ADVANCED LEARNING FOR TEXT AND GRAPH DATA

Lab 4: Word and document embeddings

Lecture: Prof. Michalis Vazirgiannis
Lab: Antoine Tixier and Konstantinos Skianis

February 1, 2019

1 Description and requirements

Today, we will play with word and document embeddings and see how they can be applied to (un)supervised document classification. We will make use of the `scikit-learn` and `gensim` Python libraries. `pyemd` must also be installed.

Finally, we will also need the 'GoogleNews-vectors-negative300.bin.gz' file that can be downloaded from https://drive.google.com/file/d/0B7XkCwpI5KDYNlNUTTlSS21pQmM/edit. This binary file contains embeddings for more than 3 million unique words and phrases, learned with word2vec from a corpus of more than 100 billion words (see Mikolov et al. 2013 for more details).

Throughout the entire session, we will be working with the 20 Newsgroup document classification dataset: http://scikit-learn.org/stable/datasets/twenty_newsgroups.html

2 Experimenting with word embeddings

First, we will get familiar with some properties of word embeddings by performing some operations manually. The steps below are implemented either fully or in part in the `word2vec.py` script. As usual, fill the gaps wherever needed.

- Load the data, clean the documents, and construct a list of lists of tokens to be passed to gensim's `build_vocab()` method. Then, load the Google News word vectors corresponding to our vocabulary words. This way, we avoid having to load all the vectors into memory (it would require 5-6 GB of RAM). Compute the cosine similarity in the embedding space between semantically close words (e.g., "man" and "woman") and between unrelated words. What do you observe? Similarly, perform some vector operations and interpret the results.

- Project the word vectors into a lower-dimensional space using PCA/t-SNE[1] and visualize the projections of the most frequent words in a 2D map. You should observe something similar to Figure 1. If you get a nonsensical plot, you may need to update scikit-learn. What can you say about the embedding space? Similarly, observe some regularities in the word embedding space (e.g., gender regularities in Figure 2).

[1] t-SNE: van der Maaten and Hinton 2008 https://lvdmaaten.github.io/tsne/

- Finally, experiment with various approaches for computing sentence similarity: cosine similarity between one-hot vectors (vector space model), between word vector centroids, and the Word Mover's Distance. In the next section, we will use the WMD for unsupervised document classification.

