Introducción al Procesamiento del Lenguaje Natural

Obligatorio 2

2013

Grupo 01

Santiago Castro (CI: 4501276)
Jennifer Esteche (CI: 5068197)
Romina Romero (CI: 4656771)

Índice de contenido

1 Introducción

El objetivo de este obligatorio es conocer y utilizar herramientas de procesamiento de lenguaje natural aplicadas a problemas prácticos. Para ello se trabajará en la implementación de un sistema de resumen de opiniones sobre restaurantes.

Se busca resumir los comentarios que han sido publicados por clientes de restaurantes en una página web. Para ello se debe procesar este corpus, extraer los comentarios e intentar resumirlos agrupándolos según su similitud.

Además de las herramientas ya utilizadas para el laboratorio anterior, se utilizarán un POS Tagger, Wordnet y algoritmos de Clustering.

Por más información, ver [12].

2 Desarrollo

2.1 Etiquetado gramatical

En primera instancia se implementó un *unigram tagger* para etiquetar las palabras, sin utilizar un tagger *backoff*[1].

Al evaluar este tagger contra el conjunto de test, que ya está correctamente *taggeado,* se obtuvo una *accuracy* que se encontraba entre los valores 0.8 y 0.9. Este valor indica que el resultado de las etiquetas aplicadas es de regular a aceptable, apreciándose una cantidad importante de etiquetas *None* tanto en adjetivos, como en sustantivos y verbos; esto constituye un impedimento para considerarse como una buena implementación, ya que luego se descartarán todas las palabras que no sean identificadas como pertenecientes a una de las tres categorías gramaticales recién mencionadas.

Para mejorar esta implementación inicial, se utilizó un *bigram tagger* que empleaba el *unigram tagger* ya implementado como *backoff* .

La evaluación de este nuevo *tagger* dio 0.9287 (se probaron varias ejecuciones ya que el corpus se divide en training y testing en forma aleatoria, en proporciones de 90% y 10% respectivamente, y el valor medio fue el indicado). Analizando los resultados obtenidos, se nota que siguen existiendo algunos valores *None*, y algunas de las palabras no están siendo correctamente etiquetadas en el contexto en el que se encuentran.

Debido a ello, se realizó una tercera mejora al *tagger*, implementando un *trigram tagger* que emplea el *bigram tagger* como *backoff* (éste a su vez utiliza el *unigram tagger* como *backoff* propio), de forma que se considere en mayor medida el contexto a la hora de elegir la etiqueta adecuada.

La evaluación de esta última implementación arrojó un valor de 0.9291. Esto significa una mejora, pero aún es pequeña respecto a la implementación anterior, originalmente se esperaba llegar a un cambio positivo más sustancial. En cuanto al análisis, siguen existiendo algunas etiquetas *None*, aunque mejora respecto a las opciones anteriores. Este es un punto en el cual se podría seguir mejorando.

Las etiquetas *None* significan que la palabra no pudo ser etiquetada por ninguno de los tres *taggers* disponibles (*unigram*, *bigram*, *trigram*). Por ello, se probó agregar un *backoff tagger* al *unigram tagger* para observar los resultados. Éste último, *default tagger*, siempre etiquetaba como sustantivos (*Noun*) a todas las palabras que le llegaban (etiqueta *'N'*). La evaluación mostró un valor muy similar a la de la implementación anterior, se carecía, lógicamente, de falta de etiquetas, pero no todos las palabras que no poseían etiqueta correspondían a sustantivos, por lo cual el etiquetado era incorrecto y en términos generales tampoco era mejor, debido a todo esto, se descartó dicha opción.

En búsqueda de un *backoff tagger* para el *unigram tagger*, se implementó un *tagger* que utiliza

1 Tagger backoff: es un tagger que agrega una etiqueta a una palabra si es posible, cuando el tagger principal no la pudo etiquetar.

expresiones regulares. Este *tagger* sencillamente etiqueta una palabra según la expresión regular que la acepta. La selección de expresiones regulares se hizo en base a los sufijos más comunes en el idioma español para las categorías de verbos, sustantivos y adjetivos (que son las categorías con las que interesa quedarse para el análisis posterior), por ejemplo el sufijo *-oso* se encuentra habitualmente en adjetivos, como ser nervi*oso*, ansi*oso*, temer*oso*, perez*oso*. Las palabras que no cumplen con ninguna de estas expresiones, siguen marcándose como *None*, es decir, sin etiqueta. Muchas de estas palabras correspondían a errores del tokenizador, por ejemplo *'...algo'* fue tokenizado como una palabra.

