es del paquete de debian que incluye dicha librería, el nombre que tiene en pypi (Python package index) es libmagic.

[3] Este tipo de patrones se encuentran actualizados constantemente aquí: http://www.garykessler.net/library/file_sigs.html

NLTK no soporta el parseo directo de los ficheros PDF, por lo que hay que realizar un paso previo: extraer esta información del fichero objetivo. Esta tarea se realiza con la librería *pdfminer* que permite extraer el texto en claro de los ficheros PDF. Cabe destacar que algunos de los ficheros incluidos en la práctica se encuentran protegidos contra la extracción de datos, por lo que hay que contemplar en el código que la extracción falle.

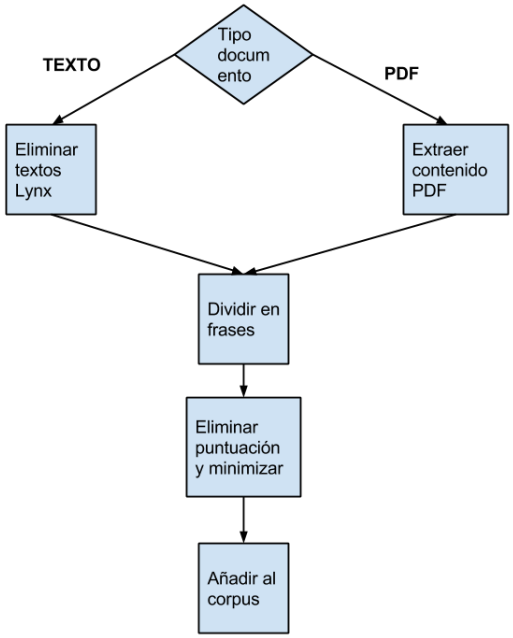


Figura 2: Proceso de generación del corpus

Los ficheros que contienen el texto detectado por *Lynx*, en la web objetivo también necesita de un pequeño procesamiento. El navegador en texto, incluye determinadas etiquetas y campos (debidamente señalados con corchetes - []) que no son parte del texto, por lo que utilizando expresiones regulares y la librería *re* de python se eliminan tanto las referencias a los links como la sección al final donde aparecen las URL's.

Merece la pena valorar si las URL's contenidas en un texto que va formar parte de un corpus contienen información interesante para el mismo o no. El hecho de que exista un enlace puede mostrar información sobre el texto a categorizar, pero para el presente estudio se han descartado.

Una vez que obtenemos el contenido del fichero, se tokeniza por frases. La tokenización por frases se realiza mediante la función *sent_tokenize* de NLTK. Esta tokenización se realiza en base a los signos de puntuación del lenguaje objetivo. Para esta práctica, el lenguaje objetivo es únicamente el inglés, pero para posteriores desarrollos sería interesante agregar un detector de idioma para

utilizar el conjunto de signos de puntuación adecuados, y de este modo realizar la división en frases independiente del lenguaje[4] (y generar corpues diferentes, claro).

Una vez que tenemos todos los textos divididos en frases, se eliminan los elementos que no sean alfanuméricos (signos de puntuación), se traducen todas las letras mayúsculas a minúsculas y se añaden al resultado que es una lista de frases que a su vez son una lista de palabras.

Generación del modelo

Una vez que se ha generado el corpus, se genera el modelo del lenguaje que lo representa. Para la presente tarea se han utilizando N-gramas de tamaños entre 1 y 5 que representan el conjunto de N-gramas que existen en el corpus.

La herramienta utilizada ha sido de nuevo el paquete de modelos de NLTK para los algoritmos de suavizado aditivos, que implementa un modelo de N-gramas y algunos algoritmos de suavizado (Laplace, Lindstone). El N-grama de NLTK, está orientado a aplicar siempre Katz-backoff, es decir, que cuando no encuentra una instancia del N-grama de longitud N, calcula su probabilidad mediante los N-Gramas de menor tamaño.

Para el modelo de suavizado de Knesser-Ney se ha utilizado la librería MITM, que implementa este algoritmo de una manera muy eficiente.

Evaluación del modelo

Para la evaluación de la caracterización realizada por el modelo se han implementado los mecanismos de evaluación de los sistemas de evaluación de los sistemas de extracción de información del libro de referencia de la asignatura [6]: MAP y curvas precision-recall.

La precisión en la extracción de información es la fracción de elementos relevantes en una posición en concreto, y recall es la fracción de elementos relevantes del total de elementos relevantes en ese punto en concreto.

Se representa del siguiente modo:

$$Precision = \mid \frac{R}{T} \mid$$

$$Recall = \mid \frac{R}{U} \mid$$

Donde, R es el número de documentos relevantes en ese punto, T es el número total de documentos y U es el número total de documentos relevante.

Otro aspecto importante es la interpolación de la precisión, para evitar los "dientes de sierra" característicos en los gráficos precision-recall, se calcula una interpolación del valor de la precisión:

$$IntPrecision(r) = max_{i>=r} Precision(i)$$

[4] Una de las implementaciones interesantes para python de detección de idiomas para utilizar en futuros desarrollos es esta: https://github.com/saffsd/langid.py

De este modo, se interpola en once puntos (desde 0 hasta 1) la precisión en función del recall, dando lugar a una curva que muestra la evolución de la precisión con respecto al total de elementos aparecidos, podemos ver un ejemplo de esta evolución en la figura 3. Para graficar los resultados se ha utilizado la librería de representación de datos "*pyplot*" de la librería "*matplotlib*"[4].

