Feature extraction for sentiment analysis on twitter data with spanish language

Victor Muñiz
Research Center in Mathematics.
Monterrey, Mexico.
Introduction

Sentiment Analysis focuses on automatically identifying whether a text expresses a positive, negative or neutral opinion about some topic.
Among all virtual opinion platforms, Twitter has become the most popular for sentiment analysis due to several reasons:

- Availability of information
- Large amount of data
- Constant update
- Worldwide available
Among all virtual opinion platforms, Twitter has become the most popular for sentiment analysis due to several reasons:

- Lot of applications
  - Opinion based marketing
  - Online ranking
  - Government and politics
  - Official statistics
  - Among many others...
One of the most popular techniques for text classification is the **Bag of Words** (Joachims, 1998), which constructs a Term Document Matrix based on term frequencies.

<table>
<thead>
<tr>
<th>$doc_1$</th>
<th>term$_1$</th>
<th>term$_2$</th>
<th>...</th>
<th>term$_N$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>3</td>
<td>0</td>
<td>...</td>
<td>1</td>
</tr>
<tr>
<td>$doc_2$</td>
<td>1</td>
<td>5</td>
<td>...</td>
<td>0</td>
</tr>
<tr>
<td>$doc_3$</td>
<td>2</td>
<td>8</td>
<td>...</td>
<td>3</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td></td>
<td>...</td>
</tr>
<tr>
<td>$doc_m$</td>
<td>7</td>
<td>1</td>
<td>...</td>
<td>5</td>
</tr>
</tbody>
</table>
However, on twitter data, the application of this (or any) technique is not straightforward:

- Short text
- Misspellings
- Abbreviations and non-standard contractions
- Emoticons, hashtags
- Unbalanced classes
Standard preprocessing techniques on twitter data are not enough, because generally we have variations of words with the same meaning:

\[
\text{pseudo-estudiantes} = \text{pseudoestudiantes} = \text{seudoestudiantes} = \text{seudestudiantes} \\
\text{separados} = \text{separa2} \\
\text{siempre} = \text{sienpre} = \text{100pre}
\]

This problem causes sparse Term Document Matrix

Bag of words it’s not enough. We need to incorporate contextual (apriori) information

**The challenge is to extract the main features of the tweet, which give us insights of the sentiment (polarity) of the text**
There is a lot of work on both feature extraction and classification for tweets, however, the vast majority are focused on English text.

Some previous work on lexical normalization of Spanish text has been done (Mosqueda & Moreda, 2012), however, there are important differences between countries and regions, even in the same language. This must be taken into account.

The objective of our work, is to implement a normalization method for Spanish text by using kernel-based methods, in order to obtain important features which can be used as input for a classification method.
Preprocessing and normalization
Data: We obtained and manually classify tweets from the API (https://dev.twitter.com/) according to some specific topics (i.e., convenience stores, cellphone services, etc).

Standard text preprocessing:

- Convert to lowercase
- Remove stopwords in spanish according to the list given by Martin Porter’s *Snowball stemming project* http://snowball.tartarus.org/. We add some words relative to the topic.
- Remove special characters: URL’s, @, RT, _, -, :, among others
Normalization

- Remove repeated characters and excess of white spaces
- Emoticon substitution according to the list: en.wikipedia.org/wiki/List_of_emoticons.

For instance:

- :-) emoticon-positivo
- :) emoticon-positivo
- :o) emoticon-positivo
- :c) emoticon-positivo
- :-D emoticon-muy-positivo
- X-D emoticon-muy-positivo
- >:[ emoticon-negativo
- =( emoticon-negativo
- :-[ emoticon-negativo
- :-|| emoticon-muy-negativo
- >:( emoticon-muy-negativo
- :: emoticon-neutral
The normalisation process consists on

1. Detection of non-conventional words
2. Substitution with similar words, (hopefully the correct ones in terms of the linguistic meaning)
Normalization