Vale aclarar que el idioma español tiene un gran número de palabras irregulares, por lo cual los sufijos más comunes no son una garantía de clasificación correcta, pero se espera que el número de errores sea suficientemente bajo en el contexto dado y para el alcance propuesto en el laboratorio. Asimismo también se presentó el inconveniente de que la herramienta permite un máximo de 100 expresiones regulares, por lo cual no se pudieron especificar todos los sufijos deseados.

Al contrastar esta implementación con el corpus de prueba, se observa que la evaluación numérica no mejoró. Sin embargo se logró etiquetar exitosamente un número significativo de palabras que no eran etiquetadas por los *taggers* de N-gramas. Se mantienen algunos inconvenientes en cuanto a la falta de etiquetas, como es el caso de la palabra *'lindo'*, y en cuanto a problemas de identificación contextual, como es el caso de *'medio crudo'*, donde se etiqueta como adjetivos calificativos tanto a *'medio'* como a *'crudo'* (cuando 'medio' es un adverbio en este contexto). Estos detalles no se pueden mejorar así como se viene modelando, se requeriría implementar un *tagger* que analice el contexto en forma distinta.

Esta última implementación fue la elegida. El uso de N-gramas disminuye un poco el rendimiento, pero se gana en las tasas de acierto.

2.2 Wordnet

Análisis de synsets obtenidos

WordNet es una base de datos léxica y semántica, al que se le carga una versión para la lengua española. Nombres (sustantivos) y verbos, principalmente, son agrupados en conjuntos de sinónimos cognitivos (synsets), cada uno de los cuales expresa un concepto distinto.

Los resultados obtenidos al consultar los synsets de la lista `['comida', 'mozo', 'bebidas', 'precio', 'feo', 'delicioso', 'costoso']` fueron:

```
[Synset('alimento.n.01'), Synset('almuerzo.n.01'), Synset('alimento.n.02'),
Synset('alimentación.n.03'), Synset('comida.n.05'), Synset('alimento.n.03'),
Synset('comida.n.07'), Synset('almuerzo.n.03'), Synset('cena.n.02'),
Synset('comida.n.10')]

[Synset('fajín.n.01'), Synset('chaval.n.03'), Synset('mozo.n.03'),
Synset('mozo.n.04'), Synset('adolescente.n.02')]

[Synset('bebida.n.01'), Synset('bebida.n.02'), Synset('bebida.n.03'),
Synset('bebida.n.04')]
```

```
[Synset('precio.n.01'), Synset('costa.n.01'), Synset('coste.n.02'),
Synset('precio.n.04'), Synset('condiciones.n.02'), Synset('precio.n.06')]

[Synset('feo.a.01'), Synset('desagradable.a.03'), Synset('feo.a.03')]

[Synset('delicioso.a.01'), Synset('agradable.a.13'), Synset('delicioso.a.03')]

[Synset('caro.a.01'), Synset('caro.a.02'), Synset('costoso.a.03')]
```

Cada synset recibe su nombre de una de las palabras que toman dicho significado, seguido de un identificador de categoría y un número. Por ejemplo, el synset "*alimento.n.01*" representa una de las acepciones del sustantivo *alimento.* Si se ejecuta el siguiente código, se puede visualizar el funcionamiento de los synsets:

```
print wncr.synset("alimento.n.01").definition

cualquier sustancia que puede ser metabolizado por un animal para dar
energía y construir tejido

print wncr.synset("alimento.n.01").lemmas

[Lemma('alimento.n.01.alimento'), Lemma('alimento.n.01.comida'),
Lemma('alimento.n.01.nutriente'), Lemma('alimento.n.01.nutrimento'),
Lemma('alimento.n.01.sustento')]
```

En este caso, dicho synset tiene el significado mostrado, y tiene 5 lemas que comparten dicho significado, i.e., se puede decir que *comida* es un sinónimo de *nutriente* bajo el significado *"cualquier sustancia que puede ser metabolizado por un animal para dar energía y construir tejido"*.

