**Semantic Analysis**

*Assignment 3*

*Computational Linguistics*

Cristina Luengo Agulló

Javier Navarro González

École Polytechnique de Louvain

Université Catholique de Louvain

December 09, 2014

**Introduction**

To solve the problems asked in this assignment, we have made a python script which forms the ***lexicon*** of the 5-gram corpus and removes the 20 most frequent words, makes the ***bag of words*** of each word of the lexicon, computes the **tf-idf representation**, calculates the **similar words** of a given word and finds the **most similar** pair of word types.

This is done over the same script using the input arguments. These are the possible tasks that the script can do and the way to write the command:

* **Bag of words:** ./assignment3.py w5\_.txt target\_word **bow**
* **Representation of tf*-*idf**: ./assignment3.py w5\_.txt target\_word **tfidf**
* **Similarity:** ./assignment3.py w5\_.txt target\_word **similarity**
* **Max. similarity:** ./assignment3.py w5\_.txt **maxSimilarity**

The script takes seconds to execute for all tasks except for the computation of the most similar pair of word types.

**Lexicon**

After removing the 250 most frequent words from the 5-gram corpus, we obtain the **lexicon** that we are going to use during the assignment. These are the top 20 most frequent words:

|  |  |
| --- | --- |
| Words | Frequency |
| Home | 33.482 |
| Reason | 33.138 |
| News | 32.772 |
| Second | 32.645 |
| Difficult | 32.014 |
| Job | 31.782 |
| Until | 31.475 |
| Own | 30.931 |
| Pay | 30.586 |
| War | 30.554 |
| Big | 30.459 |
| Looking | 30.190 |
| White | 30.155 |
| Hands | 29.849 |
| Show | 29.745 |
| Women | 29.717 |
| Order | 29.627 |
| Beginning | 29.578 |
| Different | 29.180 |
| Ago | 29.165 |

Another method we could have used to discard the 250 most frequent words is by using the *idf* computation of each word. The *‘inverse document frequency’* is a term that measures how common a word is among a collection of documents, so the more frequent a word is, the lower is its *idf* value. Therefore, one way to determine which the 250 most frequent words are would be to calculate the idf value for each word of the lexicon and then sorting the words by their value in increasing order. The first 250 words will be the most frequent ones, so deleting them will give the requested lexicon. A script for doing this computation is provided (idf.py).

**Bag of words**

The bag of words of a given word is calculated by taking all the words that appear in the same 5-grams as the target word, and storing their frequency of appearance in the 5-grams. Those words will form the context of the given word.

As an example, the bag of words of ***Fireworks*** is:

|  |  |
| --- | --- |
| Words | Frequency |
| July | 10 |
| Fourth | 21 |

These are the words that appear on the same 5-grams as fireworks. This happens in these 5-grams:

* **11**  fireworks on the fourth of
* **10** the fourth of july fireworks

The words in red are the ones that **don’t belong**to the lexicon because they have been removed (they were on the top 250 most frequent words), and the words in green are the ones that form the ***bag of words***.

The word ***july*** appears with ***fireworks*** in one 5-gram with frequency ***10***, while the word ***fourth*** appears in two 5-grams, once with frequency ***10*** and in the other ***11***, which sums ***21***, the value that appears in the bag of words.

And this is the bag of words of ***Furnace***:

|  |  |
| --- | --- |
| Words | Frequency |
| Water | 6 |
| Heater | 6 |

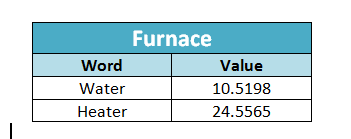
These words appear in the only 5-gram where furnace appears:

* **6** the furnace and water heater

**TF-IDF Representation**

Using the results obtained from the bag of words in the previous point, we can compute the TF-IDF representation of words . Therefore, the values of the words in the context of a certain word will be calculated considering two weights: TF and IDF.