Detection of non-conventional words

- We used Aspell (http://aspell.net/) with a spanish dictionary, and we added extra terms, such as cities and localities from Mexico and other ones relative to the topic.
- For each word in the preprocessed tweet, we did a search with the Aspell API, and if it does not appear, we consider the options given by Aspell.
- Very often, the top ranked suggestion by Aspell is not the best choice.
Normalization

Detection of non-conventional words

Consider pseudoestudiantes

- [1] "pseudo" "estudiantes" "pseudo-estudiantes"
- [4] "predestinares" "predestines" "predestinases"
- [7] "predestinareis" "predestinase" "predestinar"
- [10] "predestinas" "predestinasteis" "predestinaste"
- [13] "predestinis" "sudestada" "predestinara"
- [16] "predestinars" "predestinaseis" "sudestadas"
- [19] "predestinis" "predestinadas" "predestinados"
- [22] "predestinabas" "predestinamos"

- We need to choose the appropriate word from the suggestions.
Kernel methods and “string kernels”. Let \( x, z \in \mathcal{X} \) (input space). Consider the kernel function:

\[
k(x, z) = \langle \phi(x), \phi(z) \rangle
\]

where \( \phi \) is a map:

\[
\phi : x \in \mathcal{X} \mapsto \phi(x) \in \mathcal{H} \text{ (feature space)}
\]

Kernel trick (Schölkopf and Smola, 2002)
Normalization


Let $s$ to be a substring. The mapping to the feature space is given by

$$
\phi_s(x) = \sum_{s \in x} \lambda^{L(s_x)},
$$

where $\lambda \in (0, 1)$ is a weight and $L(s_x)$ is the length of the substring $s$ into the document $x$.

**Example:** Consider $s = \text{car}$:

if $x = \text{“cara”}$, then $L(s_x) = 3$ (cara).

$$
\phi_s(x) = \lambda^3,
$$

if $x = \text{“cuarto”}$, then $L(s_x) = 4$ (cuarto).

$$
\phi_s(x) = \lambda^4.
$$

Let $s$ to be a substring. The mapping to the feature space is given by

$$\phi_s(x) = \sum_{s \in x} \lambda^{L(s_x)},$$

where $\lambda \in (0, 1)$ es a weight and $L(s_x)$ is the length of the substring $s$ into the document $x$.

Example: Consider $s = \text{car}$:

if $x = \text{“cara”}$, then $L(s_x) = 3$ ($\text{cara}$).

$$\phi_s(x) = \lambda^3,$$

if $x = \text{“cuarto”}$, then $L(s_x) = 4$ ($\text{cuarto}$)

$$\phi_s(x) = \lambda^4.$$
**Normalization**


Let $s$ to be a substring. The mapping to the feature space is given by

$$\phi_s(x) = \sum_{s \in x} \lambda^{L(s_x)},$$

where $\lambda \in (0, 1)$ is a weight and $L(s_x)$ is the length of the substring $s$ into the document $x$.

**Example:** Consider $s = \text{car}$:

- if $x =$ “cara”, then $L(s_x) = 3$ (cara).
  $$\phi_s(x) = \lambda^3,$$

- if $x =$ “cuarto”, then $L(s_x) = 4$ (cuarto)
  $$\phi_s(x) = \lambda^4.$$
Normalization

The kernel (dot product) between documents $x$ and $y$ is given by

$$k_n(x, y) = \sum_{s \in \Sigma^n} \sum_{s \subseteq x} \sum_{s \subseteq y} \lambda^{L(s_x)+L(s_y)},$$

where $\Sigma^n$ is the set of all substrings of size $n$ from a finite alphabet $\Sigma$.