Se puede ver a un synset como un significado o acepción en concreto, y sus lemas son palabras que toman ese significado. Cabe aclarar que aquellas palabras que están en más de una categoría gramatical, tendrán significados para cada una de esas categorías.

Se observa que *comida* pertenece a 10 synsets, 3 de los cuales son nombrados usando el sustantivo *alimento* (con distintas acepciones), y también aparecen dos veces *comida* y *almuerzo.* En el caso de *mozo* sólo se repite el sustantivo *mozo*. Además se percibe que en algunos casos sus distintos synsets distan mucho en significado entre ellos. Por ejemplo, al ejecutar la sentencia `print wncr.synset('faj\xc3\xadn.n.01').definition` el resultado obtenido es `NULL`, y buscando en diccionarios, la palabra *fajín* se define como "*Ceñidor de seda de determinados colores y distintivos que pueden usar los generales o los jefes de administración y otros funcionarios"*.

En el caso de *bebida* aparecen 4 synsets relacionados, todos de la forma bebida.n.0_ (donde _ es un dígito entre 1 y 4), y al preguntar por los significados de cada uno de estos synsets, también se obtiene el resultado `NULL`.

En el caso de *precio* los nombres de los synsets son muy similares entre sí. La herramienta no cuenta con la mayoría de las definiciones para dichos synsets; se puedo obtener sin embargo las definiciones de *'precio.n.04'* que es *"gratificación por ayudar a atrapar a un delincuente"*, y de

'precio.n..06' que es "*coste de sobornar a alguien*". Estas acepciones difícilmente aplicarían al contexto de opiniones de restaurantes en el que se está trabajando. La definición de *'condiciones.n.02'*: *"cantidad de dinero necesaria para comprar algo"* es bastante apropiada al corpus.

Para la palabra *feo* se devuelven 3 synsets. Las definiciones de los synsets *'feo.a.01'* y *'feo.a.03'* no están disponibles, la de *'desagradable.a.03'* es "*desagradable a los sentidos*".

El caso de delicioso es muy similar, se cuenta con 3 synsets, presumiblemente adecuados al contexto dado, y el único synset con definición disponible es *'agradable.a.13'*: "*agradable para el sentido del gusto*".

Finalmente, costoso tiene 3 synsets: *'caro.a.01'*, *'caro.a.02'*, *'costoso.a.03'*, ninguno de los cuales cuenta con su definición.

2.3 Similitud avanzada

2.3.1 Diseño

El tokenizador primero usa *word_tokenize* para separar la oración en palabras. Siguiendo la sugerencia de la letra del proyecto, se emplea el POS Tagger para ver a qué categoría gramatical corresponde cada palabra. En caso de ser un sustantivo o adjetivo, se utilizan sus synsets pertenecientes a dicha categoría, en caso de tratarse de un verbo, se aplica el *SpanishStemmer* ofrecido por *NLTK*.

Para cada palabra que sea sustantivo o adjetivo, se toman las distintas palabras que definen los nombres de los synsets. Por ejemplo, para la palabra *comida*, se agregan los tokens: *comida*, *alimento* y *almuerzo*. Para los verbos se deja el lema, según *SpanishStemmer*, como token. El resto de las palabras no son tenidas en cuenta.

Las funciones *calcular_similitud* y *calcular_similitudes* se mantienen prácticamente incambiadas desde el laboratorio anterior. La función *imprimir_similares* sufre ligeros cambios: ahora muestra el top 50 de los pares representados en el vector *similitud*, pero si se repite algún par de forma conmutada, se omite.

Cabe destacar a su vez que la clase *CountVectorizer* no se instancia con la opción de quitar los acentos ya que el WordNet no reconocerá bien las palabras si están mal escritas.