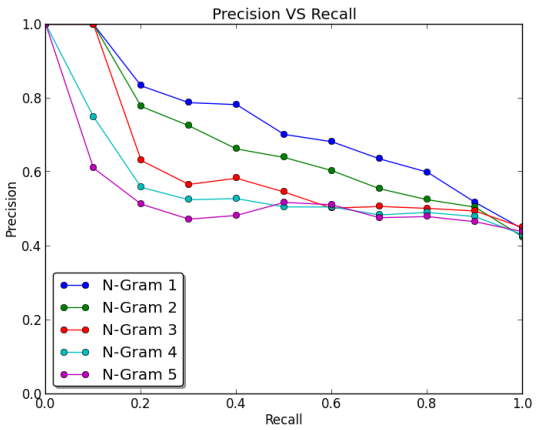


Figura 3: Ejemplo de gráfica precisión - Recall.

También se calcula la métrica MAP (*Mean Average Precision*), para evaluar el sistema de extracción de información para cada uno de los objetivos y longitudes de los N-gramas de los modelos a analizar:

$$MAP = \frac{1}{\mid R_r \mid} \sum_{d \epsilon R_r} Precision_r(d)$$

Pruebas

Se han realizado diferentes pruebas, combinando los diferentes tipos de suavizado existentes. En concreto, se prueban N-Gramas de longitud 1 a 5 con los diferentes tipos de suavizado.

Las pruebas realizadas han diferenciado los orígenes de datos, es decir, que no han unificado en una única gráfica los resultados totales, sino que tenemos diferentes gráficas y valores MAP, uno para cada uno de los orígenes de datos.

Las pruebas se han realizado de este modo para analizar cada caso en particular ya que existen términos con mayores cotas de ambigüedad y de este modo se puede realizar un análisis pormenorizado de cada uno de los términos.

Suavizado de Laplace

El suavizado de Laplace es el mas sencillo de los suavizados posibles, donde para evitar el problema de obtener una probabilidad de 0 cuando es la primera

vez que se observa un N-grama determinado. Para ello, se suma uno a todas las probabilidades:

$$P^*_{Laplace}(w_n|w_{n-1}) = \frac{C(w_{n-1}w_n)+1}{C(w_{n-1})+V}$$

Este suavizado es bastante tosco, y hace que la probabilidad "*migre*" de los elementos con mayor probabilidad a los que no han tenido ninguna ocurrencia. Esto puede hacer que probabilidades que deberían haber sido mayores se reduzcan y amortigüen por efecto de la "*cesión*" de probabilidad que han realizado a los N-gramas que no han aparecido.

Objetivo	BART	BAYER	BLOCKBUSTER
Unigrama	MAP 0.633206443197	MAP 0.82224616073	MAP 0.768257722099
Bigrama	MAP 0.624931933843	MAP 0.804485378194	MAP 0.758089121785
Trigrama	MAP 0.602644312585	MAP 0.785912282159	MAP 0.728521434681
4-Grama	MAP 0.559377397568	MAP 0.740029970505	MAP 0.698630100897
5-Grama	MAP 0.551856263615	MAP 0.693276252794	MAP 0.695661928272
Objetivo	BOINGO	CADILLAC	FENDER
Unigrama	MAP 0.663805267954	MAP 0.770020966924	MAP 0.799363512702
Bigrama	MAP 0.594950468344	MAP 0.746556275067	MAP 0.807245156791
Trigrama	MAP 0.531501811579	MAP 0.723649389713	MAP 0.79740410936
4-Grama	MAP 0.489383535473	MAP 0.713669571272	MAP 0.782024792613
5-Grama	MAP 0.469448615837	MAP 0.70583565553	MAP 0.760608530484
Objetivo	HARPERS	LUXOR	MGM
Unigrama	MAP 0.3833523511	MAP 0.668017925738	MAP 0.800155119683
Bigrama	MAP 0.452205983095	MAP 0.699314263672	MAP 0.819181085775
Trigrama	MAP 0.463894353709	MAP 0.579150819564	MAP 0.74944340884
4-Grama	MAP 0.468776799296	MAP 0.530640310106	MAP 0.669064178865
5-Grama	MAP 0.464270587034	MAP 0.493187210835	MAP 0.619117677677
Objetivo	ROVER		
Unigrama	MAP 0.782283101062		
Bigrama	MAP 0.773529724025		
Trigrama	MAP 0.672417115594		
4-Grama	MAP 0.632719177645		
5-Grama	MAP 0.605207353584		

Cuadro 1: Resultados de MAP para el suavizado de Laplace para las diferentes longitudes de N-gramas.

