Report

1.Thresholds:

The first threshold was filtering the lemmas, which doesn’t reach the 100 occurrences we also clean up the entire sentence that means, we took context, which appeared at least 100 times.

While we count context for each word no matter which variation we execute, we took only 100 most common contexts for each word, which gave us better performance.

Examination:

2. After examination of the results, we can see that in sentence similarity the words at top looks very closer but as we go, down we see that the words with context feature is closer than other. We also understand that window feature parsing is giving words with closer definition then sentence parsing. For example if we take hotel the third row shows the sentence parsing: room, window parsing: plaza and context parsing: inn

3. While we investigate the 1-st and 2-nd order we find that in context, we can probably switch the words for example instead of bus and context stop we can replace it with train and stop. We can just recreate new phrase by combining context and similar word but in some cases, it is harder to do it because in sentence co-occurrence, there is another word, which correlated, with target word but actually they aren’t similar.

4. To judge the similarity, we need to do it semanticaly and topically for the three different types.

For the word ‘**car**’:

Semantics : MAP\_car = 0.564

Topically : MAP\_car= 0.839

For the word ‘**piano**’:

Semantics : MAP\_piano = 0.813

Topically : MAP\_piano = 0.542

We can notice, that we get ‘opposite’ results. High result for topically for car and high result for semantics for piano. Indeed, for piano we get a lot of instruments’ words ‘violin’, ‘bass’… On the contrary, for car, we get more of different words into the topic : ‘driver’.

5. Implementation:

* Run over each parser(Sentence, Window, Context)
* Read the file, filter it by functions words, and add only words from contents classes.
* Create all context for each word and store it store future using
* Count each context per word
* Calculate norm for each word
* Calculate PMI sparse matrix by using only same context multiplication
* Find for each target word the closest words and print them

**Word2Vec**

6-2. We received lists of similar words for our target words according to the dependencies (Type 1) and according to word to bag k=5 (type 2).

We can notice that we receive very good results.

Through a subjective judgement, we can really see the similarity between the words. For example, the word ‘car’ we receive different type of cars (limosine), different type of means of transport (motorcycle), or what’s within the car (rear-engined). There is not a big difference between BOW and Deps. Indeed, we receive almost the same words but in a different order and some different words.

6-3. For the bag to word k=5 similarity, we get a list of similar contexts. We can notice that we get words that we can replace the target-word by them. For example, for ‘guitar’, we get ‘bass’ or ‘drums’.

For the dependency contexts, we get a list of words with their tags. We can notice, that here, we receive words that are more characteristics of the target-words. For example, for ‘piano’, we get words like ‘virtuoso’ or ‘harris’.

6-4. To judge the similarity, we need to do it semanticaly and topically for type 1 and for type 2.

Let’s do it for **car** :

Semantics : MAP\_car = 0.9615

Topically : MAP\_CAR = 0.924

We get good results for both because we get, in particular types of cars.

We do the same for **piano** :

Semantics : MAP\_PIANO = 0.975

Topically : MAP\_PIANO = 0.41

Semantics, the results are good because, we get a lot of different types of instruments (violin, cello…). But topically, we don’t get a lot of words in the piano type (pianoforte, vibraphone…)

There is, thus, a different between what we get according to the words. Topically, the results are very different for car and for piano.