Además, en el *CountVectorizer* se estableció la opción binary a *true* y se observaron las diferencias respecto a su configuración en *false*. La mayoría de las oraciones en el top 50 permanecen incambiadas, notándose dos cambios significativos:

- El par *'El lugar muy bien ambientado, la atención es de destacar y la comida de muy buena calidad y abundante.', 'El lugar es muy agradable, la atención es excelente, y la comida exquisita.'*, aumenta su similitud de 0.883 a 0.887 al habilitar la opción binary, por lo cual su posición en el ranking sube dos lugares.

- Se detecta el par *'Como dice Monine el precio es por encima de la media que dice la página.', 'Precio acorde, pero por encima del que estipula la página.'*, que no aparecía sin la opción binary, y su similitud es de 0.886, lo que lo coloca en la posición 30.

- Sin la opción binaria, se obtiene el par *'El lugar es muy agradable, la atención es fantástica y la comida es muy buena.Pedimos Saltimboca con puré rustico que estaba muy rico y Ravioles de Masa Negra de Salmón con Mejillones que también estaba delicioso.', u'El lugar es muy agradable, la atención es excelente, y la comida exquisita.'*, que ocupa la posición 38, con una similitud de 0.868, el cual queda fuera del top 50 con la opción binary. Además se observa aquí que la falta de espacio luego del punto provoca que el tokenizer no pueda diferenciar las oraciones.

Se concluye que no hay diferencias significativas entre activar o no esta opción, para el corpus dado.

2.3.2 Análisis de similitudes

Los resultados obtenidos en general tienen niveles de similitud buenos o aceptables. Sin embargo, hay algunos resultados a mejorar, por ejemplo:

```
(u'Los postres buenísimos.', u'Los postres no tanto.', 1.0)
```

Aquí el algoritmo encuentra estas dos oraciones como totalmente similares, ya que coinciden en la palabra *postres*, pero en realidad son frases contrarias (mirando el contexto, la segunda oración está diciendo que los postres no están tan bien servidos ni son tan ricos). Igualmente se puede destacar que ambas oraciones tratan sobre postres.

El uso de sinónimos mejora la búsqueda de similitudes en varios casos, por ejemplo:

```
(u'Buena relación costo calidad, muy disfrutable.', u'Excelente relación calidad-precio.', 0.89167472590032015).
```

2.3.3 Palabras obtenidas

Respecto a las palabras obtenidas, se observan algunos pocos problemas cuando una palabra de cierta categoría gramatical sigue el patrón de otra categoría. Por ejemplo la palabra “azúcar” (que en el corpus aparece como “azucar”) sigue el patrón de terminar en -ar, como uno de los grupos de verbos en infinitivo. Por ello al procesar esta palabra, es identificada como verbo, y se observa que es llevada a su supuesta forma canónica “azuc”.

Los verbos irregulares constituyen también un problema, ya que el stemmer no los lleva a la forma canónica correcta. Es así que se obtienen los feature names “*da*” “*dan*” y “*dar*”, cuando sólo se esperaría este último, y sucede algo análogo con el verbo ser (se obtienen features names *“era”, “eram”,* “*es*”).

2.4 Resumen de opiniones con Clustering

Se eligió realizar el resumen de opiniones utilizando un *clustering KMeans*. Esta decisión se debe a la simplicidad y facilidad de implementación, a la vez que se obtienen resultados aceptables.

La idea de este clustering es agrupar oraciones que se encuentran "cercanas", mirando su representación como vectores. Con esta estrategia, se busca asociar en un cluster un conjunto de oraciones similares, que traten las mismas cosas.

Como desventaja hay que elegir la cantidad de clusters que se desean, mientras que otros algoritmos calculan automáticamente este valor.

Este algoritmo logra el agrupamiento mediante un mecanismo de convergencia iterativa. Se puede seleccionar si la semilla inicial debe ser aleatoria, será dada o la elegirá el algoritmo siguiendo una heurística preestablecida (opción por default, que fue la elegida).

La cantidad de cluster fue elegida empíricamente.