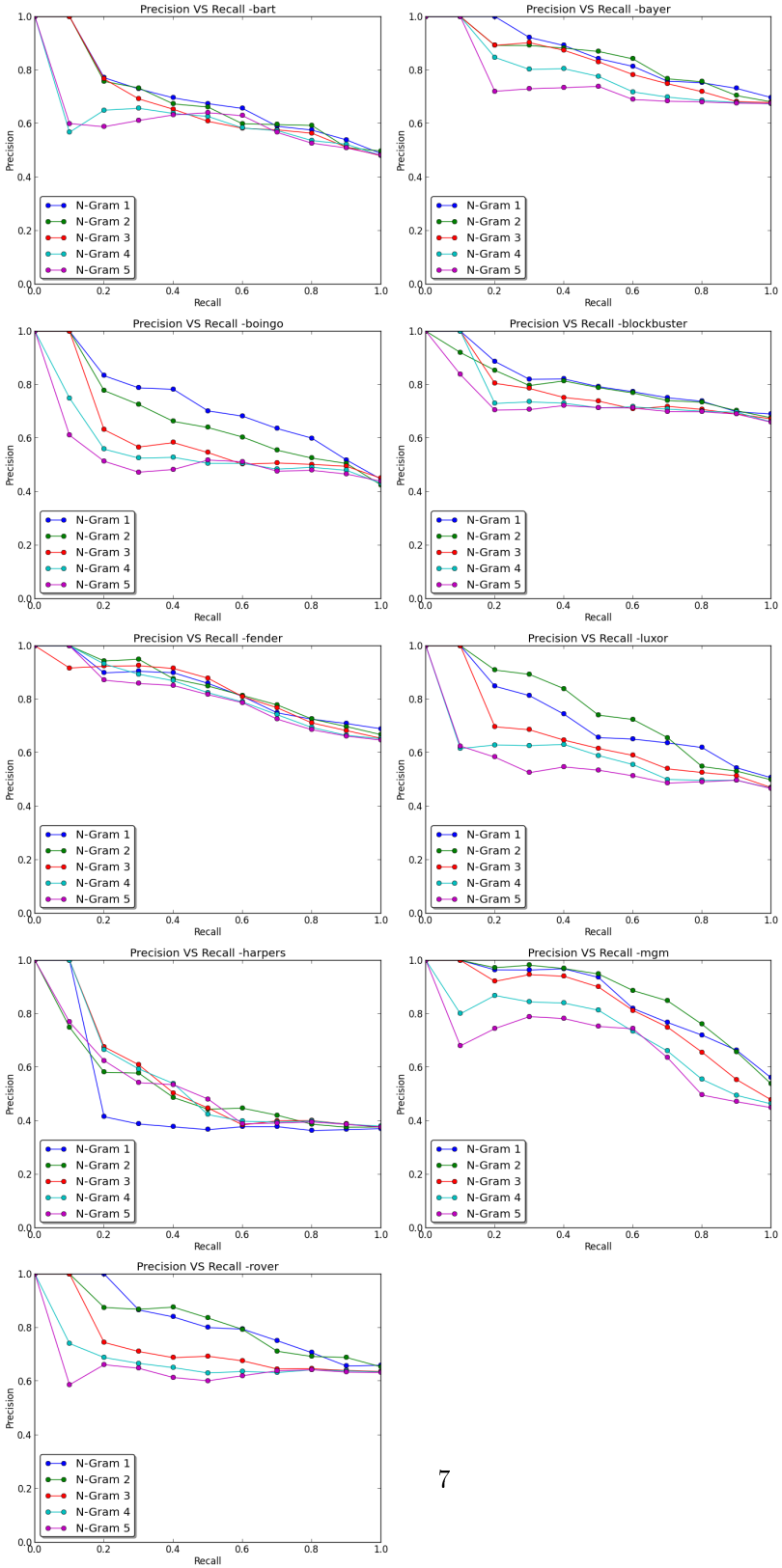


Figura 4: Pruebas con el suavizado de Laplace para las diferentes longitudes de N-gramas.

Suavizado de Lindstone

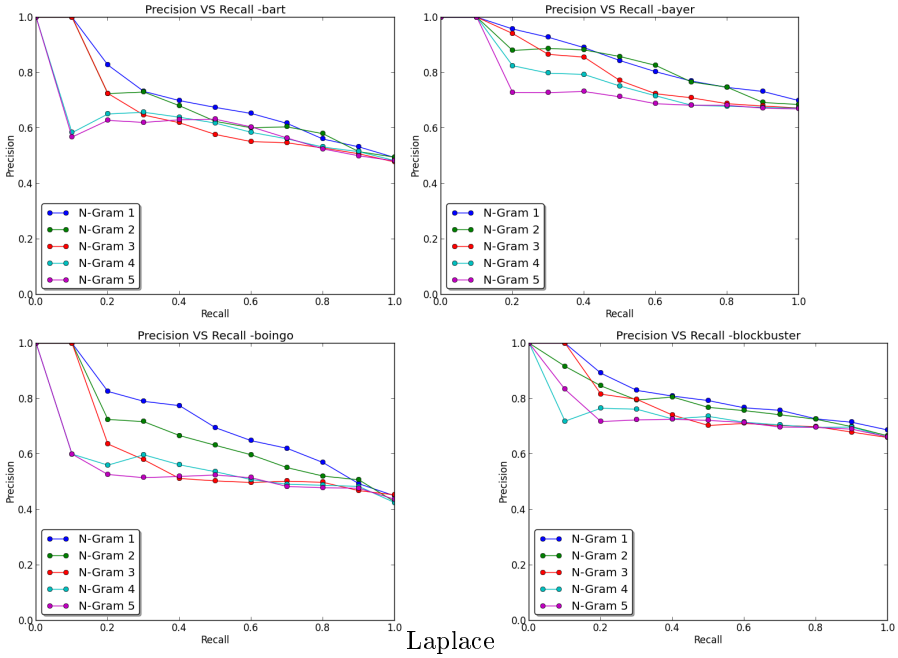
El suavizado de Lindstone es una versión de un suavizado aditivo de Laplace, que se caracteriza por tener un parámetro ajustable (λ) que permite manejar la manera en que el suavizado afecta al algoritmo:

