**REPORT OF SENTIMENT ANALYSIS**

Students:

FERREIRA, Jesse.

TALBI, Samir.

More details on github: <https://github.com/jessefilho/sentimentanalysis>

**INTRODUTION**

This report has the purpose of put in practice the knowledge achieved during the discipline of Natural Processing Languages, diving in the yield of sentimental analysis through the text classification.

Sentimental Analysis is important to detect customers feedback, political orientation, epidemic flow and a lot of other subjects related and express via pure text. Text is the data format most available on internet and mining it give a bunch of options, however, mining data with this kind of format raise several challenges such as languages, lexical morphology, symbols and characters, etc. To make possible the sentimental analysis it necessary apply several techniques such as Tokenization, Morphological analysis, Morphological disambiguation, and so on.

For sake of simplicity this report will present 3 techniques: Tokenization by simply trimming words of a given corpus, tokenization and concatenation of Part of speech form the Morphological disambiguation and ngram. Using as weighting factor the frequency of terms to the analysis of positive or negative sentences.

**TASK 1**

**Term frequencies:**

The computation of frequency of terms is an important approach that can increase the weight over a sentence, such frequency can indicate that a sentence is important or not. This study focusing on the simple computation of terms to the first part of Task 1.1, where we study the tokenization by simply trimming words. Thus, In the Task 1.2, we did terms weighting based on its POS (Part-Of-Speech), with the purpose of highlight the adjectives (JJ) and verbs (VB). Such kind of words can be helpful to distinguish whether a sentence is positive or negatives.

**Code:**

**A screenshot of a cell phone

Description automatically generated**

* 1. **- Simply Trim (without Part-Of-Speech):**

We used the basic trim offered as support script.

* 1. **- Classifiers used and results with term trimmed:**

|  |  |  |
| --- | --- | --- |
| **Name** | **Train Precision (%)** | **Test Precision (%)** |
| LibLinear | 86.375 | 87.5 |
| SVM SPegasos | 88.375 | 88 |
| RandomFlorest | 81.6875 | 70.5 |

**2.1 – Part-Of-Speech Trimmed:**

We used the basic trim offered as support script.

**2.2 - Classifiers used and results with term Part-Of-Speech Trimmed:**

|  |  |  |
| --- | --- | --- |
| **Name** | **Train Precision (%)** | **Test Precision (%)** |
| LibLinear | 80.0625 | 76.5 |
| SVM SPegasos | 85.125 | 75 |
| RandomFlorest | 86.5625 | 83.5 |

**TASK 2**

**NGRAM:**

N-grams are contiguous sequences of n-items in a sentence. N can be 1, 2 or any other positive integers, although usually we do not consider very large N because those n-grams rarely appears in many different places.

At this stage of the process, we had to apply the n-gram function to do so, we modified the trim-tagged.py file as the following code presents:

**CODE**

**A screenshot of a cell phone

Description automatically generated**

As we generated numerous files containing our words groups, we will gather them and create a new dictionary. When it comes to our arff file, we will apply our previous script applying our newly-created dictionary. Here are our results:

* 1. – **2-grams:**
  2. **Classifiers used and results with term Part-Of-Speech Trimmed:**

|  |  |  |
| --- | --- | --- |
| **Name** | **Train Precision (%)** | **Test Precision (%)** |
| LibLinear | 79.375 | 71 |
| SVM SPegasos | 98.4375 | 79.5 |
| RandomFlorest | 81.625 | 75 |

* 1. **- 3-grams:**
  2. **Classifiers used and results with term Part-Of-Speech Trimmed:**

|  |  |  |
| --- | --- | --- |
| **Name** | **Train Precision (%)** | **Test Precision (%)** |
| LibLinear | 79.375 | 71 |
| SVM SPegasos | 58.75 | 61 |
| RandomFlorest | 64.6875 | 61.5 |

At 3-grams our patterns do not appear enough to get some correct results as you can see above. It considerably decreases between 2 and 3-grams so we had to change some settings in 2-grams to get better accuracy.

**How to reduce the number of terms (n-grams) while building document vectors without loosing the performance of classification?**

One strategy is removing the tags and conserve only the tags most expressive such as JJ and VB, or NN. Then, with this strategy we had achieved a precision of **85 %** at the model withSVM SPegasos, where before from 98.44% with 14 907 terms to 85 % with 2 274. In other words, we had reduced 75% of terms with only 13% of loss on the precision.

Please see the github for more details such, time of execution, confusion matrix, etc.  
<https://github.com/jessefilho/sentimentanalysis>