Escuela Técnica Superior de Ingenieros Informáticos

Universidad Polítecnica de Madrid

TweetSC: Corrector de texto para twitter

Trabajo Fin de Máster

Máster Universitario en Inteligencia Artificial

AUTOR: Javier Moreno Vega
TUTOR/ES: Óscar Corcho García y
Víctor Rodríguez Doncel

https://jmorenov.github.io/TweetSC/

2 de julio de 2018

RESUMEN

Extensión máxima de una página

SUMMARY

Extensión máxima de una página

Índice

Índice de figuras

Índice de cuadros

1. Introducción

1.1. Motivación

Los nuevos sistemas de comunicación como la mensajería instantánea, chats, redes sociales han generado un uso diferente de los idiomas en estos ámbitos, llamado lenguaje tipo chat [49]. Una de estas redes sociales y en la que este trabajo va a centrarse es Twitter. En esta red social predomina el Uso de emoticonos, repetición de vocales o eliminación de las mismas, uso abusivo de mayúsculas o asusencia, siglas de expresiones populares; lo que dificulta el análisis de los textos. Las ventajas que ofrece esta red social para investigar sobre ella son la cantidad de datos en tiempo real y su fácil acceso.

Uno de los principales problemas a la hora de analizar textos procedentes de las redes sociales son los errores gramaticales que suelen contener, así como la presencia de elementos propios de este tipo de foros que requieren de un procesamiento especial (i.e. hashtags, formas de mencionar a otros usuarios o emoticonos y expresiones habituales en las redes). Además, la limitación en el número de caracteres existente en Twitter la convierte en un caso singular dentro de las redes sociales, ya que los usuarios tienden a adaptar su forma de escribir a dicha limitación, omitiendo palabras y creando abreviaturas que dificultan el uso de herramientas genéricas de procesamiento del lenguaje, especialmente a la hora de realizar tareas como el Análisis de Sentimientos.

Los usuarios en twitter tienden a cometer errores tipográficos, abreviaciones, sustituciones fonéticas y estructuras no gramaticales en los mensajes cortos de texto, causando problemas en las herramientas de análisis. Esto es lo que se consideran palabras mal formadas y la detección de las palabras mal formadas es difícil debido al contexto ruidoso. El objetivo es normalizar estas palabra mal formadas.

A parte de un uso puramente de investigación, este tipo de trabajo también es beneficioso para un estudio de marcas o personas y sobre lo que las persones opinan sobre ello en las redes social, ya que sin el proceso de normalización y análisis de sentimientos estaríamos ante millones de datos que costarían mucho trabajo analizar de una forma automática.

1.2. Objetivos

El objetivo principal de este trabajo es la creación de un corrector que "normalice"tweets en español.
Para cumplir con este objetivo principal se ha dividido en los siguientes subobjetivos.

- Acceso a la API de Twitter para obtener tweets.
- Tokenizar tweets.

- Detectar entre los tokens las palabras fuera del vocabulario (Out-of-Vocabulary, OOV).
- Anotar el tipo de palabras OOV.
- Corregir palabras OOV.

Estos subojetivos se cumplirán con su implementación en un módulo software que además estará disponible en una aplicación web [44].

También ejecutaremos este corrector sobre un corpus de tweets disponible en [67] y compararemos nuestros resultados con los que se consiguieron en [3].

1.3. Resumen del documento

Esta memoria explica todo el trabajo desarrollado entrando en detalle en el estado del arte y el módulo software desarrollado. Primero se expone el estado del arte desarrollado sobre el tema de la corrección de textos, específicamente en twitter y en español. En segundo lugar se presenta el análisis realizado, atendiendo a: metodologías de desarrollo utilizadas, análisis de requisitos, solución propuesta. Posteriormente se plantea la implementación que se ha seguido entrando en detalle en cómo usar el software y en detalles técnicos. A continuación se presentaran los recursos utilizados y la evaluación sobre un corpus de tweets. Por último se muestran unas conclusiones, incluyendo un resumen del trabajo desarrollado y los objetivos conseguidos.

2. Estado del arte

2.1. Introducción

En la actualidad, la normalización lingüística de tweets [29] supone un campo de gran interés y en donde la mayoría de trabajos se han realizado sobre textos en inglés y pocos en español. Además no hay ningún trabajo en donde se incluya, dentro de la normalización de tuits, el estudio de los hashtags o etiquetas y los emoticonos, y su contexto.
Una introducción al tema de normalización de tuits es el artículo [14], donde se revisa el estado del arte en NLP sobre variantes SMS y tweets, y cómo la comunidad científica ha respondido por dos caminos: normalización y adaptación de herramientas.

2.2. Normalización

El modelo de canal ruidoso [62] ha sido tradicionalmente la primera aproximación a la normalización de textos. Supone que el texto mal formado es T y su forma normalizada es S, por lo que hay que encontrar: arg max $P(S|T)$, calculando arg max $P(T|S)P(S)$, P(S) es el modelo del lenguaje y $P(T|S)$ es el modelo de error. [8] caracterizan el modelo de error calculando el producto de operaciones de probabilidad en partes de cadenas de caracteres. [66] mejoraron el modelo incorporando información de la pronunciación. [11] modela el proceso de generación de texto a nivel de palabra para mensajes SMS considerando las abreviaturas grafémicas/fonéticas y los errores tipográficos involuntarios como transiciones de estado ocultas del modelo de Markov (HMM) y emisiones, respectivamente. [12] expandieron el modelo de error introduciendo inferencias de diferentes procesos de formación erróneos, de acuerdo con la distribución de errores muestreada.