$$P^*_{Laplace}(w_n|w_{n-1}) = \frac{C(w_{n-1}w_n) + \lambda}{C(w_{n-1}) + \lambda V}$$

A continuación unas pruebas con un suavizado de 0,1 en el parámetro λ , que puede tomar valores $0 \leq \lambda \leq 1$, siendo un suavizado de Laplace en caso de llegar al extremo 1, o no existir el suavizado en caso de llegar a 0.

Objetivo	BART	BAYER	BLOCKBUSTER
Unigrama	MAP 0.635507903718	MAP 0.820735645866	MAP 0.7697755544
Bigrama	MAP 0.611855815963	MAP 0.797442856542	MAP 0.750682725287
Trigrama	MAP 0.56965678381	MAP 0.765318248954	MAP 0.712984743731
4-Grama	MAP 0.556780735467	MAP 0.72818918066	MAP 0.697759870438
5-Grama	MAP 0.553051201304	MAP 0.686487987026	MAP 0.695378906108
Objetivo	BOINGO	CADILLAC	FENDER
Unigrama	MAP 0.648474241752	MAP 0.772870659391	MAP 0.811831697705
Bigrama	MAP 0.595571283463	MAP 0.737283063696	MAP 0.801966967498
Trigrama	MAP 0.521541578503	MAP 0.6946847824	MAP 0.7737239536
4-Grama	MAP 0.498294287327	MAP 0.698156044263	MAP 0.767411497545
5-Grama	MAP 0.478530979191	MAP 0.699425447824	MAP 0.751158979071
Objetivo	HARPERS	LUXOR	MGM
Unigrama	MAP 0.388028982543	MAP 0.660056450502	MAP 0.791550691769
Bigrama	MAP 0.456652924792	MAP 0.685806555407	MAP 0.810076641956
Trigrama	MAP 0.471470104497	MAP 0.531686538368	MAP 0.689813513852
4-Grama	MAP 0.480265572215	MAP 0.505506266293	MAP 0.61258264325
5-Grama	MAP 0.476438128437	MAP 0.481469522679	MAP 0.608250205261
Objetivo	ROVER		
Unigrama	MAP 0.774455270111		
Bigrama	MAP 0.763654549062		
Trigrama	MAP 0.6455997067		
4-Grama	MAP 0.619017600928		
5-Grama	MAP 0.601815817319		

Cuadro 2: Resultados de MAP para el suavizado de Lindstone con $\lambda = 0{,}1$ para las diferentes longitudes de N-gramas.



