Evaluating Word Representations

1st Practical, Unsupervised Language Learning

5 April 2018

1 Introduction

The goal of this practical is for you to familiarise yourselves with word representation models and different techniques for evaluating them. The word representation model that you will work with is skip-gram, trained using two kinds of context: dependency-based and word window-based. The dependency based model uses dependency annotated context to learn the word representations, as described in the paper by Omer Levy and Yoav Goldberg, *Dependency-based word embeddings*, published at ACL 2014.

https://levyomer.files.wordpress.com/2014/04/dependency-based-word-embeddings-acl-2014.pdf

To complete the practical, follow these steps:

1. Obtain the pre-trained word vectors
2. Evaluate the models on the word similarity task
3. Evaluate the models on the word analogy task
4. Apply a clustering algorithm on the word vectors and analyze the results.
5. Write a 1 page report, describing what you have done and your results and findings.

2 Getting the pre-trained word vectors

Download the pre-trained word embeddings from the dependency based model, and bag of words model with $k = 2, 5$ from here:

- https://levyomer.wordpress.com/2014/04/25/dependency-based-word-embeddings/

Each line is these files is the word, followed by the embedding vector, e.g.:

- 'the -0.0845364250511 0.0323308045594 ... 0.0664685581575\n'

3 Word Similarity Task

Your next step is to evaluate the three kinds of embeddings in the word similarity task. The goal of this task is to compute similarity of two words and evaluate the model-produced similarity against human judgements. Download two commonly-used word similarity datasets:

- SimLex: https://www.cl.cam.ac.uk/~fh295/simlex.html
- MEN: https://staff.fnwi.uva.nl/e.bruni/MEN

Compute cosine similarity between the words using the three models. Evaluate the model-produced similarities against human judgements in terms of Pearson and Spearman correlation coefficients. Compare the performance of the three models on this task. Analyze the data qualitatively and report what are the differences in the kind of similarity captured by the three models. We are interested to see both quantitative results and qualitative analysis in your report.

4 Word Analogy Task

Next, evaluate the three models in the word analogy task. Given an analogy $a : a* :: b :?$, the task is to find $b*$.

Use this link to obtain the Google Word Analogy test set:

- https://aclweb.org/aclwiki/Google_analogy_test_set_(State_of_the_art)

To solve the word analogy tasks with the given word embeddings, you should use the vector offset method based on the cosine distance. This means, to answer the question $a : a* :: b : b*$, where $b*$ is unknown, to find the embedding of $b*$, we simply first compute the offset vector: $v = a * -a$, next compute the vector representation of the word we expect: $b* = b + v$. Most probably, there is no word with the exact same vector we compute this way. Hence, retrieve the word which its vector representation is closer to predicted vector based on cosine distance. Note that, in this task it is important to normalize the word vectors to unit form.

You need to report the accuracy and MRR (Mean Reciprocal Rank) of each of the word embedding models on this task. The accuracy is the percentage of the correctly answered questions from the word analogy question set.

Again, in addition to the quantitative evaluation, conduct a qualitative analysis of the differences between the three models. Include both in your report.

In case you are interested, here you can find more datasets for the word analogy task:

- *https://aclweb.org/aclwiki/Analogy_(State_of_the_art)*

5 Clustering word vectors

You can use $T-SNE$ or PCA to visualize the word vectors in 2 or 3 dimensions, in these plots you can see that there is some kind of relation between the words that are close together.

A couple of different ways to do this:

- http://www.pinchofintelligence.com/simple-introduction-to-tensorboard-embedding-visualisation/
- https://medium.com/@luckylwk/visualising-high-dimensional-datasets-using-pca-and-t-sne-in-python-8ef87e7915b

As the next step, apply a clustering algorithm such as K-Means (or an algorithm of your choice) to cluster nouns, using the three types of word representations as features. Your task is to cluster 2000 frequent nouns (from the list we provided). Apply clustering to each of the three word embedding models, experimenting with different cluster sizes (by decreasing and increasing the number of clusters). Analyze and discuss the qualitative properties of the word clusters for each of the three word embedding models.

You can use the Sklearn implementation of the KMeans algorithm:

- http://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html

```
from sklearn.cluster import KMeans

kmeans_model = KMeans(n_clusters=2, random_state=0)
kmeans_fit = kmeans_model.fit(X)
```

The list of 2000 nouns that you need to cluster is provided here:

- [The link will be added soon!]

6 Write a report

Write a 1 page report, briefly describing

- the methods you have implemented
- the experiments you have conducted
- the results of these experiments
- what you have learned about the properties of the three word representation models

Submit your report in a pdf format to Samira Abnar (s.abnar@uva.nl) by email with the title **ULL-Practical1-FullName1_FullName2**.

Submission deadline for the report is **23:59 on Thursday, 19 April**. Submit only one report per group. Upload your codes on a github repository and put a link to the repository in your report. Also, add the instructions of how to run your codes for each part of the practical to the github ReadMe.

7 Useful resources

- http://colah.github.io/posts/2014-07-NLP-RNNs-Representations/
- https://medium.com/swlh/playing-with-word-vectors-308ab2faa519
- https://www.tensorflow.org/tutorials/word2vec
- https://github.com/donnemartin/data-science-ipython-notebooks/blob/master/README.md
- https://github.com/kudkudak/word-embeddings-benchmarks
- https://github.com/k-kawakami/embedding-evaluation