Mientras el modelo de canal ruidoso es apropiado para normalización de textos, es difícil aproximar la normalización con exactitud, además estos métodos ignoran el contexto alrededor del OOV, el cual ayuda a resolver ambigüedades. La traducción automática estadística (SMT) se ha propuesto como un medio de normalización de texto sensible al contexto, al tratar el texto mal formado como el idioma de origen, y la forma estándar como el idioma de destino. Por ejemplo [5]. Normalización de textos como un problema de reconocimiento de voz [36]. [6] métodos de estado finitos combinando las ventajas de SMS y el modelo de canal ruidoso. [35] usan un enfoque de traducción automática con un preprocesador para la normalización sintáctica (en lugar de léxica).

El problema de estos trabajos anteriores es que requieren datos de entrenamiento anotados a gran escala, lo que limita su adaptabilidad a nuevos dominios o idiomas, mientras que los trabajos [71] y [29], no. Estos trabajos son una buena referencia en el campo de la normalización de tuits en inglés de forma no supervisada. En donde para detectar palabras fuera de diccionario (OOV) utilizan GNU aspell, y

los usuarios (@usuario), los hashtags y las URLs son excluidas de la normalización. La normalización tiene relación con los correctores de texto [53] pero difiere en que las palabras mal formadas en los mensajes de texto suelen ser intencionadas, para ahorrar caracteres, como identidad social, o debido a la convención en este subgénero de texto. La detección de las palabras mal formadas es difícil debido al contexto ruidoso. El objetivo es normalizar estas palabra mal formadas, además muchas palabras mal formadas son ambiguas y requieren el contexto para poder normalizarlas.

2.3. Adaptación de herramientas

En vez de adaptar el texto a herramientas de análisis otro de los caminos a seguir es adaptar las herramientas de análisis al texto. Destacan los trabajos de reconocimiento de voz [23] [52], reconocimiento de entidades [16] [56] [38], análisis gramatical [19], modelización de diálogos [55] y resumen automático de textos [63].

El reconocimiento de entidades nombradas (NER) es una tarea de extracción de información que busca localizar y clasificar en categorías predefinidas, como personas, organizaciones, lugares, expresiones de tiempo y cantidades, entidades encontradas en un texto. Las soluciones propuestas para NER suelen recaer en tres categorías: Basado en reglas [37], Basada en aprendizaje automático [18] [64] y Métodos híbridos [33]. Con la disponibilidad de datos anotados, Enron [42] y CoNLL03 [60] se han convertidos en los nuevos métodos dominantes. El estudio actual NER se centra principalmente en textos formales, de hecho, el estado del arte actual (CoNLL03) tiene un éxito del 90.8 % en textos formales y 45.8 % en tweets. En el contexto de los textos en Tweets, existe una dificultad en el reconocimiento de entidades nombradas debido a la falta de información y datos de entrenamiento.

El trabajo en el contexto de los textos de Twitter se puede dividir en tres categorías: NER en tweets, NER en no tweets y aprendizaje semi-supervisado para NER. El trabajo principal de NER sobre tweets es [16], en donde se anotan los tweets y se entrena el modelo con CRF. En cuanto a los trabajos de NER sobre no tweets: [37] utilizan reglas manuales para extraer entidades de tipos predefinidos, [73] utilizan HMM (Hidden Markov Model) mientras que [17] usa CRF. En la tercera categoría, aprendizaje semi-supervisado para NER, se encuentran los trabajos de [34] que utiliza un algoritmo de bootstrapping balanceado, [72] también utiliza un algoritmo de bootstrapping, [41] clusters de palabras, [9] aprende desde texto sin etiquetar y [28] introduce Latent Semantic Association (LSA) para NER. El trabajo más importante y actual de NER para tweets es [38] donde replantea el tema de reconocimiento de entidades nombradas en corpus de tuits. Combina un clasificador KNN con CRF (Conditional Random Fields).

La desambiguación léxica o etiquetado gramatical (POST) es una parte muy importante y útil en la tarea de normalización de textos ya que nos permite definir el

subconjunto de palabras debido a su categoría gramatical que con una probabilidad pueden ser la normalización de un OOV. Además un gran porcentaje de palabras en un texto son palabras que pueden ser asignadas a más de una clase morfológica, a más de un part-of-speech (PoS). Uno de los trabajos más importantes y probado para español es [65], este trabajo presenta un método de POST de ventana deslizante (SWPoST), asigna el part-of-speech de una palabra basado en la información que dan las palabras en una ventana fija de alrededor. Puede ser implementado como una máquina de estados finitos (Máquina de Mealy).