**Example:** Consider the words cat, car, bat and bar with $|s| = 2$:

<table>
<thead>
<tr>
<th></th>
<th>c-a</th>
<th>c-t</th>
<th>a-t</th>
<th>b-a</th>
<th>b-t</th>
<th>c-r</th>
<th>a-r</th>
<th>b-r</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\phi$ (cat)</td>
<td>$\lambda^2$</td>
<td>$\lambda^3$</td>
<td>$\lambda^2$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$\phi$ (car)</td>
<td>$\lambda^2$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>$\lambda^3$</td>
<td>$\lambda^2$</td>
<td>0</td>
</tr>
<tr>
<td>$\phi$ (bat)</td>
<td>0</td>
<td>0</td>
<td>$\lambda^2$</td>
<td>$\lambda^2$</td>
<td>$\lambda^3$</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$\phi$ (bar)</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>$\lambda^2$</td>
<td>0</td>
<td>0</td>
<td>$\lambda^2$</td>
<td>$\lambda^3$</td>
</tr>
</tbody>
</table>

$k(car, cat) = \lambda^4$, $k(car, car) = k(cat, cat) = 2\lambda^4 + \lambda^6$. 
Normalization

The kernel (dot product) between documents $x$ and $y$ is given by

$$k_n(x, y) = \sum_{s \in \Sigma^n} \sum_{s \subset x} \sum_{s \subset y} \lambda^{L(s_x) + L(s_y)},$$

where $\Sigma^n$ is the set of all substrings of size $n$ from a finite alphabet $\Sigma$.

**Example:** Consider the words cat, car, bat and bar with $|s| = 2$:

<table>
<thead>
<tr>
<th></th>
<th>c-a</th>
<th>c-t</th>
<th>a-t</th>
<th>b-a</th>
<th>b-t</th>
<th>c-r</th>
<th>a-r</th>
<th>b-r</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\phi($cat$)$</td>
<td>$\lambda^2$</td>
<td>$\lambda^3$</td>
<td>$\lambda^2$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$\phi($car$)$</td>
<td>$\lambda^2$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>$\lambda^3$</td>
<td>$\lambda^2$</td>
<td>0</td>
</tr>
<tr>
<td>$\phi($bat$)$</td>
<td>0</td>
<td>0</td>
<td>$\lambda^2$</td>
<td>$\lambda^2$</td>
<td>$\lambda^3$</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$\phi($bar$)$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>$\lambda^2$</td>
<td>0</td>
<td>0</td>
<td>$\lambda^2$</td>
<td>$\lambda^3$</td>
</tr>
</tbody>
</table>

$$k(car, cat) = \lambda^4, \quad k(car, car) = k(cat, cat) = 2\lambda^4 + \lambda^6.$$
The kernel (dot product) between documents $x$ and $y$ is given by

$$k_n(x, y) = \sum_{s \in \Sigma^n} \sum_{s \subseteq x} \sum_{s \subseteq y} \lambda^{L(s_x)+L(s_y)},$$

where $\Sigma^n$ is the set of all substrings of size $n$ from a finite alphabet $\Sigma$.

**Example:** Consider the words cat, car, bat and bar con $|s| = 2$:

<table>
<thead>
<tr>
<th></th>
<th>c-a</th>
<th>c-t</th>
<th>a-t</th>
<th>b-a</th>
<th>b-t</th>
<th>c-r</th>
<th>a-r</th>
<th>b-r</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\phi$(cat)</td>
<td>$\lambda^2$</td>
<td>$\lambda^3$</td>
<td>$\lambda^2$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$\phi$(car)</td>
<td>$\lambda^2$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>$\lambda^3$</td>
<td>$\lambda^2$</td>
<td>0</td>
</tr>
<tr>
<td>$\phi$(bat)</td>
<td>0</td>
<td>0</td>
<td>$\lambda^2$</td>
<td>$\lambda^2$</td>
<td>$\lambda^3$</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$\phi$(bar)</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>$\lambda^2$</td>
<td>0</td>
<td>0</td>
<td>$\lambda^2$</td>
<td>$\lambda^3$</td>
</tr>
</tbody>
</table>

$k(car, cat) = \lambda^4$, $k(car, car) = k(cat, cat) = 2\lambda^4 + \lambda^6$. 
Normalization

- There are different types of string kernels (spectrum, constant, sequence, exponential, boundrage), depending on the weight $\lambda$ and the substring size $s$.
- If $\lambda(s) = 0$ for substrings starting and ending with white space, we obtain the “bag of words” kernel.
- It can be computationally expensive for large documents.
Normalization