Al seleccionar 50 clusters, se agrupan comentarios positivos en un cluster, y negativos en otro, pero también hay clusters con graves problemas de similitud semántica. Por ejemplo, uno de los clusters agrupa el comentario "Fuimos a festejar mi cumpleaños y salimos contentos" e "y para peor no nos trajeron tips (a otras mesas sí les llevaron)". Se aumentó la cantidad de clusters deseados a 70 y 90, pero el problema persiste. Con un valor demasiado elevado, dado que son 309 oraciones, la partición es demasiado grande, por lo cual el agrupamiento pierde valor. Esto fue comprobado al utilizar valores de K (cantidad de clusters) como 100, 200 y 300.

Se decidió dejar el número de clusters en 50, ya que salvando algunos graves errores de asociación, muchas oraciones son semánticamente similares.

3 Problemas encontrados y mejoras sugeridas

3.1 Expresiones negadas

Se siguen observando problemas con las expresiones negativas, más aún si para su análisis se requiere hacer un análisis pragmático. Por ejemplo '*Son excelentes!*' y '*Nos fuimos realmente desconformes, no se si volveré a darles otra chance, ya que las criticas en su mayoría son buenas*' tiene una similitud alta (0.79) debido a los sinónimos "buenas" y "excelentes".

3.2 Problema de poner un feature por nombre distinto de synset

Para implementar la búsqueda de similitudes con sinónimos, se presenta el inconveniente de que si una palabra tiene muchos synset con nombres distintos asociados (por ejemplo *mozo* que tiene *mozo*, *fajín*, *chaval* y *adolescente*), ésta generará muchas columnas en la matriz que se obtiene a partir de las oraciones y el *countVectorizer*, y cualquier oración que contenga dicha palabra tendrá un valor 1 (o mayor) en cada una de dichas columnas. Esto causa que dos oraciones que compartan solamente esa palabra, serán indicadas como muy similares porque tendrán en común muchas columnas a la vez, sólo por una palabra, incluso dando como resultado mucho más similares que oraciones que tienen varias palabras en común. Este fenómeno distorsiona la medición, y hace que no sea real en dichos casos, ni totalmente efectiva.

En un principio se agregaban todos los distintos lemas de todos los synsets como tokens para los sustantivos y adjetivos en la tokenización. Se quiso disminuir el problema, y se buscó poner un token por cada nombre de synset en los sustantivos y adjetivos. Para mitigar aún más este inconveniente, se buscó disminuir la cantidad de columnas obtenidas en la matriz observando que varias palabras, por ejemplo *comida*, estaban en distintos synsets, muchos de los cuales coincidían en el inicio de su nombre (*alimento.n.01*, *alimento.n.02*, etc.). Se decidió parsear los nombres de los synsets, quedándose con el nombre hasta el primer punto, y construir una columna para cada uno de esos nombres truncados, quedándose con los distintos. Con esto se observó una mejora respecto a la implementación anterior (la similitud de oraciones no muy similares, que coincidían en pocas palabras de muchos synsets, bajó), pero igualmente el problema de que esa cantidad de columnas en la matriz distorsiona la evaluación persiste. Debería buscarse una mejor estrategia que solucione la situación de forma más precisa. Otra opción más detallista, que restringe el contexto de aplicación, es revisar manualmente las definiciones de cada synset, descartar aquellos que no se adecuan al contexto y entre los synsets restantes, elegir uno (elegir aquel que tenga la mayor probabilidad de aparecer en el contexto constituiría un criterio bastante bueno). Eventualmente también se podrían elegir unos pocos representativos, e incluso dentro de ellos asignar con una probabilidad dada para cada palabra a cuál de ellos pertenece. Si esta probabilidad toma en cuenta el contexto en el que aparece la palabra, los resultados serían aún más adecuados. Este tipo de análisis requiere un manejo mucho mayor de la semántica, y podría requerir análisis pragmáticos.

Por otro lado es apreciable que si dos oraciones usan la misma palabra tengan más similitud que dos oraciones que tienen sinónimos, ya que es más probable que hayan querido hablar sobre lo mismo.