2.4. Normalización en español

Una introducción a la normalización de tuits en español es [3][2]. Este trabajo propuso en 2013 una tarea o competición en la que los participantes proponían soluciones de normalización de tweets. Los organizadores de la competición ofrecían dos datasets de tweets ya notados uno de desarrollo y otro para test, junto con un tercero que no era público y que era usado para la última evaluación.

Las soluciones ofreciddas por los participantes se pueden dividir en dos categorías, los que utilizan generación de candidatos junto un modelo del lenguaje, y los que utilizan transductores o FSTs (Finite State Transducers). El participante que mejor accurancy consiguió, Sistema RAE [21] con un 0.781, optó por la segunda categoría e implementó un siststem basado en FSTs para la tarea de normalización léxica de mensajes de Twitter en Español. El sistema desarrollado consiste en transductores que se aplican a tokens OOV. Los transductores implementan modelos de variación lingüística que generan conjuntos de candidatos acordes a un léxico. Un modelo estadístico del lenguaje se usa para obtener la secuencia de palabras más probable. El sistema tiene tres componentes principales que se aplican secuencialmente. Un analizador que ejecuta tokenización y análisis léxico sobre palabras en forma estándar y otras expresiones (números, fechas, ...). Un componente que genera palabras candidatas para los tokens OOV. Un modelo estadístico del lenguaje para obtener la mejor secuencia de palabras. Y finalmente un truecaser para capitalizar correctamente las palabras asignadas a los tokens OOV. El conjunto de confusión de un token OOV se genera aplicando el algoritmo de camino mínimo a la expresión: $W \circ E \circ L$. Donde W es el automata que representa el token OOV, E es un transductor de editado que genera todas las posibles variaciones de un token, y L es un conjunto de palabras objetivo. Dentro de esta categoría se encuentran los trabajos de la tarea: [1] en donde usan una batería de módulos para generar diferentes propuestas de corrección para cada palabra desconocida. La corrección definitiva se elige por votación ponderada según la precisión de cada módulo, [3] que además utiliza un modelo para el reconocimiento de voz para la generación de candidatos y [32] presentan dos estrategias basadas en FSTs una con reglas diseñadas manualmente y la otra automática.

Entre los participantes que optaron por la primera categoría destaca [57][69] que

usa reglas de preproceso, un modelo de distancias de edición adecuado al dominio y modelos de lengua para seleccionar candidatos de corrección según el contexto. Su arquitectura está formada por: preproceso basado en expresiones regulares y listas customizadas, generación de candidatos mediante una técnica de mínima de distancia de editado, ranking de candidatos mediante una combinación con pesos de la puntuación del modelo del lenguaje y la distancia de editado y la puntuación del modelo de lenguaje es n-grama utilizando la distancia Levenshtein. El sistema obtuvo resultados superiores a la media en la tarea. Una mejora a este trabajo por los mismos autores es [58] en donde utilizan un sistema basado en reglas para seleccionar los candidatos. Otros trabajos en esta categoría son: [22] que propone un sistema basado en [30], [68] y [47] que emplea técnicas de RAH (reconocimiento del habla) mediante la herramienta TENOR [45] junto con un modelo del lenguaje. Otro trabajo basado en la tarea de Tweet-Norm pero que no participó en ella es [10], ellos optaron por normalizar los OOV basándose en similaridad entre grafemas y fonemas; generan el conjunto de confusión (de candidatos) usando grafemas y fonemas, seguido de transductores aplicados mediante reglas para las palabras extranjeras y acentos, la selección de candidatos mediante un modelo del lenguaje con la herramienta Kenlm [31].

Fuera de estas dos categorías nos encontramos con los trabajos: [59] que utiliza conversiones basadas en reglas hasta una forma final normalizada. Después de recibir una lista con las posibles correcciones el sistema selecciona la más común acorde con una lista de palabras ordenada por frecuencia, [70] utilizan una lista de prioridad para los candidatos obtenidos y una tabla de frencuencias de palabrar para puntuarlos, [30] presentan una estrategia basada en búsquedas rápidas mediante una lista de frecuencias aprendida desde un corpus de tweets, [74] no generan candidatos simplemente selecciona palabras OOV y las corrigen con un corrector externo y [48] generan candidatos y seleccionan el mejor mediante una función de distancia. Una mejora a este último trabajo por parte de los autores fue [13] donde añaden un modulo de puntuación para la selección de candidatos.

Otros trabajos sobre normalización en español son [45] en donde se generan candidatos con indexación fonética y se seleccionan el candidato calculando la similaridad léxica junto con un modelo del lenguaje trigrama y [51]. Estos trabajos son principalmente sobre mensajes SMS, y no abordan la normalización de tuits en su conjunto. Dentro de la normalización en español existen otras tareas relacionadas como es la tokenización y aquí destaca el trabajo [24] que estudia la tokenización de textos SMS.

