

# Deep Learning for Natural Language Processing

By Thibault Cordier

## #1 - Monolingual embeddings (/6)

In "nlp\_project.ipynb", we write functions for computing the nearest neighbors of any word, without using an external package. We build two classes for word vectors and bag-of-words vectors, such that you get the desirable outputs (see code).

## #2 Multilingual word embeddings (/4)

The goal is to find a mapping  $W$  that will map a source word space (e.g French) to a target word space (e.g English), such that the mapped source words will be close to their translations in the target space. For this, we need a dictionary of "anchor points". Here, we will use the identical character strings in both languages.

We can show that the solution of  $\operatorname{argmin}_{W \in O_d(\mathbb{R})} \|WX - Y\|_F$  has a closed form.

**Question:** Using the orthogonality and the properties of the trace, prove that, for  $X$  and  $Y$  two matrices:  
 $W^* = \operatorname{argmin}_{W \in O_d(\mathbb{R})} \|WX - Y\|_F = UV^T$  with  $U\Sigma V^T = \operatorname{SVD}(YX^T)$ .

$$\begin{aligned}
\operatorname{argmin}_{W \in O_d(\mathbb{R})} \|WX - Y\|_F &= \operatorname{argmin}_{W \in O_d(\mathbb{R})} \langle WX - Y, WX - Y \rangle_F && \text{with } W^T W = I \text{ because } W \in O_d(\mathbb{R}) \\
&= \operatorname{argmin}_{W \in O_d(\mathbb{R})} \|X\|_F^2 + \|Y\|_F^2 - 2 \langle Y, WX \rangle_F && \langle WX, WX \rangle_F = \langle X, X \rangle_F \\
&= \operatorname{argmax}_{W \in O_d(\mathbb{R})} \langle YX^T, W \rangle_F \\
&= \operatorname{argmax}_{W \in O_d(\mathbb{R})} \langle U\Sigma V^T, W \rangle_F && \text{with } \operatorname{SVD}(YX^T) = U\Sigma V^T \\
&= \operatorname{argmax}_{W \in O_d(\mathbb{R})} \langle \Sigma, U^T W V \rangle_F && \text{with } \operatorname{SVD}(YX^T) = U\Sigma V^T
\end{aligned}$$

Because  $U^T W V$  is an orthonormal matrix (as it is a product of orthonormal matrices), the previous expression is maximised when equals the identity matrix  $I$  (because  $\Sigma$  is diagonal).

$$\operatorname{argmin}_{W \in O_d(\mathbb{R})} \|WX - Y\|_F \iff U^T W V = I \iff W = UV^T \quad \text{with } \operatorname{SVD}(YX^T) = U\Sigma V^T$$

In "nlp\_project.ipynb", we create  $X$  and  $Y$  using the identical character strings in each language, compute  $W$  and output target nearest neighbors of source words in the shared space.

## #3 Sentence classification with BoV (/4)

In this section and the following, we obtain the train, dev and test sets of the Stanford Sentiment Treebank fine-grained sentiment analysis task. It consists of input sentences that we have to classify into 5 classes.

For the test set, we obtain the input samples, not the ground-truth labels. We have to produce our predictions using our best model, and send it. The quality of our predictions will be evaluated.

We use scikit-learn to learn a logistic regression on top of bag-of-words embeddings on the SST task.

**Question:** What is your training and dev errors using either the average of word vectors or the weighted-average?

### Using logistic regression model:

#### Using the average of word vectors:

Accuracy on train set: 49.36%	Accuracy on dev set: 40.42%
Error on train set : 50.64%	Error on dev set : 59.58%

#### Using the weighted-average of word vectors:

Accuracy on train set: 50.27%	Accuracy on dev set: 41.24%
Error on train set : 49.73%	Error on dev set : 58.76%

### Using linear neural network classifier model:

#### Using the average of word vectors:

Accuracy on train set: 47.46%	Accuracy on dev set: 42.69%
Error on train set : 52.54%	Error on dev set : 57.31%

#### Using the weighted-average of word vectors:

Accuracy on train set: 50.22%	Accuracy on dev set: 40.87%
Error on train set : 49.18%	Error on dev set : 59.13%

## #4 Deep Learning models for classification (/6)

**Question:** Which loss did you use? Write the mathematical expression of the loss you used for the 5-class classification.

