NLP project

1 Monolingual embeddings

Note In the function `most_similar` I included the option not to output the first most similar word to the query word w. This is useful for example, for the first task when we are asked to output the most similar words to "cat". It is not relevant or useful to have "cat" in the output. Indeed our system does not handle unknown words (I did not find it necessary to implement some sort of "word corrector" when a query word was not found for this homework), so the first most similar word will always be the query word.
However, when doing translations, if we want to find the best translation, the most similar translated word will usually be the one with highest similarity score, so we want to keep the whole list of propositions.
Also, note that our system is case sensitive.

2 Multilingual word embeddings

Question Using the orthogonality and the properties of the trace, prove that, for X and Y two matrices :

$$W = \text{argmin}_{W \in O_d(\mathrm{R})} ||WX - Y||_F = UV^T \text{ with } U\Sigma V^T = \text{SVD}(YX^T) \tag{1}$$

Answer $||.||_F$ is the Frobenius norm, defined by $||A||_F = \sqrt{\text{Tr}(AA^T)}$
First, we can note that $W = \text{argmin}_{W \in O_d(\mathrm{R})} ||WX - Y||_F = \text{argmin}_{W \in O_d(\mathrm{R})} ||WX - Y||_F^2$

$$\begin{aligned} ||WX - Y||_F^2 &= \text{Tr}((WX - Y)(WX - Y)^T) \\ &= \text{Tr}(WXX^TW^T) + \text{Tr}(YY^T) - 2 < WX, Y >_F \\ &= \text{Tr}(W^TWXX^T) + \text{Tr}(YY^T) - 2 < WX, Y >_F \\ &\quad \text{where we used the cyclic property of trace.} \\ &\quad \text{Moreover, as } W \text{ is orthogonal, } W^TW = I \\ &= ||X||_F^2 + ||Y||_F^2 - 2 < WX, Y >_F \end{aligned} \tag{2}$$

Hence, minimizing $||WX - Y||_F^2$ amounts to maximizing $< WX, Y >_F$.
Using Cauchy-Schwarz's inequality and the SVD decomposition of $YX^T = U\Sigma V^T$:

$$\begin{aligned} < WX, Y >_F &= \text{Tr}(YX^TW^T) \\ &= \text{Tr}(U\Sigma V^TW^T) \\ &= < U\sqrt{\Sigma}, \sqrt{\Sigma}V^TW^T >_F \\ &\leq ||U\sqrt{\Sigma}||_F ||\sqrt{\Sigma}V^TW^T||_F \\ &\leq ||\sqrt{\Sigma}||_F ||\sqrt{\Sigma}||_F \\ &\text{using invariance under orthogonal transformations} \\ &\leq \text{Tr}(\Sigma) \end{aligned} \tag{3}$$

This upper bound is reached when $\text{Tr}(U\Sigma V^TW^T) = \text{Tr}(\Sigma)$, so for W such that $\Sigma V^TW^TU = \Sigma$.
Hence $W = UV^T$ maximizes $< WX, Y >_F$.
We have proved that :

$$W = \text{argmin}_{W \in O_d(\mathrm{R})} ||WX - Y||_F = UV^T \text{ with } U\Sigma V^T = \text{SVD}(YX^T) \tag{4}$$

One funny example I'm just including an output that I found pretty hilarious but actually not that irrelevant (the other examples gave very accurate translations, so it was not as funny) :
Most similar words in French to : glitter

— aerosmith
— minogue
— minaj
— pretty
— smile

3 Sentence classification with BoV

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

Answer We use `scikit-learn` to learn a logistic regression on top of bag-of-words embeddings on the SST task, and tune the C parameter of the logistic regression to have the best score on the validation set. The best results are reported in 2. We choose C parameter such that the score on the dev set is highest. When using weighted-average we can notice that the score on the dev set does not vary much with C. Given identical dev scores, we choose C for which the difference between training and dev score is lowest (we want to minimize overfitting).

Table 1 – Best scores for logistic regression on train and dev set

	Model scores on training and dev data		
Model	Training	Dev	C parameter
Average of words	0.48162	0.44323	0.5
Weighted-average	0.50152	0.40962	0.2

We keep the model for which we have highest dev score, that is the model using average of words, with parameter C = 0.5. We can notice that the model with weighted-average overfits very much the training set, which could lead to very poor results on the test set.
For the prediction that we actually sent in for the `logreg_bov_y_test_sst.txt` file, we concatenates train and dev set to have a bigger training set. The score obtained for the training (which is now train + dev) is 0.48118. We can expect a score for the test set between 0.40 and 0.45 (we would be very lucky if we had a higher score, but maybe many examples in the test set are similar to those in the train or dev set).

