MASSACHUSETTS INSTITUTE OF TECHNOLOGY
Department of Electrical Engineering and Computer Science
6.806/6.864 Advanced Natural Language Processing
Fall 2015

Assignment 4, Due: 9 AM on Friday, November 13.

Mapping distributional to model-theoretic semantic spaces

Objectives

Word embeddings have been shown to be useful across state-of-the-art systems in many NLP tasks, ranging from speech recognition to dependency parsing. In this homework, we will explore word embeddings and their utility for modeling language semantics. In particular, we will analyze whether word embeddings may help predict features of a given concept.

We will investigate the method used in the paper Building a shared world: Mapping distributional to model-theoretic semantic spaces published at EMNLP this September by Aurelie Herbelot and Eva Maria Vecchi[1]. It received an honorable mention for best paper. We will go a little further than what a typical reviewer could do and actually test the veracity of some of the claims.

Example

Here is an example of a concept along with some of its features:

```
yam a_vegetable all all all
yam eaten_by_cooking all most most
yam grows_in_the_ground all all all
yam is_edible all most all
yam is_orange some most most
yam like_a_potato all all all
```

The concept *yam* is annotated with six features (*a_vegetable*, *eaten_by_cooking*, *grows_in_the_ground*, *is_edible*, *is_orange*, and *like_a_potato*). Each feature is annotated by three different humans. The annotation is a quantifier that reflects how frequently the concept has a feature. Five quantifiers are used: *no*, *few*, *some*, *most*, and *all*. In this example, the concept *yam* has been annotated as *some*, *most* and *most* for the feature *is_orange*.

Following the paper, each 5 quantifier is converted into a numerical format with the

[1]For the purpose of this homework, you can ignore sections 6, 7 and 8 of the paper.

following (a bit arbitrary) mapping: no $\rightarrow$ 0; few $\rightarrow$ 0.05; some $\rightarrow$ 0.35; most $\rightarrow$ 0.95; all $\rightarrow$ 1. The value is averaged over the three annotators. Using this mapping, we can map a concept word into a "model-theoretic vector"[2] (which we will sometimes call feature vector). If a feature has not been annotated for a concept, then the element in the model-theoretic vector corresponding to the feature will have value 0. As a result, any element of a model-theoretic vector that has value 0 may correspond to a feature that has either been annotated as *no* by the three annotators, or not been annotated (presumed *no*). Given that there are many features and only some of them are annotated for each concept word, the "model-theoretic" vector is likely quite sparse.

In the yam example, if we only included features annotated with yam, the model-theoretic vector would be as follows:

$$\begin{bmatrix} \frac{\text{all+all+all}}{3} \\ \frac{\text{all+most+most}}{3} \\ \frac{\text{all+all+all}}{3} \\ \frac{\text{all+most+all}}{3} \\ \frac{\text{some+most+most}}{3} \\ \frac{1+1+1}{3} \end{bmatrix} = \begin{bmatrix} \frac{1+1+1}{3} \\ \frac{1+0.95+0.95}{3} \\ \frac{1+1+1}{3} \\ \frac{1+0.95+1}{3} \\ \frac{0.35+0.95+0.95}{3} \\ \frac{1+1+1}{3} \end{bmatrix} \approx \begin{bmatrix} 1 \\ 0.967 \\ 1 \\ 0.983 \\ 0.75 \\ 1 \end{bmatrix}$$

The additional coordinates corresponding to all the remaining features would be zero. Each concept word will have a sparse vector of the same dimension (number of unique features) in the same dataset. The coordinates mean the same from one concept to another. For example, feature `is_vegetable`, appears in the same coordinate position in all the vectors.

Model

In the previous example, we have seen how to convert a concept into a model-theoretic vector. The goal of the study is to analyze whether there exists a transformation from the word embedding of a concept to its model-theoretic vector. The word embeddings are taken from the word2vec pre-trained GoogleNews-vectors-negative300 word embeddings[3] (300 dimensions), which were trained on part of the Google News data set (about 100 billion words).