Se pensó también por momentos elegir para cada palabra uno de los synsets o lemas según algún criterio. Por ejemplo, elegir uno al azar, o mejor aún, elegir el más común (pudiendo usar el método

count() de los Lemmas, pero este método no está incorporado en esta versión de WordNet.). El problema grave que tiene esta implementación es que si dos palabras que son sinónimos eligen lemas distintos como tokens, no serán reconocidos como similares bajo dicho sinónimo, lo cual disminuye el aprovechamiento del uso de los synsets.

3.3 Definición de un synset

Sería interesante poder contar con una herramienta que brinde todas las definiciones de los synsets, y no sólo las de algunos, para realizar un análisis más efectivo y también para poder realizar otras tareas de procesamiento de lenguaje, por ejemplo, un programa que ofrezca sinónimos al usuario, podría ofrecérselos según los distintos significados brindados. No se ha investigado si hay herramientas con dichas características de acceso libre para el idioma español. Esta mejora escapa a la realización en sí de este laboratorio.

3.4 Verbos irregulares

Los verbos irregulares constituyen un problema para ser llevados a su forma canónica. El *SpanishStemmer* no realiza correctamente esta acción, y este hecho resulta en una pérdida de calidad en el análisis. Muchos de los verbos más usados en español, tales como ser, estar e ir, son extremadamente irregulares, y esto deriva en que, por ejemplo, *"fuimos"* no sea llevado a su forma canónica *"ir"*.

3.5 Relaciones entre sustantivos y verbos

Para un trabajo más detallado sería interesante encontrar similitudes entre los verbos y sus sustantivos derivados. Por ejemplo "se come muy bien" y "la comida es muy buena" son frases semánticamente muy similares, donde *comer* y *comida* son palabras fuertemente relacionadas. Para analizar las oraciones del ejemplo, también es necesario tener el cuidado de reconocer *bien* y *bueno/a/os/as* como sinónimos.

Asimismo, puede ser de interés analizar las relaciones entre verbos y adjetivos derivados, y entre sustantivos y adjetivos emparejados.

3.6 Tagger

Para realizar un etiquetado más adecuado de las expresiones se debería hacer especial hincapié en el reconocimiento de adverbios, en contextos como *'medio crudo'* (ejemplo mencionado anteriormente). Para ello se debería analizar el contexto de una forma distinta, ya que los métodos empleados no fueron eficaces en este aspecto. La introducción de algunas reglas gramaticales (como el hecho de que la palabra *medio* antes de un adjetivo suele ser un adverbio) podrían facilitar esta tarea.

4 Observaciones y conclusiones

La utilización de sinónimos para encontrar similitudes entre oraciones es una buena estrategia conceptualmente, ya que permite mejorar significativamente las coincidencias que se dan a nivel semántico con palabras distintas.

Los resultados arrojados si bien no son malos, necesitan sen perfeccionados para considerar que la herramienta tiene un elevado nivel de aciertos. Un corpus de mayor tamaño podría ayudar a que el algoritmo obtenga mejores resultados, ya que se tendrían más datos para entrenar.

El taggeo obtenido es bueno, si bien se puede mejorar. El límite en la cantidad de expresiones regulares que se pueden utilizar para crear el *POS tagger* (100 expresiones) dificulta la construcción de una herramienta más completa. Además, el idioma Español está lleno de irregularidades que dificultan el procesamiento automático, por ejemplo en el caso de los sufijos que suelen representar una cierta categoría gramatical, pero hay palabras de otras categorías con las mismas terminaciones (como en el caso de los verbos terminados en *-ar* y el sustantivo *azúcar*).

En lo que respecta a la utilización de taggers con unigramas, bigramas y trigramas, se percibe que los taggers que utilizan bigramas y trigramas sin utilizar el backoff funcionan peor que el tagger con unigramas. Sin embargo éstos anidados con el tagger unigrama funcionan significativamente mejor que el tagger de unigrama solo.

El tema de los verbos irregulares dificulta el reconocimiento del *SpanishStemmer*, como ya fue mencionado. Se considera que mejorar este aspecto es muy importante para mejorar la comparación entre oraciones.