Bonus question I also tried to use SVC and AdaBoost classifiers. The best result was obtained with SVC classifier using average of words.

Table 2 – Best scores for SVC classifier on train and dev set

	Model scores on training and dev data		
Model	Training	Dev	C parameter
Average of words	0.51158	0.45141	2
Weighted-average	0.52118	0.41871	0.1

4 Deep Learning models for classification

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

Answer I used the categorical cross-entropy loss as we have a multiclass problem. Cross entropy gives a measure of the distance between what the original distribution really is (our data) and what the model believes the output distribution should be. The mathematical formulation is

$$H(q,p) = -\sum_i q_i \log(p_i)$$

where q refers to the true distribution, and p the distribution given by our model.

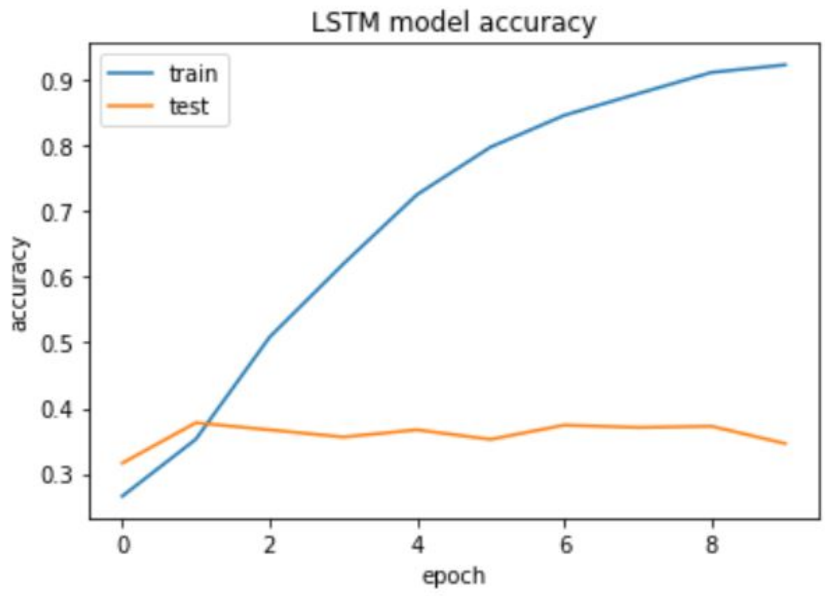
Question Plot the evolution of train/dev results w.r.t the number of epochs.

Answer We plot the evolution of train/dev results w.r.t the number of epochs in Fig. 1.After just a few epochs, we see that the loss on the dev set increases a lot while the accuracy converges very fast. The curves for train and test set diverge greatly after 3 epochs, so when fitting the model to use it to predict the test set later on, we will stop at 3 epochs. Using a LSTM layer with 128 units, and dropout rate 0.5 followed by a dense layer with 64 units and 0.5 dropout rate, we reached a maximum accuracy on the dev set of 0.3778 % after 2 epochs (we run the code several times and results would range 36-38%).

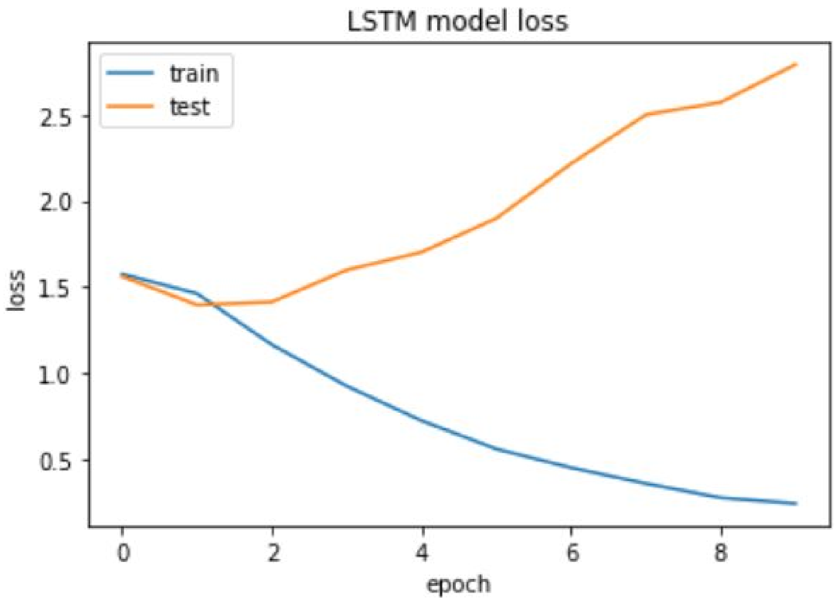
Question Be creative : use another encoder. What are your motivations for using this other model ?