Laplace

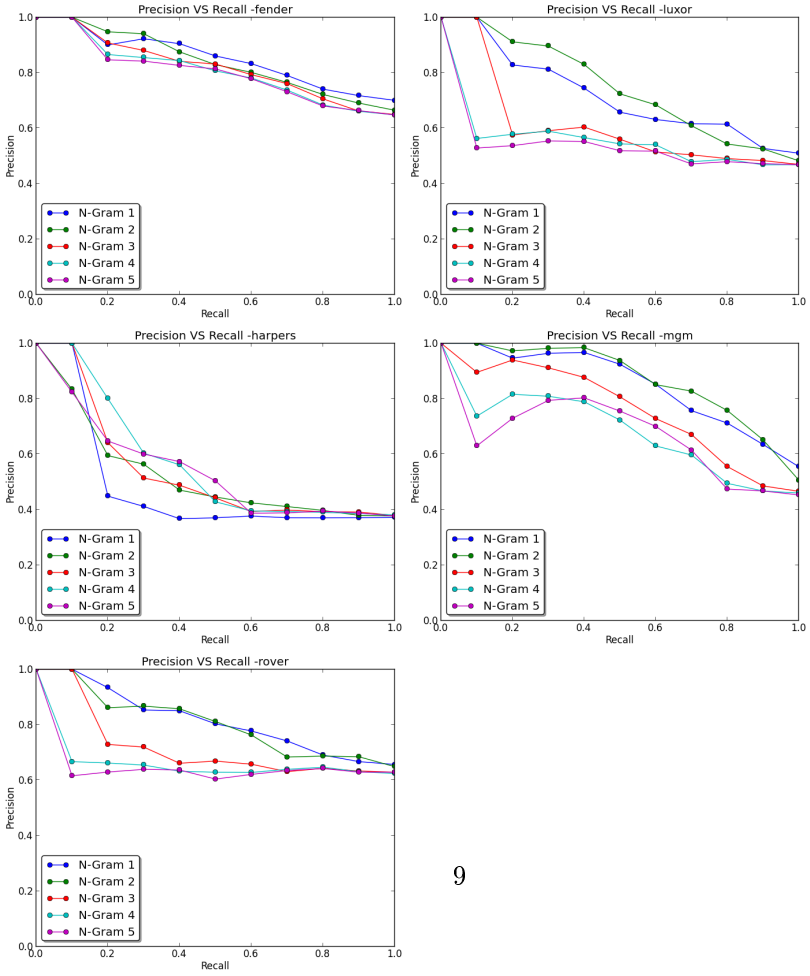


Figura 5: Pruebas con el suavizado de Lindstone $\lambda = 0{,}1$ para las diferentes longitudes de N-gramas.

Knesser-Ney

El algoritmo de Knesser-Ney pertenece a la familia de algoritmos que se desarrollaron a partir del algoritmo presentado por Chen y Goodman (Good-Turing[2]) esta familia de algoritmos se basa en interpolar la probabilidad de un N-grama en base a los N-gramas conocidos de orden inferior. Es decir, en lugar de "*como se ajusta el trigrama A B C*" se calcula "*Como de bueno es C tras el bigrama A B*",es decir, utiliza un nuevo enfoque que, en lugar de ver cuántas veces ha aparecido el objetivo, calcula el número de veces que esperamos ver una palabra en un contexto desconocido.

Esta probabilidad se expresa de la siguiente manera:

$$P_{CONTINUATION}(w_i) = \frac{|\{w_{i-1} : C(w_{i-1}w_i) > 0\}|}{\sum_{w_i} |\{w_{i-1} : C(w_{i-1}w_i) > 0\}|}$$

Para esta prueba no se han podido utilizar la implementación de N-gramas de la librería NLTK como en las pruebas anteriores, dado que arrastra problemas muy importantes de implementación, para suplir esta falta, se ha utilizado la librería MITLM[3][5], que implementa el algoritmos Knesser-Ney como algoritmo de suavizado a la hora de calcular las probabilidades.

[5]https://code.google.com/p/mitlm/ , esta librería se basa en otra mas conocida como SRILM de la que toma parte de la implementación.

Objetivo	BART	BAYER	BLOCKBUSTER
Bigrama	MAP 0.54346978406	MAP 0.799672519698	MAP 0.750221421178
Trigrama	MAP 0.549231554052	MAP 0.798775279208	MAP 0.747837083881
4-Grama	MAP 0.552844091613	MAP 0.801453971516	MAP 0.754394668195
5-Grama	MAP 0.553150498522	MAP 0.802419043272	MAP 0.752356540215
Objetivo	BOINGO	CADILLAC	FENDER
Bigrama	MAP 0.515773762992	MAP 0.696997132072	MAP 0.801028153946
Trigrama	MAP 0.510059135346	MAP 0.701134008626	MAP 0.779750327842
4-Grama	MAP 0.51062486088	MAP 0.702238208654	MAP 0.780848094334
5-Grama	MAP 0.510934281121	MAP 0.704932566572	MAP 0.782868851455
Objetivo	HARPERS	LUXOR	MGM
Bigrama	MAP 0.4502363827	MAP 0.654622354803	MAP 0.687729566755
Trigrama	MAP 0.457204218006	MAP 0.650630606059	MAP 0.687661367659
4-Grama	MAP 0.459701011981	MAP 0.634537157661	MAP 0.690842316148
5-Grama	MAP 0.458203190883	MAP 0.636313827786	MAP 0.690519816052
Objetivo	ROVER		
Bigrama	MAP 0.773510116557		
Trigrama	MAP 0.757525174966		
4-Grama	MAP 0.76189992338		
5-Grama	MAP 0.763837564772		

Cuadro 3: Resultados de MAP para el suavizado Knesser-Ney para las diferentes longitudes de N-gramas.