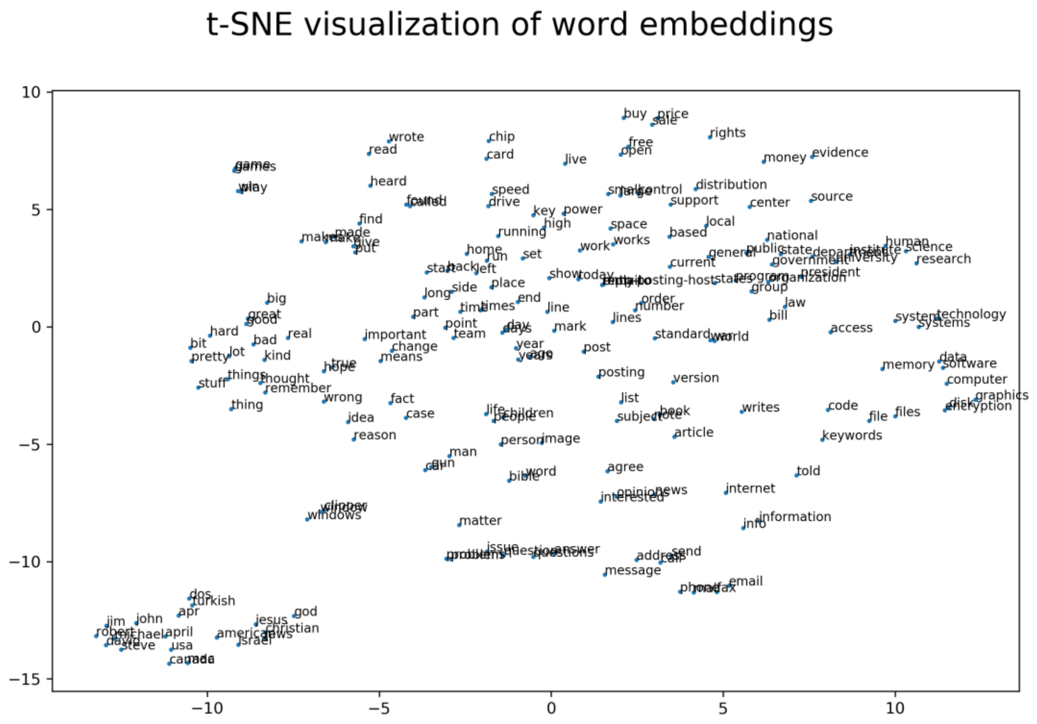

Figure 1

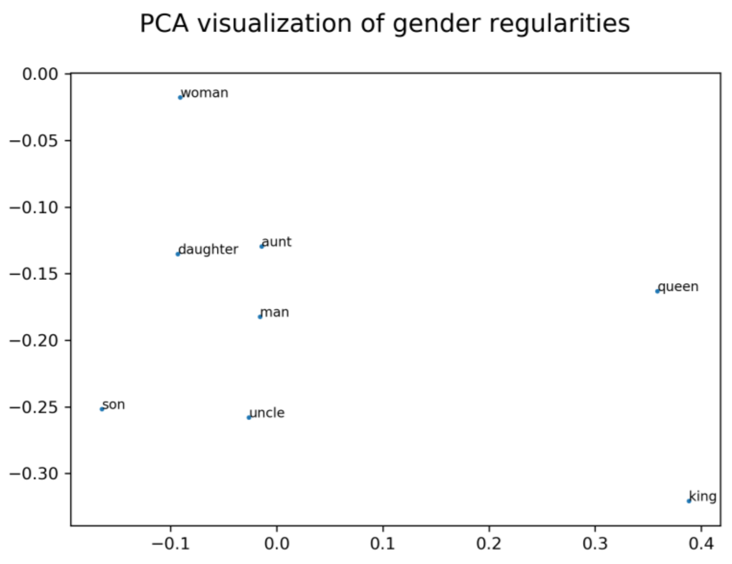

Figure 2

3 Word embeddings for unsupervised document classification

Averaging or summing (element-wise) the vectors of the words in a document may seem like a good way to represent it, but as you will see, this centroid approach does not give good results. A more principled idea was introduced by Kusner et al. (2015) as the Word Mover's Distance (WMD). The WMD is based on the Earth Mover's Distance (Rubner et al. 2000), which extends the notion of distance between single elements to that of distance between sets of elements.

The WMD for two documents d and d' is defined as the minimum cumulative Euclidean distance that needs to be traveled in the embedding space to send all the words of document d to document d'. The documents are represented as normalized bag of words vectors, that is, with entries $c_i / \sum_{k=1}^{n} c_k$, where c_k is the count of word k, and n is the size of the vocabulary (unique words in the collection).

Computing the WMD is equivalent to solving a *transportation problem* that can be formulated as follows:

$$\min_{T \geq 0} \sum_{i,j}^{n} T_{i,j} . \left\| x_i - x_j \right\|_2$$

Where x_i and x_j are the embeddings of words i and j in documents d and d' (resp.), and $T_{i,j}$ is a flow matrix indicating how much of word i in d travels to word j in d'. Solving the problem comes down to computing T with the following constraints:

$\sum_{i=1}^{n} T_{i,j} = c_j / \sum_{k=1}^{n} c_k$: "the incoming flow to word j cannot exceed the weight of word j" (a word cannot receive more than its weight)

$\sum_{j=1}^{n} T_{i,j} = c_i / \sum_{k=1}^{n} c_k$: "the outgoing flow from word i cannot exceed the weight of word i" (a word cannot send more than its weight)

These constraints ensure that d is entirely transformed into d'. All the entries of T are positive as the flow always goes the same direction (no reverse traveling). Specialized solvers for this linear programming optimization task have been developed (Pele and Werman 2009). However, complexity remains high: $O(m^3 \log m)$, where m is the number of words in the two documents. The WMD is a very expensive metric.

As in the original WMD paper, we will use a K nearest-neighbors (Knn) classifier[2] to categorize documents. For each example in the test set, the Knn classifier computes the distances between the example and all the instances in the training set in order to identify the k nearest neighbors of the example. Then, majority voting (the most frequent labels in the set of neighbors) is used to generate the prediction.

We will compare the WMD against cosine similarity between TFIDF vectors and cosine similarity between word vector centroids (note that a distance is the opposite of a similarity measure!). Because the WMD is expensive, we only run our experiments on a small subset of the collection. The

[2] https://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm

code can be found in the `knn_classification.py` file that **should be run from the command line**. Parts of the file have been blanked out and are yours to fill.

4 Document embeddings for supervised text classification

Finally, we will experiment with doc2vec (Le and Mikolov 2014, Dai et al. 2015). You should fill the gaps in the `doc2vec.py` script. Doc2vec is an extension of word2vec that can learn vectors for all the documents in a collection in a fully unsupervised manner. The embeddings can thus be used for unsupervised or supervised classification. In a supervised setting, an inference stage is required to obtain the vectors of the documents in the test set. The model is simply trained on the new documents, with all parameters fixed. Before using doc2vec features with a SVM for supervised classification, we will learn document embeddings for our training set with the PV-DM and PV-DBOW architectures (using `gensim`), and concatenate the resulting vectors as suggested by (Le and Mikolov 2014). Are the similarities between documents meaningful? Do similar documents share the same labels? Visualize 2D maps of the document embedding space (you should obtain something similar to Figures 3 and 4).