The transformation used in the paper is based on Partial Least Squares Regression

[2] For more information about model theory if you are curious, see: http://plato.stanford.edu/entries/model-theory/

[3] https://code.google.com/p/word2vec/

(PLSR). This is linear regression but with multiple outputs. The PLSR is fitted on the training set: the inputs are the word embeddings for each concept, and the outputs are the model-theoretic vectors for each concept. You may be a bit surprised that the assumed relationship is linear. Indeed, but we will see how well (or not) it works.

To assess the quality of the predictions, we compute the Spearman rank-order correlation coefficient between the predictions and the gold model-theoretic vectors, ignoring all features for which a concept has not been annotated. The idea is that some of the features might be present but not given as options during annotation. We shouldn't penalize the method for suggesting them. Figure 1 illustrates the model.

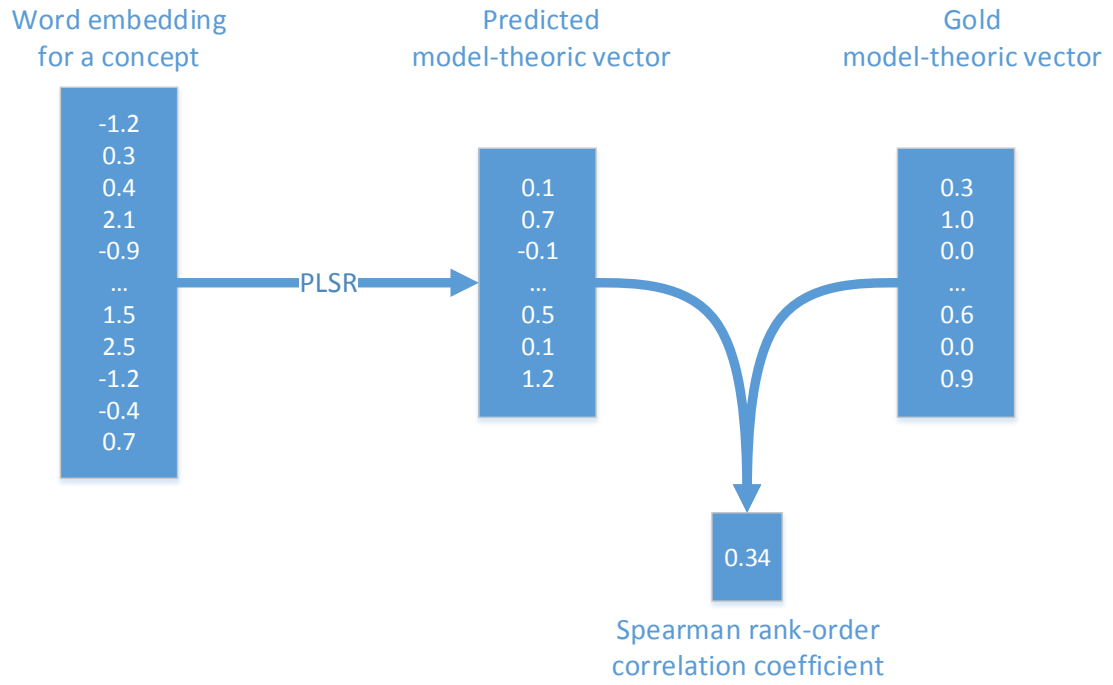

Figure 1: Overview of the system. The word embedding of a concept is transformed to a model-theoric vector via a PLSR. The quality of the predicted model-theoric vector is assessed with the Spearman rank-order correlation coefficient between the predictions and the gold model-theoretic vectors. Note that some of the elements that equal 0 in the gold model-theoretic vector may correspond to features that are not annotated for the concept. Such features are omitted when evaluating the Spearman rank-order correlation coefficient. Also, the dimension of the model-theoretic vectors could be larger or smaller than the dimension of the word embedding. They will be smaller in the AD data set, and larger in the QMR data set.

Data sets

Two data sets are used, AD and QMR:

- AD contains 73 concepts, 54 features. All concepts are animals, and for each concept all features are annotated. There are 3942 annotated pairs of concept-feature ($73 * 54 = 3942$). The dimension of the model-theoretic vectors will therefore be 54.