2.5. Word2Vec

Muchos sistemas y técnicas actuales de NLP tratan las palabras como unidades atómicas, no hay noción de similaridad entre palabras y son representadas como índices en un vocabulario, por ejemplo el modelo N-grama, para tratar de resolver

este problema aparecen las representaciones continuas de palabras. Las representaciones continuas de palabras entrenadas sobre corpus sin etiquetas son útiles para muchos trabajos de NLP. Muchos tipos diferentes de modelos han sido propuestos para estimar representaciones continuas de palabras, incluyendo Latent Semantic Analysis (LSA) y Latent Dirichlet Allocation (LDA). En este trabajo se centran en las representaciones distribuidas de palabras aprendidas por redes neuronales, ya que se demostró que su eficacia era considerablemente mejor que LSA para preservar regularidades lineales entre palabras, LDA además es computacionalmente caro en datasets grandes. Además se ha demostrado las redes neuronales basadas en modelos del lenguaje mejoran significativamente los modelos N-grama [7] [39] [61].

Los dos modelos de redes neuronales que destacan basados en modelos del lenguaje son: Feedforward Neural Net Language Model (NNLM) [7] y Recurrent Neural Net Language Model (RNNLM) que mejora algunas limitaciones de NNLM. El problema de estos modelos es que con grandes cantidades de datos son muy costotos computacionalmente. Para resolver este problema en el trabajo [40] desarrollado por Google se presentaron dos nuevos modelos de arquitecturas para calcular representaciones continuas de vectores de palabras a partir de grandes datasets, además crearon un framework de bibliotecas llamado Word2Vec [26]. El principal objetivo de este trabajo es introducir técnicas que puedan ser usadas para aprender vectores de palabras de gran calidad a partir de grandes datasets con millones de palabras y con millones de palabras en el vocabulario. Decidieron explorar modelos más simples que aunque no puedan representar los datos de forma tan precisa como las redes neuronales pero pueden ser entrenados con muchos más datos de forma más eficiente. Estos modelos son: Continuous Baf-of-Words model (CBOW) similar a NNLM, la capa oculta no-lineal se elimina y la capa de proyección es compartida por todas las palabras; y Continuous Skip-gram model similar a CBOW pero en vez de predecir la palabra actual basándose en el contexto intenta maximizar la clasificación de la palabra basándose en otra palabra de la misma frase.

La mayoría de las técnicas de representación continua de vectores de palabras representan cada palabra del vocabulario como un vector distinto, sin parámetros compartidos. En particular se ignora la estructura interna de las palabras lo que es una importante limitación en lenguajes ricos morfológicamente. Para intentar resolver este problema en el trabajo [54], desarrollado por Facebook [15] y llamado fastText, se propone un nuevo enfoque basado en el modelo skipgram [40] donde cada palabra se representa como una bolsa de caracteres n-gramas. Una representación de vector está asociada con cada caracter n-grama, las palabras se representan como la suma de estas representaciones. Al usar una representación de vector distinta para cada palabra, el modelo skipgram ignora la estructura interna de las palabras y en este nuevo trabajo se implementa una función de puntuación diferente para tener en cuenta esta información.

3. Solución propuesta

La solución propuesta y a la que hemos llamado TweetSC (Tweet Spell Checker) [44] se llegó a ella a partir de varios análisis y evaluaciones que se hicieron con diversas bibliotecas y algoritmos, y todos ellos se pueden encontrar en la solución final para su uso.

En la primera versión de nuestra solución se construyó un corrector de texto sencillo basándonos en el creado por Peter Norvig [50], el cuál utiliza un diccionario para seleccionar las palabras incorrectas y las corrige mediante el teorema de Bayes usando probabilidades. Se usa la fórmula: $argmax_{c \in candidates} P(c|w)$, que mediante el teorema de Bayes es equivalente a: $argmax_{c \in candidates} P(c)P(w|c)/P(w)$, y como P(w) es igual para cada candidato c: $argmax_{c \in candidates} P(c)P(w|c)$. Esta fórmula trata de seleccionar el candidato de probabilidad máxima para cada palabra. Para calcular la probabilidad se usan dos diccionarios, uno de palabras en Español y otro de nombres propios. Se utilizó esta primera versión como punto de partida para ir creando versiones más avanzadas.

El resultado final y por tanto nuestra versión definitiva consiste en un proceso iterativo sobre el tweet que se puede dividir en 6 fases: Tokenización, reglas de preproceso, detección de OOVs, generación de candidatos para cada OOV, ranking de candidatos y postproceso.

Además para convertir el sistema en uno más dinámico se ha desarrollado una aplicación web con acceso a la API de Twitter para obtener los tweets mediante querys introducidas en un formulario de nuestra aplicación web.

3.1. Tokenización

Cómo realizan los analizadores léxicos en los compiladores, en la primera fase de nuestro proceso se realiza una tokenización del texto o tweet, un tokenizador genera una salida compuesta de tokens o símbolos.

Para mejorar la versión inicial se hizo uso de la biblioteca Stanford NLP [27]. Esta biblioteca creada por Stanford NLP Group ofrece tanto etiquetado gramatical (POS Tagging o POST) cómo deteccion de etiquetas (Named Entity Recognition o NER), nosotros la hemos utilizado para esta fase de tokenización. Además de StanfordNLP para la tokenización hemos utilizado Freeling [20], también se ha añadido al sistema el análisis que ofrece freeling.

En esta primera fase se recibe como entrada el texto del tweet y genera una lista de tokens que pasaran a la siguiente fase.