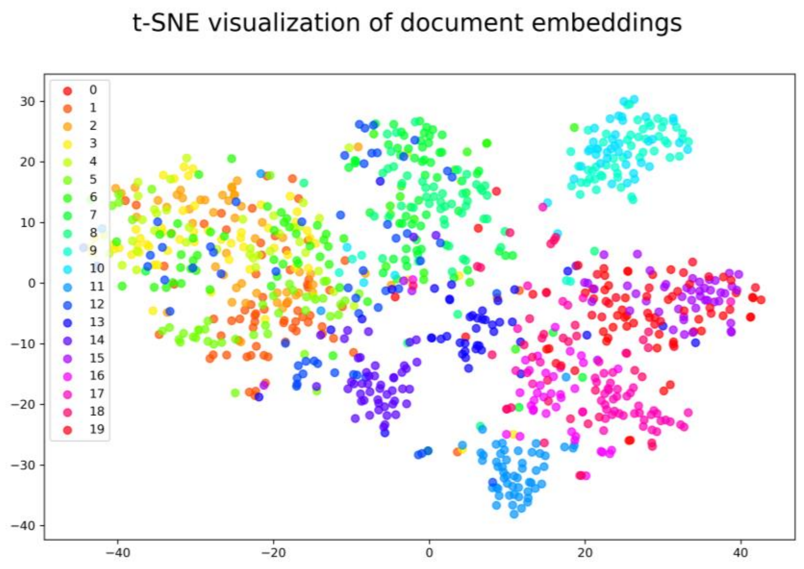

Figure 3

What can you say about the document embedding space? For supervised classification with SVM, how does doc2vec features compare to TF-IDF features? Remember that neural models like doc2vec require very large amounts of training data, and that doc2vec has many tuning parameters (architecture, dimensionality, window size…)

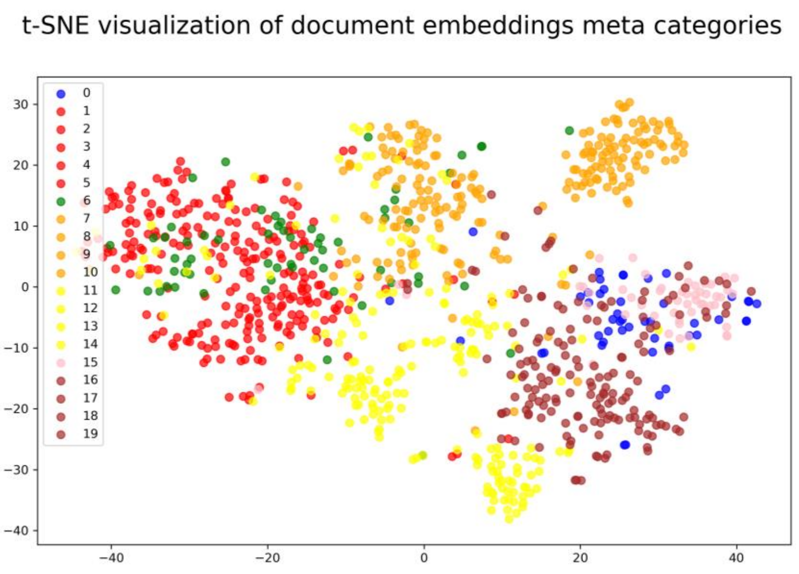

Figure 4

5 References

Dai, Andrew M., Christopher Olah, and Quoc V. Le. "Document embedding with paragraph vectors." *arXiv preprint arXiv:1507.07998* (2015).

Kusner, M., Sun, Y., Kolkin, N., & Weinberger, K. Q. (2015). From Word Embeddings To Document Distances. In Proceedings of the 32nd International Conference on Machine Learning (ICML-15) (pp. 957-966).

Le, Q., & Mikolov, T. (2014). Distributed representations of sentences and documents. In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)* (pp. 1188-1196).

Maaten, Laurens van der, and Geoffrey Hinton. "Visualizing data using t-SNE." *Journal of Machine Learning Research*9.Nov (2008): 2579-2605.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., & Dean, J. (2013). Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems* (pp. 3111-3119).

Pele, O., & Werman, M. (2009, September). Fast and robust earth mover's distances. In *Computer vision, 2009 IEEE 12th international conference on* (pp. 460-467). IEEE.

Rubner, Y., Tomasi, C., & Guibas, L. J. (2000). The earth mover's distance as a metric for image retrieval. *International journal of computer vision*, *40*(2), 99-121.