Answer For this model, I did not use one-hot encoding, but rather, used keras `Tokenizer` and `texts_to_sequences` functions to build `X_train`,`X_dev` and `X_test`.
I used a bidirectional LSTM. This structure allows the networks to have backward and forward information about the sequence at each time step. This time I tested several Dense layers after the bidirectional LSTM layer but the best results were obtained without hidden Dense layer afterwards. We obtain slightly better results than before on the dev set : 39-41 % with this method, stopping after 4 epochs. We plot the evolution of train/dev results w.r.t the number of epochs in Fig. 2.

FIGURE 1 – LSTM model accuracy and loss on train and dev set

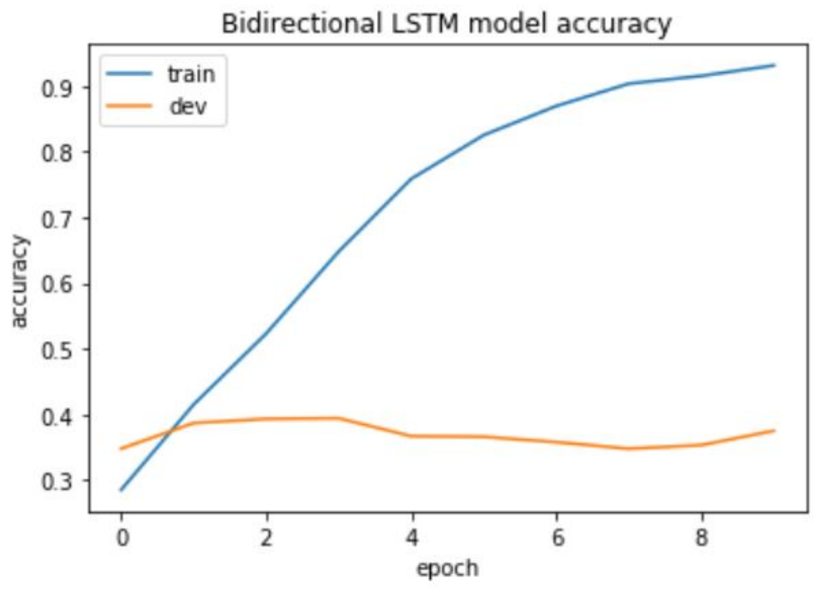


(a) Model accuracy w.r.t. number of epochs

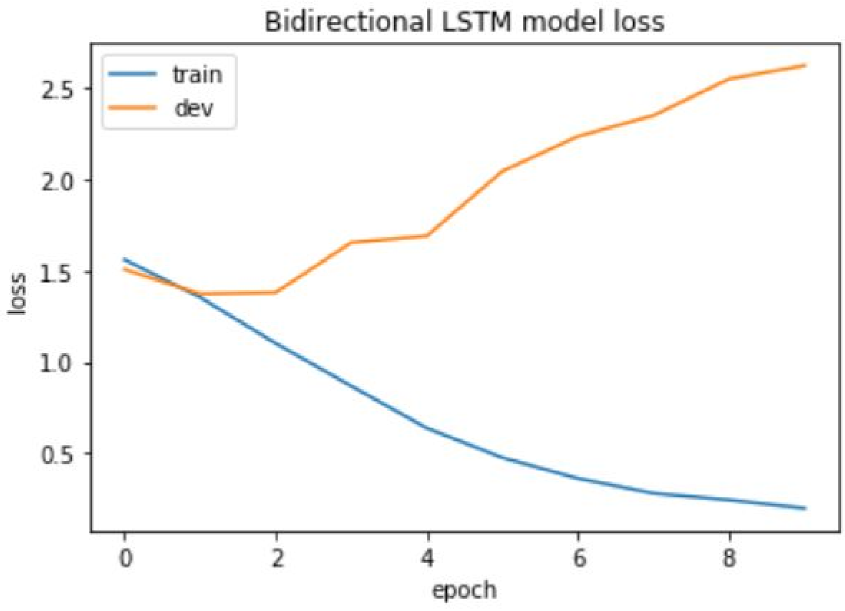


(b) Model loss w.r.t. number of epochs

FIGURE 2 – Bidirectional LSTM model accuracy and loss on train and dev set



(a) Model accuracy w.r.t. number of epochs



(b) Model loss w.r.t. number of epochs