3.2. Preprocess rules

Una vez que hemos obtenido todos los tokens de un tweet se aplican unas reglas de preproceso para normalizar palabras típicas de la red social, pictogramas, fonogramas, onomatopeyas, números, acrónimos, etc. Tras aplicar estas reglas a los tokens que las acepten, se crean OOVs con estos token, se anotan como variaciones y se eliminan de la lista de tokens para las fases siguientes. Los OOV generados se añaden a la lista final de OOV.

3.3. Detección de OOV

Esta fase tiene como elementos de entrada los tokens restantes de la fase anterior, y se ejecuta token por token el detector de OOV. Para detectarlos se aplican reglas y se van descartando los tokens que son URLs, usuarios de twitter, hashtag de twitter y fechas; los elementos restantes se comparan con tres diccionarios utilizados como recursos: diccionario de español, diccionario de inglés y diccionario de entidades.

Los token que se detecten dentro del diccionario de español se descartan como OOV, los que se detecten en el diccionario de inglés se anotan como NoEs (No español o ininteligible) y los que se detecten en el diccionario de entidades se anotan como Correct (palabras correspondientes a una entidad o un nuevo préstamo). Para el resto de token que no han sido aceptados en ninguna regla se crea una lista de OOV y son los que pasaran a la siguiente fase pudiendo al final ser anotados como Variation o NoEs.

3.4. Generación de candidatos OOV

La generación de candidatos se puede considerar la primera fase de la corrección en sí, ya que sólo se trabaja con OOV a los que se va a buscar una corrección, en ese caso candidatos para ese OOV. Esta fase tiene como entrada la lista de OOV que no han sido etiquetados en la fase anterior, es decir, los que pueden ser Variation o NoEs. Para cada OOV se generarán una lista de candidatos con diferentes métodos. Los métodos que hemos utilizado con los nombres que hemos definido son: LevenshteinFST, Metaphone, L_L, FastTest.

- **LevenshteinFST:** Método que utiliza un FST (Finite State Transducers) para generar variaciones en el OOV con un máximo de distancia de editado según la distancia Levenshtein.
- **Metaphone:** Método que utiliza el algoritmo del metáfono en español [46]. Su funcionamiento consiste en generar los fonemas de todos los diccionarios que hemos utilizado para después comparar el fonema del OOV y seleccionar los de mayor similaridad con los fonemas de los diccionarios.
- **L_L:** Los candidatos generados con este método son las palabras aceptadas por el lenguaje L(_L)+.

- **FastText:** Este método hace uso de la biblioteca fastText [15], a partir de un modelo generado mediante redes neuronales y representando las palabras de forma continua. Los OOV se convierten a vectores de palabras y se comparan con los vectores del modelo generado para obtener los candidatos más parecidos a partir del la similaridad del coseno.

Estos métodos se ejecutan sobre todos los OOV y generan una lista de candidatos que pasarán a la siguiente fase.

3.5. Ranking de candidatos

El ranking de candidatos es la fase que define la corrección de un OOV, o si no tiene corrección (se anota como NoEs). Para generar este ranking hemos utilizado dos marcadores, uno un modelo del lenguaje N-Gram mediante la biblioteca OpenNLP [4] y el otro la distancia de editado Damerau-Levenshtein.

3.6. Postprocess

4. Implementación

En esta sección se pretende explicar toda la implementación software que se ha realizado de nuestra solución. Primero se realizará una introducción comentando lenguajes y herramientas utilizadas. Segundo unos instrucciones sobre dónde encontrar y cómo utilizar nuestro software . Y por último la documentación generada del código fuente.

4.1. Introducción

La implementación se ha realizado en tres módulos o componentes, por una parte tenemos la biblioteca con la funcionalidad necesaria para corregir textos de twitter y acceder a su API, después un modulo que es la aplicación web y por último otro que ofrece funcionalidad para utilizar la biblioteca desde línea de comandos.

El lenguaje utilizado en todo el proyecto ha sido Java y hemos hecho uso de Google Cloud Engine [25] para que la aplicación web esté disponible para cualquier usuario [43]. Además de Gradle para compilar el código.

4.2. Cómo usarlo

Para utilizar nuestro software primero es necesario tener instalado Git y Java 1.8. Después realizar un clonado del código fuente:

```
git clone https://github.com/jmorenov/TweetSC
```

Posteriormente se compila el código con gradle:

```
gradlew clean build && gradlew createJar
```

y para ejecutarlo:

```
java -jar org.jmorenov.tweetsc-0.3.0-alpha.jar -text "Texto de prueba"
```

4.3. Documentación del código

(Javadoc del código)

4.4. Aplicación web

(Capturas de pantalla)

5. Recursos utilizados

Diccionario de inglés: https://github.com/dwyl/english-words

6. Evaluación

Esta sección describe la evaluación que se ha hecho de la solución propuesta. Primero se define la metodología utilizada para evaluar la solución, en segundo lugar explicamos el corpus utilizado como datos de entrada, posteriormente el gold standard actual y por último los experimentos que hemos realizada con sus resultados.

