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/

22 de junio 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 [**?**]. 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 estariamos 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.

- Corregir tweets (palabras y gramaticalmente)
- Procesar emoticonos deduciendo su significado en el contexto de análisis de sentimientos

- Expandir hashtags
- Procesar las conversaciones de los tweets, así como las URLs o imágenes para añadir contexto

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

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.

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 [**?**] 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 [**?**], 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 [**?**] 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. [**?**] caracterizan el modelo de error calculando el producto de operaciones de probabilidad en partes de cadenas de caracteres. [**?**] mejoraron el modelo incorporando información de la pronunciación. [**?**] 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. [**?**] 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 [**?**]. Normalización de textos como un problema de reconocimiento de voz [**?**]. [**?**] métodos de estado finitos combinando las ventajas de SMS y el modelo de canal ruidoso. [**?**] 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 el trabajo [**?**] no. Este trabajo es una buena referencia en el campo de la normalización de tuits en inglés. 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 [?] 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 [?]; [?], reconocimiento de entidades [?]; [?]; [?], análisis gramatical [?], modelización de diálogos [?] y resumen [?]. El trabajo [?] sobre NER (reconocimiento de entidades) replantea el tema de reconocimiento de entidades nombradas en corpus de tuits. Combina un clasificador KNN con CRF (Conditional Random Fields).

2.4. Normalización en español

Una introducción a la normalización de tuits en español es [?][?]. Utiliza la herramienta Freeling [?] para detectar palabras OOV. Uno de los sistemas de normalización de tuits en español, que participó en Tweet-Norm 2013 [?], es [?][?], 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. El sistema obtuvo resultados superiores a la media en la tarea [?][?]. Una mejora a este trabajo por los mismos autores es [?]. El trabajo que mejores resultados obtuvo en Tweet-Norm 2013 fue RAE [?], consiste en transductores que se aplican a tokens OOV. Los transductores implementan modelos lingüísticos de variación que generan conjuntos de candidatos de acuerdo con un léxico. Se usa un modelo de lenguaje estadístico para obtener la secuencia de palabras más probable. En el trabajo [?] hace uso de una combinación de varios "módulos expertos" independientes, cada uno especializado en una tarea concreta de la normalización de tuits, en lugar de centrarse en una sola técnica. En este trabajo además realiza un estado del arte actual de la normalización de tuits y en concreto para el idioma español. Otros trabajos sobre normalización en español son [?] [?] [?] principalmente sobre mensajes SMS, pero que no abordan la normalización de tuits en su conjunto.

2.5. Análisis de sentimientos

Un campo muy relacionado con la normalización de tuits es el análisis de sentimientos y un trabajo que realiza un estudio sobre técnicas de análisis de sentimientos de tuits en español es [?]. El trabajo [?] se centra en una técnica Naive-Bayes para el análisis de sentimientos en tuits en español.

3. Solución propuesta

La solución propuesta y a la que hemos llamado TweetSC (Tweet Spell Checker) [**?**] 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 [**?**], 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 e ir creando versiones más avanzadas.

Para mejorar la versión inicial se hizo uso de la biblioteca Stanford NLP [**?**]. Esta biblioteca creada por Stanford NLP Group ofrece tanto etiquetado gramatical (POS Tagging o POST) cómo deteccion de etiquetas (Named Entity Recognition o NER), y ambas funcionalidades fueron usadas para nuestra solución.

Además se ha utilizado la API ofrecida por IBM [**?**] para análisis de lenguaje y cómo con la biblioteca de Stanford la utilizamos tanto para POST como NER.

Tras utilizar estas dos tecnologías en la primera fase llamada preproceso, dónde se seleccionan las palabras incorrectas y se generan los candidatos, utilizaremos los Modelos de Markov (MMO) y n-gramas para seleccionar el candidato final para cada palabra incorrecta.

Cómo se utilizan varias tecnologías y algoritmos en las diferentes fases se ha utilizado un estudio y evaluación de cuál es el mejor para la solución final.

Para facilitar el uso de nuestra solución con tweets de twitter se ha definido una conexión con su API para acceder a los mismos.

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 [**?**] para que la aplicación web esté disponible para cualquier usuario [**?**]. 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)

5. 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.

5.1. Metodología

La metodología que hemos seguido ha sido la misma que en la tarea compartida Tweet-Norm 2013 [**?**]. 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.

5.2. Corpus

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

5.2.1. Gold Standard

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

5.3. Experimentos

(Experimentos realizados)

6. 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 [**?**].

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.

7. 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