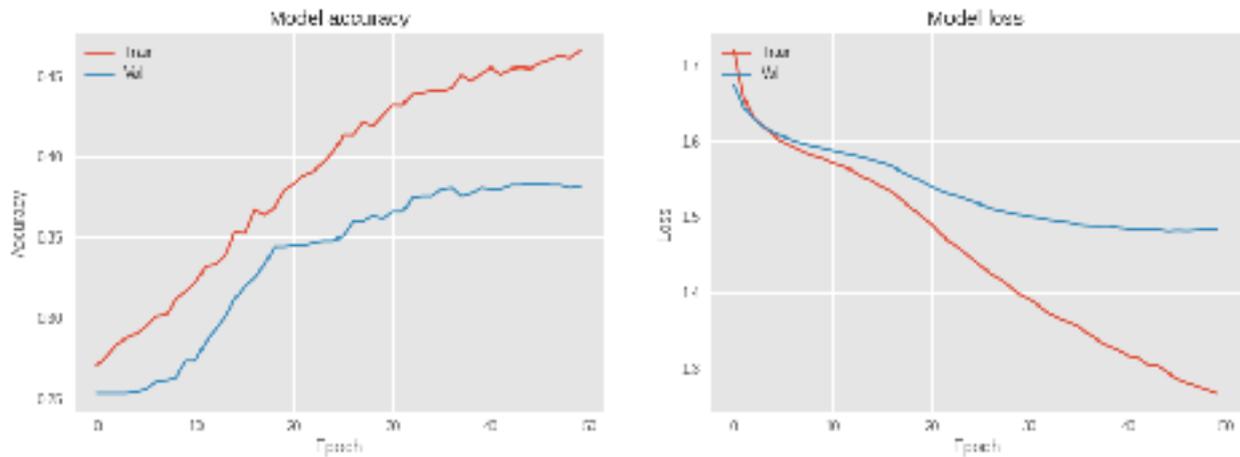
We used the “**categorical crossentropy**” loss which measures the performance of a classification model where the prediction input is a probability value between 0 and 1 for each category.

The mathematical expression of the loss we used for the 5-class classification is:

$$J(\theta) = -\frac{1}{N} \sum_{n=1}^N \sum_{k=1}^{K=5} y_{n,k} \log(\hat{y}_{n,k}(\theta)) .$$

with  $\theta$  the parameters of the model,  $y = (y_{n,k}) \in \mathcal{M}_{N,K}$  the one-hot encoding labels matrix and  $\hat{y} = (\hat{y}_{n,k}) \in \mathcal{M}_{N,K}$  the predictions of the model.

**Question:** Plot the evolution of train/dev results w.r.t the number of epochs.



*Evolution of accuracy and loss for train and dev set w.r.t the number of epochs.*

**Question:** Be creative: use another encoder. What are your motivations for using this other model?

We use a **self-attention encoder** for classification task on the Stanford Sentiment TreeBank (SST) set.

My motivations for using this model:

- This is a model of state-of-the-art.
- In June 2017, Google's machine translation team published a paper at arXiv entitled "Attention is All You Need" which use self-attention mechanisms. This model became a hot topic in neural network attention research and proved useful in a wide variety of tasks.
- In fact, I am just interested about the encoder part of "The Transformer - a model that uses attention to boost the speed with which these models can be trained."
- My goal is then to encode sentences into a feature encapsulating useful parameters for sentiment classification - like a common classifier model.
- See reference (<http://jalammar.github.io/illustrated-transformer/>)