- QMR contains 541 concepts covering living and non-living entities (e.g. alligator, chair, accordion), as well as 2201 features. One concept are annotated with 11.4 features on average. There are 6187 annotated pairs of concept-feature ($541 * 11.4 \approx 6187$). The dimension of the model-theoretic vectors will therefore be 2201 (and each model-theoretic vector will have on average $2201 - 11.4 = 2189.6$ elements set to 0 due to unannotated features).

Code

The provided code is in the file code/hw4xp.py. It takes care of loading the data, fitting the PLSR, and assessing the predictions using the Spearman rank-order correlation coefficient. For a given experiment, it performs 10 runs, where each run are identical except for the train/test split, which is random. The main parameters of the experiment (data set, classifier, type of word vector, etc.) can be changed in the first lines of the main function of code/hw4xp.py. To run the code, use *python hw4xp.py*.

The code requires Python 2.7, SciPy 0.16.0 or higher, scikit-learn 0.16.1 or higher, and NumPy 1.9.2 or higher. The AD, QMR, and GoogleNews-vectors-negative300 data sets are provided in the /data folder.

Questions

(Question 1.1) [10 points] Run the code and compare the results you obtain with the results presented in the article in Table 3 (numbers 0.346 and 0.634). The results are a bit different: it may be due to the choice of hyper-parameters of your model. What could these hyper-parameters be?

(Question 1.2) [10 points] Without any baseline, it is difficult to assess the quality of the results. Unfortunately, the paper fails to mention any baseline. One way to obtain a baseline is changing the input: the paper uses word embeddings as input to the PLSR, we could instead simply use random vectors of same dimension (300). Implement this baseline and report the results for both the AD and QMR data sets. Use a continuous uniform distribution between 0 and 1.

(Question 1.3) [10 points] Another way to build a baseline is using a dummy or obvious predictor. Implement the following two predictors:

1. A predictor that outputs, for each feature, the most common feature value (a.k.a. the mode) on the training set. E.g. if a feature is annotated as *no* for most concepts, then the predictor will always output *no* for this feature. Note that in finding the most common value we ignore all the concepts for which the feature is not annotated. The resulting predictor has nothing to do with the concepts. Indeed, the predicted values are always the same. But if most concepts have the same value for the feature, the predictor may not be bad.

2. A predictor that outputs for any concept X (new or in the training set) the model-theoretic vector from the training set corresponding to the most similar concept in the training set. Similarity is based on the cosine similarity of the word vectors. This is a simple nearest neighbor predictor.

Implement these two baselines and report the results for both the AD and QMR data sets.

(Question 1.4) [5 points] In the light of the baseline results you have obtained, describe in a few sentences how good you think Table 3's results (numbers 0.346 and 0.634) are.

(Question 1.5) [5 points] In order to improve the approach, we can go in two different directions. We could formulate a better model for this problem (plenty of room for improvement), or we could keep the model the same and try to improve the word vectors used as input. We will only try the latter.

We can apply retrofitting to word embeddings in order to leverage relational information from semantic lexicons by encouraging linked words to have similar vector representations. Using Manaal Faruqui's retrofitting tool[4], try to retrofit the word embeddings we provided (GoogleNews-vectors-negative300-short.txt) on each of the 4 data sets present in the retrofitting tool (framenet.txt, ppdb-xl.txt, wordnet-synonyms+.txt, and wordnet-synonyms.txt[5]), and report your results. You should therefore report 8 different results: 4 data sets to retrofit on, times the 2 semantic spaces AD and QMR, using PLSR. Does retrofitting improve the Spearman rank-order correlation coefficient, compared to word embeddings without retrofitting?

[4] https://github.com/mfaruqui/retrofitting
[5] https://github.com/mfaruqui/retrofitting/tree/master/lexicons

Note: the code you need to write should be relatively short. We have suggested places where to edit the code in code/hw4xp.py by "#TODO". A few sentences are enough to answer to each question.

You should submit your code as well as a PDF containing the answers.