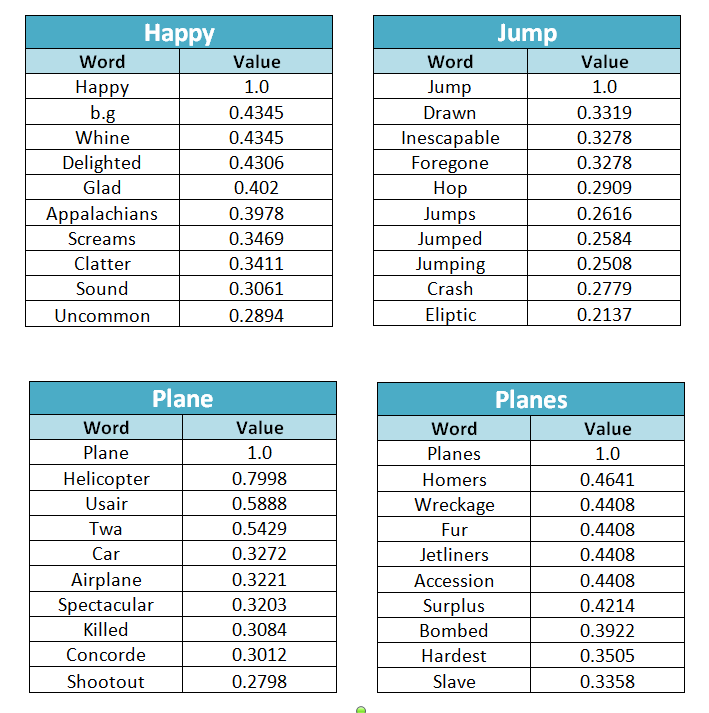
* **TF:** The number of times a word appears in the context (we can reuse the values from the bag of words for this computation).
* **IDF:** It is calculated based on the formula , where *nContextsPresent* is the number of times the word appears in other contexts (in other bags of words) divided by the number of contexts (which is equal to the number of words in the lexicon). According to our understanding of the handout, we consider all the contexts of all the words (included or not in the lexicon) to compute the number of contexts in which a word appears (we associate the concept of context with a document, so it would represent the number of different documents that contain the word), and the total number of contexts.

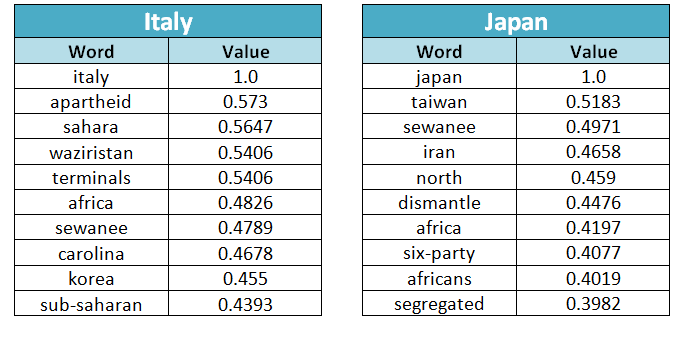
The TF-IDF representation for the words Fireworks and Furnace is the following one:

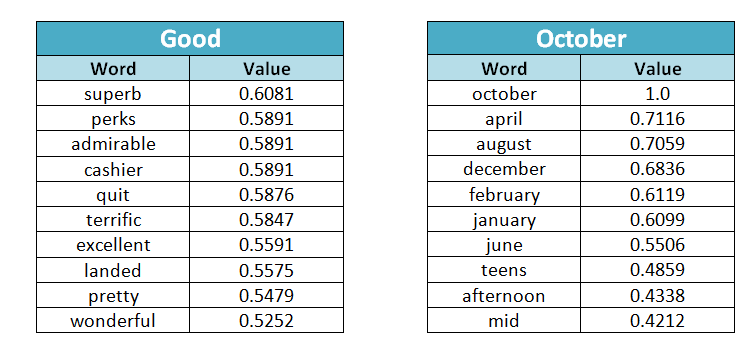
|  |  |
| --- | --- |
| Fireworks | |
| Word | **Value** |
| July | 26.5342 |
| Fourth | 46.9439 |

Using the values of the TF-IDF representation as vectors, we can compute the similarity between two words. Therefore, each vector will have all the TF-IDF values of the words contained in the context of a given word, and also, zeros on the positions of the words that are in the other context vector and not in the current one.