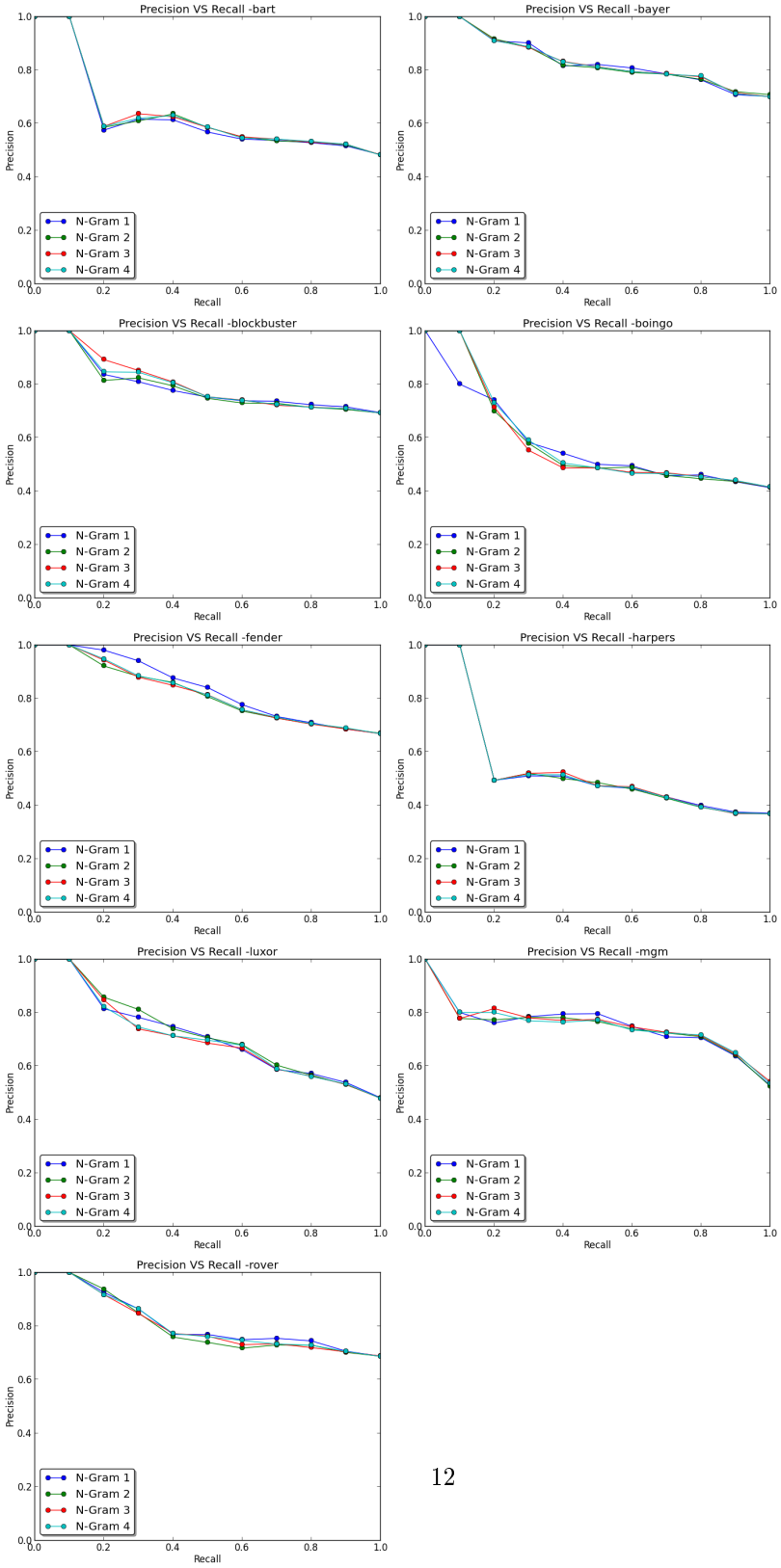


Figura 6: Pruebas con el suavizado de Knesser-Ney para las diferentes longitudes de N-gramas.

Como se puede observar en los datos de esta prueba en el Cuadro 3 , lo más relevante es la homogeneidad que se encuentra entre los resultados para los diferentes N-gramas. En los casos con los suavizados aditivos, los N-gramas de ordenes mayores perdían precisión, porque se añadía información no relevante al modelo (aunque la implementación de NLTK soporta katz backoff), pero con el algoritmo de suavizado de Knesser-Ney, observamos como los N-gramas de órdenes mayores, mantienen la precisión. Esto se da por que al ajustarse en base a los N-gramas de órdenes inferiores mantienen la información de éstos y se ajustan de una manera mas adecuada que únicamente comparando el N-grama.

Limitación del Corpus

Uno de los problemas que podemos encontrar a la hora de utilizar un recurso web es que mucha de la información contenida en la misma es completamente superflua, lo que nos lleva a almacenar N-gramas en nuestro modelo que realmente son completamente superfluos (en la Figura 7 se señala cómo en una web existe mucho texto superfluo que no tiene que ver con el objetivo).

En la práctica las páginas web están repletas de textos superfluos en lugar de contener únicamente información interesante. Para esta prueba se ha procedido a limitar el corpus a las frases que contengan alguna instancia del objetivo. Aunque a priori parezca una técnica bastante burda, en la práctica aumenta el valor todas los resultados de MAP del modelo para todos los N-gramas.

Esta simple prueba demuestra que existe una pérdida de precisión por las frases superfluas existentes en el corpus, y que una manera de mejorar los resultados se basa en realizar un preprocesamiento a los datos insertados en el corpus.

Figura 7: Ejemplo de la complejidad de extracción de información de la web (Extracto del curso online de Dan Jurafsky[5]).

