Deep Learning for Natural Language Processing

Omar CHEHAB

MVA 2019

1 Monolingual Embeddings

We use a classes, organized essentially with the same functionalitis: embedding first, computing similarity scores, finding K-Nearest-Neighbor matches based on the latter.

The first building block is words, for which we have the class *Word2Vec*. The second building block is sentences, for which we have the class BoV built on top of *Word2Vec*.

We obtain the following results for word-matching and sentence-matching, respectively:

```
Loaded 25000 pretrained word vectors
cat dog 0.671683666279249
dog pet 0.6842064029669219
dogs cats 0.7074389328052403
paris france 0
germany berlin 0
['cat', 'cats', 'kitty', 'kitten', 'Cat']
['dog', 'dogs', 'puppy', 'Dog', 'canine']
['dogs', 'dog', 'Dogs', 'puppies', 'cats']
input word is not in word2vec loaded vocab
[]
input word is not in word2vec loaded vocab
[]
```

```
Loaded 5000 pretrained word vectors
Top 5 NNs of 1 smiling african american boy .
1/ 1 smiling african american boy .
2/ blond boy waterskiing .
3/ a boy jumps .
4/ a boy jumps .
5/ a boy smiles underwater .
[[0.60894451]]
Top 5 NNs of 1 smiling african american boy .
1/ 1 smiling african american boy .
2/ 5 women and 1 man are smiling for the camera .
3/ a man rides a 4 wheeler in the desert .
4/ 3 males and 1 woman enjoying a sporting event
5/ a man in black is juggling 3 flamed bottles .
[[0.59633357]]
```

Regarding words, the three first matches make semantic sense. The pairs with score 0 are explained by the absence of these words in our vocabulary. The list of nearest neighbors (in decreasing order) are also coherent with common use of language.

Regarding sentences, classic word-averaging seems over-emphasize the word 'boy', so that it appears in proposed matches without similar context. Idf-weighted averaging better aggregates different information (ex: 'boy' and 'smiling') that appear in proposed matches, but once again context is lacking. This motivates investigating better sentence-comparison structures, for example LSTM-based, that capture *context* and therefore *sequential information*.

2 MultiLingual Word Embeddings

Once we operate in *one* 'language space', the manner of proceeding is the same as above. So the goal is to *transport* this *two*-language problem to *one* space. The easiest means of transport is an orthogonal matrix, representing diluting-rotation of one space to match the geometry of another, setting up the *Procrustes* problem:

$$\begin{aligned} W^* &= \text{argmin}_{W \in O_d(\mathbf{R})} ||WX - Y||_F \\ &= \text{argmin}_{W \in O_d(\mathbf{R})} ||WX - Y||_F^2 \\ &= \text{argmin}_{W \in O_d(\mathbf{R})} ||X||_F^2 + ||Y||_F^2 - 2 < WX, Y > \\ &= \text{argmax}_{W \in O_d(\mathbf{R})} \text{tr}(WXY^T) \end{aligned}$$

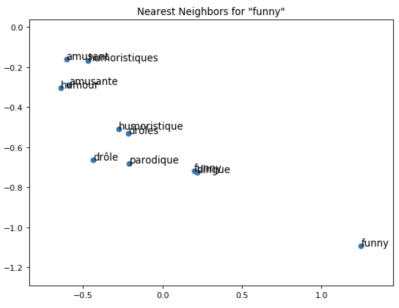
We introduce Singular Value Decompositions:
$W = U_1 \Sigma_1 W_1 \qquad \text{and} \qquad YX^T = U_2 \Sigma_2 W_2$.
By plugging them in and using the cyclic permutation invariance of the trace, we obtain:

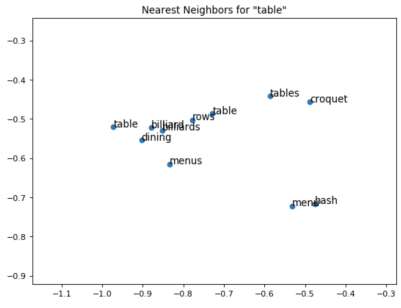
$$\begin{aligned} \text{tr}(WXY^T) &= \text{tr}((U_2^T U_1)\Sigma_1 (V_2 V_1)\Sigma_2)) \\ &\leq \text{tr}(\Sigma_1 \Sigma_2) \\ &\quad \text{using Von Neumann's theorem on the trace} \end{aligned}$$

The upper bound is reached for $U_2^T U_1 = I$ and $V_2 V_1 = I$, which leads to $W = U_2 \Sigma_1 V_2^T$. By invoking the characterization of 'W is an orthogonal matrix', $W^T W = I$, we obtain $\Sigma_1^2 = \Sigma_1 = I$, thus:

$$W = U_2 V_2^T$$

We then build a class *BilingWord2Vec* which, after aligning the input language to the output language space, reprises the generic architecture of *Word2Vec* (for words) and *BoV* (for sentences) to find matches.





-

3 Sentence classification with BoV

Using the sentence embedding provided by the *BoV* class, we put a classifier on top of it predict multi-class labels for sentiment analysis.

Using Logistic Regression as a baseline, we have:

Train Accuracy	0.455
Dev Accuracy	0.407
Train Accuracy (idf)	0.462
Dev Accuracy (idf)	0.386

Using a Random Forest to suggest an improvement margin, we obtain:

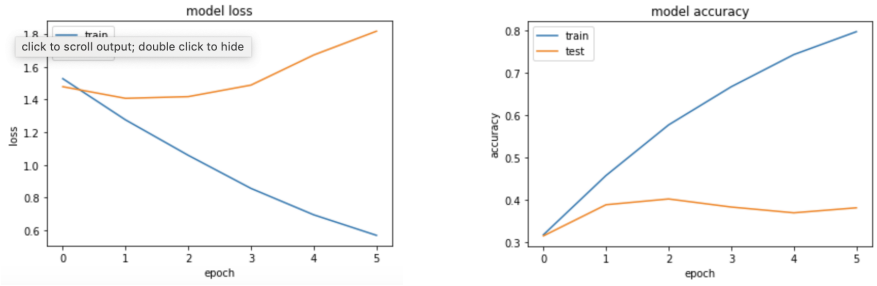
Train Accuracy	0.999
Dev Accuracy	0.338
Train Accuracy (idf)	0.999
Dev Accuracy (idf)	0.361

which performs on par with, if not more poorly than, the Logistic Regression.

4 Deep Learning Models for Classification

Given the sentiment analysis task is a classification problem, we use a *cross-entropy loss*: $L = -\sum_{x \in \mathcal{X}} p(x)\log(q(x))$, where:
x is an input sentence
$p(x)$ is the ground truth: a 5-dimensional vector one-hot encording of the ground truth (1 in position of the sentiment label, 0 everywhere else)
$q(x)$ is a soft-encoding of the model prediction: a 5-dimensional vector with probabilities of corresponding to each class

Using a simple LSTM-based Deep Learning model, we obtain as a first result:



Clearly, the model is overfitting, like the previously tested Random Forest.

We try a better-suited Deep Learning architecture, replacing the LSTM with a bi-directional LSTM (better at identifying context; used in BERT), pretraining with the crawl file, tuning optimization parameters, and adding a SVM classifier on top. The dev set accuracy is at 39.2% while the previous LSTM one was at 38.2%: the 40% cap still holds! However, this may be due to small dev-set size and we hope that this model holds better generalization capacity.