For example, given the two contexts c1 = [‘hi’,’Karl’] and c2 = [‘hi’,’Charles’], with TF-IDF values of tfidf1 = [0.5,0.4] and tfidf = [0.8,0.3], we would build the vectors v1 = [0.5,0.4,0] and v2 = [0.5,0,0.3].

Once we have the vector representations of the contexts of two words, we can compute the similarity between them. As an example, the top ten similar words for the requested words in the handout are shown below.





Word ***Good*** isn’t part of the lexicon after the pre-processing task (deleting the 250 most frequent words), but its context is computed anyway by taking all the words that appear in the same 5-grams as it and belong to the lexicon.

Most of the similar words have sense for us because are **synonyms**, **verbal tenses**, **derivate words** or share a relationship with their similar word (feelings, same characteristics, etc.).

This is what we have found over the similar words of the requested ones in the assignment:

* **Happy:** Words that also express feelings (glad, delighted).
* **Jump:** Synonyms (hop) and different verbal tenses (jumped, jumping).
* **Plane:** Different aircrafts with common features (helicopter, airplane, concorde).
* **Planes:** Words related with the army (bombed, wreckage).
* **Italy:** Countries, continents and regions (korea, africa, carolina).
* **Japan:** Countries and continents (taiwan, iran, africa).
* **Good:** Other similar adjectives (excellent, terrific, wonderful).
* **October:** Mostly other months (april, august, december, etc.).