Objetivo	BART	BAYER	BLOCKBUSTER
Unigrama	MAP 0.63904951643	MAP 0.83440708137	MAP 0.776433231731
Bigrama	MAP 0.618475813132	MAP 0.813435090918	MAP 0.750507845654
Trigrama	MAP 0.596840106029	MAP 0.755174383894	MAP 0.710705147757
4-Grama	MAP 0.573659747031	MAP 0.72987980051	MAP 0.70858551067
5-Grama	MAP 0.553462083218	MAP 0.689775653032	MAP 0.707083422505
Objetivo	BOINGO	CADILLAC	FENDER
Unigrama	MAP 0.684757197128	MAP 0.783494246437	MAP 0.839700252864
Bigrama	MAP 0.574987764368	MAP 0.741269462163	MAP 0.8045036879
Trigrama	MAP 0.49984339451	MAP 0.708320993611	MAP 0.791546707989
4-Grama	MAP 0.477786671007	MAP 0.708509059478	MAP 0.766182585804
5-Grama	MAP 0.458538699968	MAP 0.704944906477	MAP 0.745088520461
Objetivo	HARPERS	LUXOR	MGM
Unigrama	MAP 0.376999657984	MAP 0.671098773812	MAP 0.804186634862
Bigrama	MAP 0.435881692491	MAP 0.683877021233	MAP 0.808794003531
Trigrama	MAP 0.401981064434	MAP 0.567677886747	MAP 0.681572420049
4-Grama	MAP 0.391796446343	MAP 0.554199644713	MAP 0.63098673354
5-Grama	MAP 0.388841549827	MAP 0.533344155468	MAP 0.613099304146
Objetivo	ROVER		
Unigrama	MAP 0.784296079656		
Bigrama	MAP 0.764154449264		
Trigrama	MAP 0.645022640962		
4-Grama	MAP 0.622518202196		
5-Grama	MAP 0.60338914108		

Cuadro 4: Resultados de MAP para el corpus limitado.

Problemas encontrados

Generación del corpus

A la hora de la generación del corpus se han encontrado algunas dificultades:

- Documentos heterogéneos: el corpus contenía tanto texto claro como texto en PDF, por lo que se tuvo que implementar un mecanismo de recuperación de textos de documentos PDF.
- Algunos de estos textos en PDF tienen mecanismos que no permiten la extracción de datos de los mismos, lo que hay que controlar en el proceso de extracción de datos.
- URL's y datos asociados a HTLM. Aunque Lynx (que es el navegador con el que se han generado los volcados de HTML que componen el corpus) separa las URL's inserta también determinados campos y URLs para hacer

navegable su interfaz de texto. Aunque de las URL's se puede extraer información útil, para el presente proyecto no se han considerado, por lo que se ha optado por recurrir expresiones regulares para filtrar URL's y las etiquetas insertadas por Lynx.

- Otro de los dilemas que preceden a la generación del corpus son las palabras mayúsculas, para evitar problemas, tanto el corpus como los tweets se han traducido a minúsculas.

Evaluación

Uno de los primeros problemas encontrados a la hora de evaluar los tweets, ha sido que los modelos se evaluaban mejor los tweets cortos que los tweets largos. Este fenómeno se intensificaba conforme en grado los N-gramas subía, por lo que todo apuntaba hacia la necesidad de la normalización de la perplejidad (nPP) [7] para eliminar la desventaja que tenían los tweets largos (que tienen mas probabilidad de contener N-gramas ante los cuales el modelo de un valor de perplejidad mayor).

Modelos

Uno de los mayores problemas encontrados durante el desarrollo ha sido los continuos problemas con los algoritmos de suavizado en los modelos de N-Gramas de NLTK.

Los algoritmos de Good-Turing (en sus dos versiones, Simple-GoodTuring y el GoodTuring clásico), Knesser-Ney y Witten-Bell fallan con el modelo de N-gramas, así como los modelos de Knesser-Ney y Witten-Bell. Esto hace que sea prácticamente imposible comprobar con NLTK cualquier algoritmo de suavizado fuera de los suavizados aditivos.

Este problema ha hecho que utilicemos la utilidad "MIT Language Modeling" o mitlm, para poder probar el sistema contra alguritmos de suavizado diferente de los aditivos (En este caso Knesser-Ney).

Estos problemas hacen realmente muy poco recomendable la utilización de esta librería y de su modelo para futuros desarrollos, incluso, dado el ámplio número de incidencias relacionadas con los n-gramas (muchas de ellas relacionadas con los algoritmos de suavizado), en la versión de desarrollo de nltk han eliminado la implementación del repositorio[6].

Resultados, recomendaciones y trabajo futuro

Los resultados obtenidos se pueden calificar como satisfactorios, ya que los modelos basados en N-gramas clasifican de un modo bastante acertado los tuits en función del corpus que se ha creado a partir de una búsqueda en internet. A continuación se detallan los resultados por áreas y se presentan algunas recomendaciones y futuras líneas de trabajo.

[6] https://github.com/nltk/nltk/commit/73f7e7b9ec301c39dc11035f138a27feb3c436cf

Clasificación basada en N-Gramas

Aunque los N-gramas se hayan clasificado utilizando dos librerías diferentes y sea complicado comparar los resultados obtenidos de ambos modelos, se pueden apreciar patrones comunes bien en general y bien por objetivo:

- Excepto excepciones (justificadas a continuación), en los algoritmos aditivos los unigramas representan mejor la información que N-gramas mas largos, incluso aunque se haya normalizado la perplejidad en función de la longitud de la frase objetivo: Esto se da porque en los N-gramas se almacena también información no relevante, o superflua que hace que la caracterización no sea la adecuada.

