Natural Language Processing
Assignment 3

Arie Cattan 336319314
Moche Uzan F957022

Abstract:

In this assignment, we represented words in the corpus data as distributional vectors based to three types of co-occurrence between the target word and its features:
-co-occurring word in a sentence
-co-occurring word in a widow of two content words on each side
-dependency edge between the target word and the feature

The corpus contains 774858 sentences.

Using these vectors, we then computed word similarities and got the 20 most similar words of several words for each type of co-occurrence.
Since these vectors are very sparse, we represented them as dense vectors and implemented efficient algorithm to compute simultaneously similarity between a target word and all other words.

At the end, we compared our results with the similar words based on the word2vec.

1. Threshold for target word: 100
   Threshold for context word: 20
   Threshold for word feature pair : 10

   Number of words considered for similarity: 8426

   Sentence co-occurrence: min_features = 4, max_features = 23302
   Window co-occurrence: min_features = 1, max_features = 12571
   Dependency: min_features = 1, max_features = 11198

2. The list of the 20 most similar words appears in the file similarities.txt.

   We can observe that the sentence and the window approach are so similar, they find words in the same domain than the target word, whereas the dependency approach tends to find words behave similar to the target word. In *car*, the top similar words regarding to sentence and window co-occurrence types are motor, drive, driver, race… related to the car domain The dependency approach find other means of transport like automobile, truck, bus, motorcycle, boat …
   In one word, sentence and window co-occurrence type find **topic** relation, while dependency co-occurrence type find **semantic** behavior.

3. The list of the 20 top context attributes appears in the file contexts.txt.
   The above list contains topically or semantic **similar words** and this list contains content words **appear usually near** the target word.

4. Manual judgments appear in the files car_dist_judgment.txt and piano_dist_judgment.txt.
   The first column corresponds to the topic relation and the second to the semantic relation.
   As I write in the part 2, I think the two first co-occurrence types find topic relation and dependency co-occurrence type find semantic relation, this the reason why I've computed the MAP values regarding to topical relation for the two first, and regarding to semantic relation for the last.

   Car: sentence co-occurrence : MAP = 0.41857102272727287
   window co-occurrence : MAP = 0.3996351674641148
   dependency relation : MAP = 0.40341270874378454

Piano: sentence co-occurrence : MAP = 0.26531443250377074
window co-occurrence : MAP = 0.2301006394920869
dependency relation : MAP =0.4107304778554777

5. The first step consists to extract all the word context pairs from the corpus based on the different types of co-occurrence.

   python extract_features.py data/wikipedia.sample.trees.lemmatized -option
   (option can be -s for sentence co-occurring, -w for window co-occurring, -d for dependency or -all for all of them).

   Then, we loaded only the pairs occur more the word context threshold, and computed pmi values.
   We created a dict of dict to store the pmi values : word_context[word][context] = pmi

   We chose to use this pmi formula because we can easily compute p(y|x) and p(y)
   In our case, x is the target word and y is the context.
   p(y) — probability of the context
   p(y|x) — probability of the the context given the target word

   $$\log \frac{P(y|x)}{P(y)}$$

   In order to implement the efficient algorithm for computing all similarities for a target word, we first built a list of target words for each possible attribute, and stored the norm of all word vectors.
   Then, we implemented this algorithm with the pmi values, as follow:

```
for att, pmi1 in word_context[word].items():
    for word2 in context_word[att]:
        words[w2i[word2]] += pmi1 * word_context[word2][att]
```

6. The lists of most similarities and contexts are respectively in the files word2vec_similarities.txt and word2vec_contexts.txt.

   In addition to what we learned in class, the reasons why we don't receive the same results in the both approach are the word2vec vectors were computed using the word itself and not the lemma and they were trained with much more data.

   As our distributional similarities in the part 2, 4, the word2vec vectors based on the window co-occurrence and often find **topically** similar words, whereas these based on the dependency relation find **semantic** similar words.
   But this hypothesis is not always true; for the words car and piano, the both co-occurrence types find semantic similar words, thus we computed the MAP values regarding to semantic relation.

   Manual judgments appear in the files word2vec_car_dist_judgment.txt and word2vec_piano_dist_judgment.txt.

   Car: bow5 : MAP = 0.40999409965
   deps : MAP = 0.525

   Piano: bow5 : MAP = 0.473194325895
   deps : MAP = 0.437377420084