6.1. Metodología

La metodología que hemos seguido ha sido la misma que en la tarea compartida Tweet-Norm 2013 [67]. Ellos utilizan como medida de evaluación la corrección de errores, sólo tiene en cuenta si la forma propuesta es correcta en base a los criterios: **correcta** si la forma original era correcta y no se ha realizado ninguna normalización o si la forma original era incorrecta y el candidato seleccionado es el correcto; **errónea** en cualquier otro caso. La evaluación final es el número de decisiones realizadas correctamente sobre el total de palabras OOV.

6.2. Corpus

El corpus utilizado es el mismo que en la tarea compartida Tweet-Norm 2013 [67], en donde utilizan dos subconjuntos uno de desarrollo con 500 tweets y otro de evaluación con 600 tweets.

6.2.1. Gold Standard

Nuestro gold standard ha sido el sistema propuesto RAE [21] en Tweet-Norm 2013 [67] donde consiguieron un resultado de 0.781 de precisión. Su sistema se basa en trasductores de estados finitos con pesos.

6.3. Experimentos

(Experimentos realizados)

7. Conclusiones

El proyecto realizado está compuesto de una parte de investigación, cómo se demuestra con el estado del arte y las diferentes soluciones que se han ido realizando hasta llegar a nuestra solución final. Además de la otra parte de desarrollo e implementación de software, ofreciéndolo para todos en código abierto y en una aplicación web [44].

Este desarrollo software se ha dividido en tres componentes o módulos:

- TweetSCCore: Núcleo del proyecto con la funcionalidad para corregir textos de twitter.
- TweetSCWeb: Aplicación web para corregir textos.
- TweetSCExecutable: Ejecutable java para corregir textos desde línea de comandos.

Se puede concluir que nuestro objetivo era construir un corrector de texto para twitter (sección 1.2) y se ha conseguido cómo se ha demostrado en las secciones 3, sección 4 y 5.

8. Líneas Futuras

Las líneas futuras son muy ámplias ya que estamos en un tema bastante reciente, sobre todo en español, y se los resultados se pueden mejorar de bastantes formas.

ANEXOS

Glosario de términos

- **Modelo (estadístico) del lenguaje:** Un modelo estadístico del lenguaje es una distribución de probabilidad sobre secuencias de palabras. Un tipo de modelo del lenguaje es el unigrama, también se suele llamar modelo de bolsa de palabras. La dispersidad en los datos es un problema al construir modelos del lenguaje. La secuencia de palabras más probable puede no aparecer en los datos de entrenamiento. Una solución es realizar la suposición de que la probabilidad de una palabra sólo depende de las n palabras previas. Esto es conocido como el modelo n-grama, unigrama cuando n=1. Los modelos del lenguaje neuronales o modelos del lenguaje continuos: modelo del lenguaje Skip-gram, base de word2vec.

- **Modelo del lenguaje N-grama:** Un modelo de n-grama es un tipo de modelo probabilístico que permite hacer predicción estadística del próximo elemento de cierta secuencia de elementos sucedida hasta el momento. Un modelo de n-grama puede ser definido por una cadena de Márkov de orden n-1. Predice x_i basándose en los n elementos anteriores.

- **Cadena de Márkov:** En la teoría de la probabilidad, se conoce como cadena de Márkov o modelo de Márkov a un tipo especial de proceso estocástico discreto en el que la probabilidad de que ocurra un evento depende solamente del evento inmediatamente anterior. En matemáticas se define como un proceso estocástico discreto que cumple con la propiedad de Márkov, es decir, si se conoce la historia del sistema hasta su instante actual, su estado presente resume toda la información relevante para describir en probabilidad su futuro.

- **Proceso de Márkov:** Fenómeno aleatorio dependiente del tiempo para el cual se cumple la propiedad de Márkov. Frecuentemente el término cadena de Márkov se usa para dar a entender que un proceso de Márkov tiene un espacio de estados discreto (infinito o numerable).

- **Modelo oculto de Márkov:** Un modelo oculto de Márkov (Hidden Markov Model, HMM) es un modelo estadístico en el que se asume que el sistema a modelar es un proceso de Márkov de parámetros desconocidos. El objetivo es determinar los parámetros desconocidos (u ocultos) de dicha cadena a partir de los parámetros observables. Un HMM se puede considerar como la red bayesiana más simple.

- **Etiquetado gramatical:** El etiquetado gramatical (part-of-speech tagging, POS tagging o POST) se considera el proceso de asignar a cada palabra de un texto su categoría gramatical. Las soluciones se pueden dividir en dos grandes grupos: aproximaciones lingüísticas basadas en un conjunto de reglas establecidas manualmente por expertos aprendidas de forma (semi)automática, y

las aproximaciones de aprendizaje automático que usan textos, generalmente anotados, para establecer los modelos. Además se pueden encontrar aproximaciones híbridas que combinan ciertos aspectos de las anteriores.

Referencias

[1] Alicia Ageno, Pere R. Comas, Lluís Padró, and Jordi Turmo. The talp-upc approach to tweet-norm 2013, 2013.

[2] Iñaki Alegria, Nora Aranberri, Pere R. Comas, Víctor Fresno, Pablo Gamallo, Lluis Padró, Iñaki San Vicente, Jordi Turmo, and Arkaitz Zubiaga. Tweetnorm: a benchmark for lexical normalization of spanish tweets, 2015.