Respecto al resumen de opiniones automático, los resultados en general fueron buenos, pero se encontraron algunos agrupamientos para nada adecuados. Hay problemas que se pueden considerar inherentemente semánticos, es difícil saber también cuál de todas las acepciones de una palabra se quiso emplear.

Para realizar un resumen automático, faltaría seleccionar dentro de los conjuntos que agrupan oraciones similares, una o más oraciones representativas, y luego unir esas oraciones adecuadamente (una estrategia podría ser poner una oración a continuación de la otra). La decisión de cuántas oraciones representativas extraer de cada grupo y cuáles debería hacerse teniendo en cuenta la naturaleza de las similitudes de cada grupo, y conceptos clave (que se repitan en todas o muchas de las oraciones). Si el conjunto es muy “homogéneo”, con una oración representativa podría alcanzar, en cambio si existen mayores diferencias o el nivel de similitudes es bajo, podría ser necesario elegir más de una, o realizar un nuevo agrupamiento más adecuado.

En conclusión, el uso de sinónimos para encontrar similitudes y realizar resúmenes no solamente es una estrategia buena sino fundamental para poder abarcar los textos en lenguaje natural. La complejidad del idioma es muy grande, y el Español se caracteriza por una gran cantidad de excepciones y modificaciones posibles para construir palabras. El sistema implementado realiza de forma buena el trabajo pedido, pero está muy lejos de abarcar la inmensa complejidad del lenguaje natural. En este sentido se han intentado señalar los puntos a mejorar y posibles caminos para ofrecer un enfoque más cuidadoso y apropiado a la naturaleza del español.

5 Bibliografía y referencias

[1] Información de CountVectorizer**.** En línea: http://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.CountVectorizer.html. Última visita: Octubre 2013.

[2] Expresiones regulares en python. En línea: http://docs.python.org/2/library/re.html. Última visita: Octubre 2013.

[3] NLTK**.** En línea: http://nltk.org/. Última visita: Octubre 2013.

- Categorizing and tagging words http://nltk.org/book/ch05.html
- Tag package documentation http://nltk.org/api/nltk.tag.html

[4] Información de python**.** En línea: http://docs.python.org. Última visita: Octubre 2013.

[5] Información de NLTK**.** En línea: http://stackoverflow.com/questions/4867197/failed-loading-english-pickle-with-nltk-data-load. Última visita: Octubre 2013.

[6] Información de Unicode**.** En línea: http://stackoverflow.com/questions/10288016/usage-of-unicode-and-encode-functions-in-python. Última visita: Octubre 2013.

[7] Gramática del español – Nombres aumentativos y diminutivos**.** En línea: http://es.wikisource.org/wiki/Gramática_de_la_lengua_castellana_destinada_al_uso_de_los_americanos:_Capítulo_XII. Última visita: Octubre 2013.

[8] Etiquetas eagle (v2.0)**.** En línea: http://nlp.lsi.upc.edu/freeling/doc/tagsets/tagset-es.html. Última visita: Octubre 2013.

[9] Wikicorpus – Sitio oficial**.** En línea: http://www.lsi.upc.edu/~nlp/wikicorpus/. Última visita: Octubre 2013.

[10] Wordnet – Sitio oficial**.** En línea: http://wordnet.princeton.edu/. Última visita: Octubre 2013.

[11] Multilingual central repository**.** En línea: http://adimen.si.ehu.es/web/MCR. Última visita: Octubre 2013.

[12] Letra del laboratorio**.** En línea: https://eva.fing.edu.uy/mod/resource/view.php?id=33729. Última visita: Octubre 2013.

[13] Sufijos del español**.** En línea: http://es.wiktionary.org/wiki/Wikcionario:Lista_de_sufijos_en_el_espa%C3%B1ol. Última visita: Octubre 2013.

[14] Taggers**.** En línea: http://nltk.googlecode.com/svn/trunk/doc/howto/tag.html. Última visita: Octubre 2013.

[15] Clustering**.** En línea: http://scikit-learn.org/stable/modules/clustering.html. Última visita: Octubre 2013.