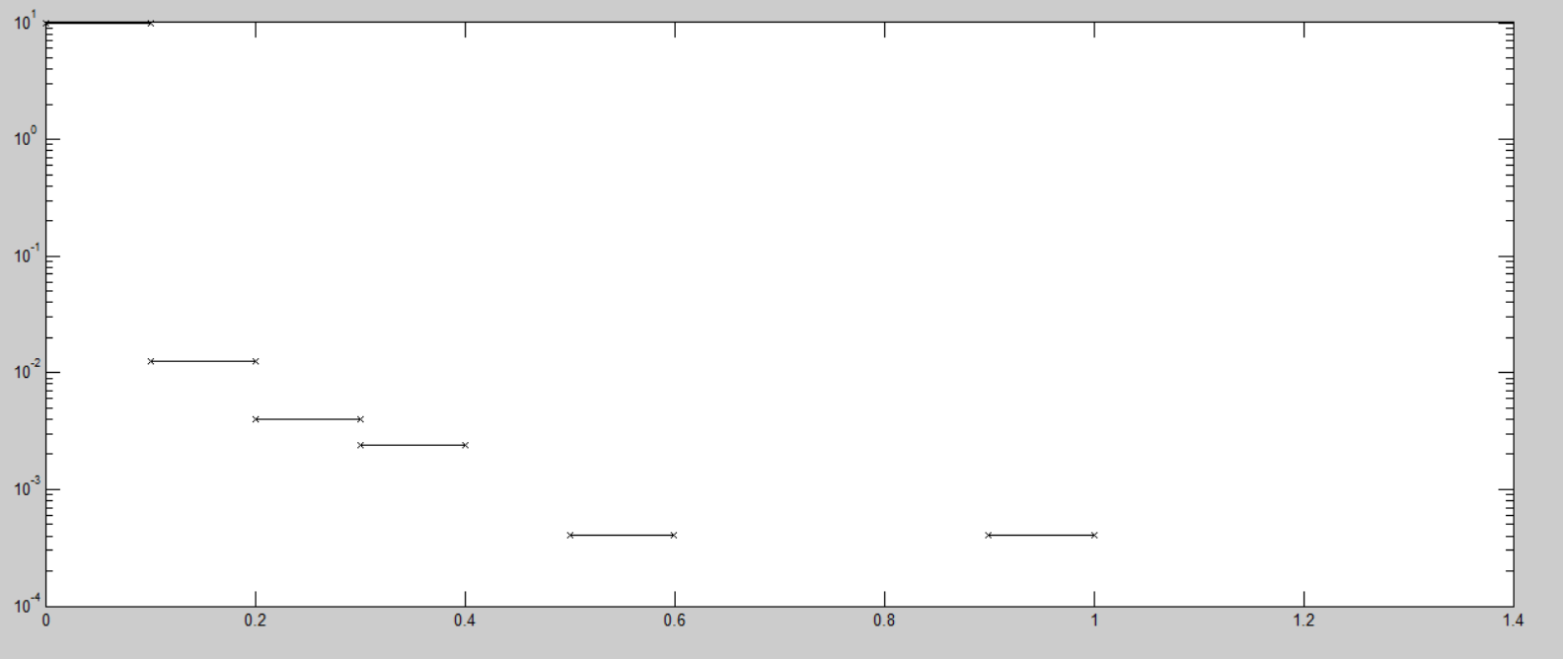
**Bonus task**

Due to time constraints when computing the two most similar words, we weren’t able to obtain the final value in time (the script didn’t finish executing in time), but the code to compute it is in *assignment.py*.

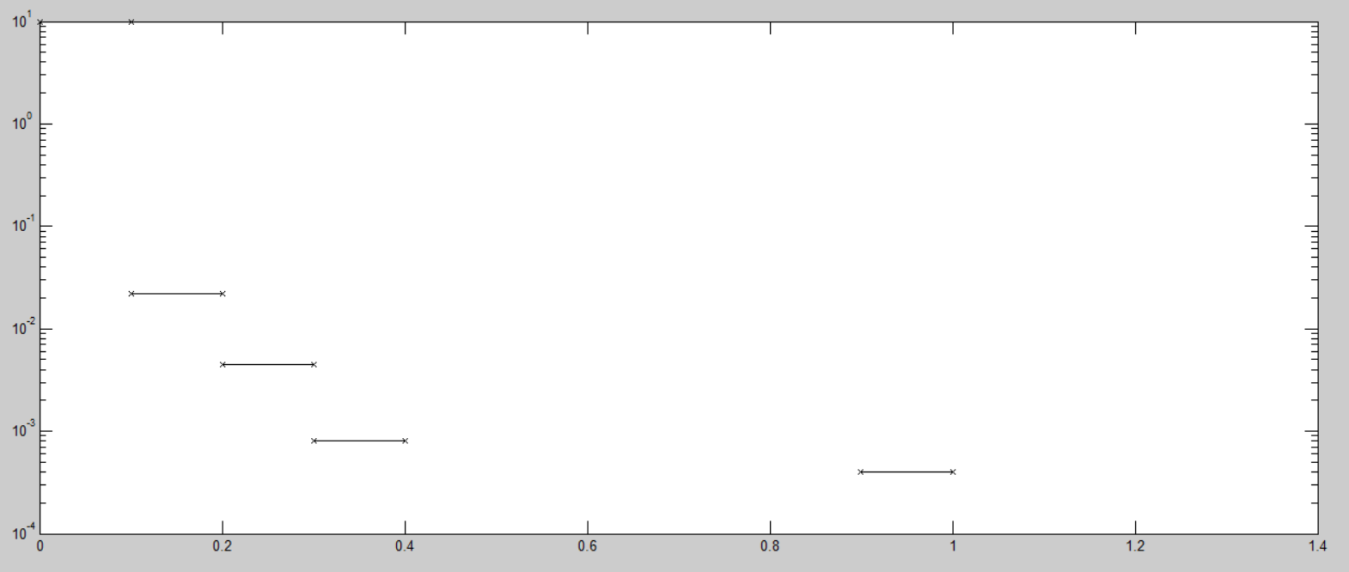
**Distribution of similarity scores**

The empirical distribution of similarity scores between the whole lexicon and the terms *Christmas* and *Gift* are shown below.

For ***Christmas***:



For ***Gift***:



The Y axis represents the cumulative probability of the values in the X axis.

As can be seen, the probability distribution in both graphs shows that the most common case is that of words that are not similar to the given words. Therefore, only a few words from the whole lexicon will be more similar to the requested ones (being the same word the one that has the lowest value in the Y axis and the highest one in the X axis).