[3] Iñaki Alegria, Nora Aranberri, Víctor Fresno, Pablo Gamallo, Lluis Padró, Iñaki San Vicente, Jordi Turmo, and Arkaitz Zubiaga. Introducción a la tarea compartida tweet-norm 2013: Normalización léxica de tuits en español, 2013.

[4] Apache. Opennlp. `http://opennlp.apache.org`.

[5] AiTi Aw, Min Zhang, Juan Xiao, and Jian Su. A phrase-based statistical model for sms text normalization, 2009.

[6] Richard Beaufort, Sophie Roekhaut, Louise-Amelie Cougnon, and Cedrick Fairon. A hybrid rule/model-based finite-state framework for normalizing sms messages, 2002.

[7] Y. Bengio, R. Ducharme, and P. Vincent. A neural probabilistic language model, 2003.

[8] Eric Brill and Robert C. Moore. An improved error model for noisy channel spelling correction, 2000.

[9] Peter F. Brown, Peter V. deSouza, Robert L. Mercer, Vincent J. Della Pietra, and Jenifer C. Lai. Class based n-gram models of natural language, 1992.

[10] Jhon Adrián Cerón-Guzmán and Elizabeth León-Guzmán. Lexical normalization of spanish tweets, 2016.

[11] Monojit Choudhury, Rahul Saraf, Vijit Jain, Animesh Mukherjee, Sudeshna Sarkar, and Anupam Basu. Investigation and modeling of the structure of texting language, 2007.

[12] Paul Cook and Suzanne Stevenson. An unsupervised model for text message normalization, 2009.

[13] J.M. Cotelo, F.L. Cruz, J.A. Troyano, and F.J. Ortega. A modular approach for lexical normalization applied to spanish tweets, 2015.

[14] Jacob Eisenstein. What to do about bad language on the internet, 2013.

[15] Facebook. fasttext. `https://fasttext.cc/`.

[16] Tim Finin, Will Murnane, Anand Karandikar, Nicholas Keller, Justin Martineau, and Mark Dredze. Annotating named entities in twitter data with crowd-sourcing, 2010.

[17] Jenny Rose Finkel, Trond Grenager, and Christopher Manning. Incorporating non-local information into information extraction systems by gibbs sampling, 2005.

[18] Jenny Rose Finkel and Christopher D. Manning. Nested named entity recognition, 2009.

[19] Jennifer Foster, Ozlem Cetinoglu, Joachim Wagner, Joseph Le Roux, Joakim Nivre, Deirdre Hogan, and Josef van Genabith. From news to comment: Resources and benchmarks for parsing the language of web 2.0, 2011.

[20] Freeling. Freeling. `http://nlp.lsi.upc.edu/freeling/`.

[21] Pablo Gamallo, Marcos García, and Santiago Fernández-Lanza. Word normalization in twitter using finite-state transducers, 2013.

[22] Pablo Gamallo, Marcos García, and José Ramon Pichel. A method to lexical normalisation of tweets, 2013.

[23] Kevin Gimpel, Nathan Schneider, Brendan O'Connor, Dipanjan Das, Daniel Mills, Jacob Eisenstein, Michael, Heilman, Dani Yogatama, Jeffrey Flanigan, and Noah A. Smith. Part-of-speech tagging for twitter: annotation, features, and experiments, 2011.

[24] Jose M. Gomez-Hidalgo, Andrés A. Caurcel-Díaz, and Yovan Iñiguez del Rio. Un método de análisis de lenguaje tipo sms para el castellano, 2013.

[25] Google. Google cloud engine. `https://cloud.google.com/`.

[26] Google. Word2vec. `https://github.com/deeplearning4j/deeplearning4j`.

[27] Stanford NLP Group. Stanfordnlp core. `https://stanfordnlp.github.io/CoreNLP/`.

[28] Honglei Guo, Huijia Zhu, Zhili Guo, Xiaoxun Zhang, Xian Wu, and Zhong Su. Domain adaptation with latent semantic association for named entity recognition, 2009.

[29] Bo Han and Timothy Baldwin. Lexical normalisation of short text messages: Makn sens a twitter, 2011.

[30] Bo han, Paul Cook, and Timothy Baldwin. unimelb: Spanish text normalisation, 2013.

[31] K. Heafield. Faster and smaller language model queries, 2011.

[32] Mans Hulden and Jerid Francom. Weighted and unweighted transducers for tweet normalization, 2013.

[33] Martin Jansche and Steven P. Abney. Information extraction from voicemail transcripts, 2002.

[34] Jing Jiang and ChengXiang Zhai. Instance weighting for domain adaptation in nlp., 2007.

[35] Joseph Kaufmann and Jugal Kalita. Syntactic normalization of twitter messages, 2010.

[36] Catherine Kobus, Franois Yvon, and Graldine Damnati. Transcrire les sms comme on reconnat la parole, 2008.

[37] George R. Krupka and Kevin Hausman. Isoquest: Description of the netowltm extractor system as used in muc-7., 1998.