Substitution of non-standard words

- Kernel PCA based on Aspell suggestions for pseudoestudiantes. We used a sequence string kernel with $s = 3$ and $\lambda = 0.5$. 
Normalization

Substitution of non-standard words

- The projection of **pseudoestudiantes** in the first and second Principal Components is

![Diagram showing the projection of pseudoestudiantes in the first and second Principal Components]

- By using the minimum distance criterion with 3 PC, the most similar word is pseudo-estudiantes, which is correct.
Simulation test: randomly change 1 letter in a sample of 200 words from spanish dictionary
Simulation test: randomly change 2 letters in a sample of 200 words from spanish dictionary
Some results:

Original

Ese cafe del oxxo si que levanta!!!
Puuues a chambearts!!!

No se cual cafe sea mas malo
si el del @7ElevenMexico o el
de @Tiendas_OXXO pero ambos
son malisiiiiimoooooosss!!!
Pesima mezcla

@paolastonexxx eso es
lo bonito... Puedo pagar
en OXXO. :)

Normalized

ese cafe si levanta pues
chamba

no se cual cafe
sea mas malo si
pero ambos son
musimos pesima
mezcla

eso es bonito puedo
pagar
emoticon-positivo
Classification
We implement a classification algorithm similar to that used by Melville et al. (2013).

- We used **bag of words** in the preprocessed and normalized tweets (positive, negative and neutral), to obtain **relevant words** for each category by using a mutual information measure (Yang and Pedersen, 1997).

- We add a **topic-related** list of words for each category as an apriori information.
We implement a multinomial naive Bayesian classifier, where the class $c$ of a tweet is given by

$$c^* = \arg \max_c p(c|d),$$

where

$$p(c|d) = \alpha_1 p_1(c|d) + \alpha_2 p_2(c|d),$$

$p_1(c|d), \alpha_1$: class probabilities and weight using bag of words

$p_2(c|d), \alpha_2$: class probabilities and weight using the topic-related words. And

$$p(c|d) = \frac{p(c) \sum p(w|c)^{n_i(d)}}{p(d)}.$$
We use 800 tweets previously classified.

- We use 80% for training and 20% for testing by using a Cross Validation criteria.
- The Mean Average Error (MAE) for the training data was $0.192 \pm 0.015$. The MAE for test data was $0.23 \pm 0.021$.
- But...
We use 800 tweets previously classified.
We use 80% for training and 20% for testing by using a Cross Validation criteria
The Mean Average Error (MAE) for the training data was $0.192 \pm 0.015$. The MAE for test data was $0.23 \pm 0.021$
But...
We use 800 tweets previously classified.

We use 80% for training and 20% for testing by using a Cross Validation criteria.

The Mean Average Error (MAE) for the training data was $0.192 \pm .015$. The MAE for test data was $0.23 \pm .021$.

But...
We use 800 tweets previously classified.

We use 80% for training and 20% for testing by using a Cross Validation criteria

The Mean Average Error (MAE) for the training data was 0.192 ± .015. The MAE for test data was 0.23 ± .021

But...
<table>
<thead>
<tr>
<th></th>
<th>N</th>
<th>O</th>
<th>P</th>
</tr>
</thead>
<tbody>
<tr>
<td>N</td>
<td>14</td>
<td>28</td>
<td>6</td>
</tr>
<tr>
<td>O</td>
<td>9</td>
<td>479</td>
<td>63</td>
</tr>
<tr>
<td>P</td>
<td>0</td>
<td>15</td>
<td>5</td>
</tr>
</tbody>
</table>

Error: 0.195

clase real

clase estimada
It is necessary to use classification algorithms sensitive to unbalanced categories
It is necessary to use classification algorithms sensitive to unbalanced categories.
Work in progress
We are improving tweets normalization:
- Implementing the methaphone algorithm
- Testing different types of string kernels
- Improving the apriori information (word list for normalization and classification)

- Hashtags information
- Classification methods (SVM, Boosting)
- Cost sensitive classification
- Spatial and temporal analysis of tweets
Thank you for your attention!