    - Para evitar este efecto de "almacenamiento de información superflua", se recomienda seguir el modelo marcado por la prueba con el corpus limitado (que obtiene mejores resultados que sin limitar el corpus a las frases en las que se menciona la entidad objetivo), y generar un corpus utilizando únicamente las partes en las que la entidad objetivo es una entidad nombrada[7].

- El algoritmo Knesser-Ney funciona correctamente para órdenes mayores de n-gramas que los algoritmos aditivos.

- Existen objetivos que se comportan particularmente bien con bigramas, como por ejemplo "*fender*", "*luxor*" y "*mgm*", esta característica es porque la probabilidad de utilizar la palabra objetivo junto con otra como *"fender stratocaster"*, *"luxor hotel"* o *"mgm movies"* es muy alta y caraceriza muy bien el tweet.

- Existe también un caso paradigmático como "*harpers*", donde los 4-gramas y los 5-gramas se comportan mejor que los unigramas o bigramas. Esto es porque los tweets relacionados son referencias a nombres de artículos de la revista, que por supuesto aparecen también en el corpus, por lo que estos N-Gramas mas largos funcionan mejor (al ser una copia literal del titulo), que las probabilidad de encontrar las palabras sueltas.

- Es particularmente llamativo el resultado de "*mgm*" en los suavizados aditivos para los n-gramas de altos órdenes en los gráficos precision - recall: cómo pierden precisión para posteriormente ganarla. Esto puede darse porque existen N-gramas de orden 4-5 que se repiten mucho (frases hechas o expresiones) que hacen que se distorsione el corpus y el modelo, por lo que tweets repletos de expresiones se clasifican demasiado alto.

- Utilizar la perplejidad normalizada (nPP) es obligatorio para no perjudicar a los tuits largos.

[7] De hecho esta es la tarea voluntaria de la práctica

Recomendaciones

Las recomendaciones que resultan del presente estudio serían:

1. Utilizar un corpus reducido (utilizando la identificación de entidades nombradas, o algún tipo de preprocesamiento de la información) ya que como se ha visto en el apartado *"corpus reducido"*, el hecho de limitar qué frases se incorporan al corpus, hace que la precisión de la clasificación aumente.

2. Utilizar un modelo basado en n-gramas:

   a) Si se elige un algoritmo de suavizado aditivo, elegir el de Lindstone con el valor de λ más bajo posible (para no distorsionar el sistema al "*robar*" probabilidad del resto de elementos) junto con una longitud corta de N-gramas, bigramas o incluso unigramas (que son superiores en todas las pruebas a los n-gramas de órdenes mayores), aunque quizá no sea la mejor alternativa en nombres de entidades que puedan contener ambigüedades.

   b) Si se elige un algoritmo avanzado del tipo Knesser-Ney, la recomendación sería elegir una longitud de N-grama lo más larga posible, dentro de las capacidades de cómputo que existan.

3. Normalizar la perplejidad obtenida en función de la longitud del tweet analizado.

4. Utilizar N-gramas lo mas largos posibles para los objetivos mas ambiguos.

Posibles ampliaciones del sistema

El hecho de que el corpus sea el resultado de una consulta en un buscador de internet hace que se pueda automatizar la generación del corpus diariamente, de modo que podamos actualizar el corpus con los últimas noticias, y dar una clasificación actualizada, o incluso comparar o generar un informe sobre cuáles son las claves o las palabras que se asocian a cada marca / objetivo.

Otra de las ampliaciones interesantes sería dar soporte para múltiples idiomas. El corpus provisto únicamente contenía textos en inglés, pero se puede detectar automáticamente el idioma, y utilizar los elementos específicos del idioma para dividir las sentencias y tokens a la hora de generar el corpus.

Referencias

[1] Steven Bird, Ewan Klein, and Edward Loper. *Natural language processing with Python*. O'Reilly Media, Inc., 2009.

[2] Stanley F Chen and Joshua Goodman. An empirical study of smoothing techniques for language modeling. In *Proceedings of the 34th annual meeting on Association for Computational Linguistics*, pages 310–318. Association for Computational Linguistics, 1996.

[3] Bo-June Hsu and James Glass. Iterative language model estimation: efficient data structure & algorithms. In *Proceedings of Interspeech*, volume 8, pages 1–4, 2008.

[4] J. D. Hunter. Matplotlib: A 2D graphics environment. *Computing In Science & Engineering*, 9(3):90–95, 2007.

[5] Dan Jurafsky and Chris Manning. CKY Parsing - Standford open course online.

[6] Dan Jurafsky, James H Martin, Andrew Kehler, Keith Vander Linden, and Nigel Ward. *Speech and language processing: An introduction to natural language processing, computational linguistics, and speech recognition*, volume 2. MIT Press, 2000.

[7] Xunying Liu, Mark JF Gales, and Philip C Woodland. Context dependent language model adaptation. In *INTERSPEECH*, pages 837–840, 2008.