[38] Xiaohua Liu, Shaodian Zhang, Furu Wei, and Ming Zhou. Recognizing named entities in tweets, 2011.

[39] T. Mikolov, A. Deoras, S. Kombrink, L. Burget, and J. Cernocký. Empirical evaluation and combination of advanced language modeling techniques, 2011.

[40] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space, 2013.

[41] Scott Miller, Jethran Guinness, and Alex Zamanian. Name tagging with word clusters and discriminative training, 2004.

[42] Einat Minkov, Richard C. Wang, and William W. Cohen. Extracting personal names from email: applying named entity recognition to informal text, 2005.

[43] Javier Moreno. Tweetsc spell checker app. `https://jmorenov.github.io/TweetSC/`.

[44] Javier Moreno. Tweetsc web. `https://tweetsc.github.io`.

[45] Mosquera, Alejandro, Elena Lloret, and Paloma Moreda. Towards facilitating the accessibility of web 2.0 texts through text normalisation, 2012.

[46] Alejandro Mosquera. Algoritmo del metáfono. `https://github.com/amsqr/Spanish-Metaphone/blob/master/phonetic_algorithms_es.py`.

[47] Alejandro Mosquera and Paloma Moreda. Dlsi en tweet-norm 2013: Normalizacion de tweets en español, 2013.

[48] Juan M. Cotelo Moya, Fermín L Cruz, and Jose A. Troyano. Resource-based lexical approach to tweet-norm task, 2013.

[49] Forsyth Eric N. and Craig H. Martell. Lexical and discourse analysis of online chat dialog, 2007.

[50] Peter Norvig. How to write a spelling corrector. `http://norvig.com/spell-correct.html`, 2007.

[51] Jesús Oliva, José I. Serrano, María D. Del Castillo, and Angel Iglesias. Sms normalization: combining phonetics, morphology and semantics, 2011.

[52] Olutobi Owoputi, Brendan O'Connor, Chris Dyer, Kevin Gimpel, Nathan Schneider, and Noah A. Smith. Improved part-of-speech tagging for online conversational text with word clusters, 2013.

[53] James L. Peterson. Computer programs for detecting and correcting spelling errors., 1980.

[54] Bojanowski Piotr, Grave Edouard, Joulin Armand, and Mikolov Tomas. Enriching word vectors with subword information, 2017.

[55] Alan Ritter, Colin Cherry, and Bill Dolan. Unsupervised modeling of twitter conversations, 2010.

[56] Alan Ritter, Sam Clark, Mausam, and Oren Etzioni. Named entity recognition in tweets: an experimental study, 2011.

[57] Pablo Ruiz, Montse Cuadros, and Thierry Etchegoyhen. Lexical normalization of spanish tweets with preprocessing rules, domain-specific edit distances, and language models, 2013.

[58] Pablo Ruiz, Montse Cuadros, and Thierry Etchegoyhen. Lexical normalization of spanish tweets with rule-based components and language models, 2014.

[59] Arturo Montejo Ráez, M. Carlos Diaz Galiano, Eugenio Martíne Cámara, M. Teresa Martín Valdivia, Miguel A. García Cumbreras, and L. Alfonso Ureña López. Sinai at twitter-normalization 2013, 2013.

[60] Erik F. Tjong Kim Sang and Fien De Meulder. Introduction to the conll-2003 shared task: language independent named entity recognition, 2003.

[61] H. Schwenk. Continuous space language models, 2007.

[62] Claude Elwood Shannon. A mathematical theory of communication, 1948.

[63] Beaux Sharifi, Mark-Anthony Hutton, and Jugal Kalita. Summarizing microblogs automatically, 2010.

[64] Sameer Singh, Dustin Hillard, and Chris Leggetter. Minimally-supervised extraction of entities from text advertisements, 2010.

[65] Enrique Sánchez-Villamil, Mikel L. Forcada, and Rafael C. Carrasco. Unsupervised training of a finite-state sliding-window part-of-speech tagger, 2004.

[66] Kristina Toutanova and Robert C. Moore. Pronunciation modeling for improved spelling correction, 2002.

[67] Tweet-Norm. Tweet-norm. `http://komunitatea.elhuyar.eus/tweet-norm/`.

[68] Xabier Saralegi Urizar and Iñaki San Vicente Roncal. Elhuyar at tweetnorm 2013, 2013.

[69] Sistema Vicomtech. Sistema vicomtech. `https://github.com/pruizf/tweet-norm-es`.

[70] Jesús Vilares, Miguel A. Alonso, and David Vilares. Prototipado rápido de un sistema de normalización de tuits: Una aproximación léxica, 2013.

[71] Casey Whitelaw, BenHutchinson, Grace Y. Chung, and Gerard Ellis. Using the web for language independent spellchecking and autocorrection, 2009.

[72] Dan Wu, Wee Sun Lee, Nan Ye, and Hai Leong Chieu. Domain adaptive bootstrapping for named entity recognition, 2009.

[73] GuoDong Zhou and Jian Su. Named entity recognition using an hmm-based chunk tagger., 2002.

[74] Óscar Muñoz-García, Silvia Vázquez, and Nuria Bel. Exploiting web-based collective knowledge for micropost normalisation, 2013.