NLP Project : Text Classification

1 Data presentation

We work on the Cooking Stack Exchange data set. This data set contains a training set (12404 observations), a test set (1500 obs) and a validation set (1500 obs). Each observation is a subject from the web forum Cooking Stack Exchange, and the labels are the tags linked to the subject. Each subject can have several tags, which will be the main issue in this project.

2 Logistic Regression classifier and One-vs-the-rest strategy

2.1 Embedding

A classical and easy method to produce text embedding is the TF-IDF (term frequency–inverse document frequency) followed by a PCR to reduce the dimensionality without losing to much information. It has been shown to be effective Salton et Buckley [1988], in spites of its simplicity.

2.2 Method Presentation

We are facing a multi-label problem since each subject can have several tags. For instance, the subject "A great book with recipes for Tapas" is tagged in "resources", "cookbook" and "spanish-cuisine". We can't use a multi-class classification approach, since it assumes several classes but a unique label per observation. To adopt a multi-label approach, we transform each label in a binary variable, so instead of a vector of 730 labels, we have a matrix (of dimension $nb_{observations} * 730$ filled in with 0 and 1. For line i, a 1 in columns "sushi" and "sashimi" and 0 everywhere else mean the labels of this observation or "sushi" and "sashimi".
We adopt a One-vs-the-rest (OvR) multilabel strategy, which means we fit a classifier for each label. We use a logistic regression as classifier, and we choose to optimise the hyper-parameters (penalty and regularisation) by a grid search on $['l1','l2'] * [0.01, 0.1, 1, 10, 100]$. We use the train set and test set given in the database for the cross-validation, and we try to maximise the precision at 1 (the True Positive rate) on the test set.

2.3 Results

We have the following results for the grid search in Table 1 :

c	0.01	0.1	1	10	100
L1	0.0	0.025	0.253	0.398	0.423
L2	0.0	0.003	0.102	0.268	0.323

Table 1: Cross validation results

Our preferred model is a regularization parameter $c = 100$ and a $L1$ penalty. We now estimate the performances of this model on the validation set, and we compare these performances to thus obtained on the training set and the test set. The Gross Loss (mean of absolute errors) is not very useful since the matrices are mostly filled by 0. Thus we use 5 other more adapted metrics : the Gross Loss, Precision at 1, Recall at 1, Correct Label rate : the rate of subjects with the corrects predicted labels (Correct LR), At least one correct label : the rate of subjects with at least one correct label predicted (One Correct LR) and without any label rate : the rate of subjects where we didn't predict any label (WoLabel rate). We can see the result on Table **??**

We see that the precision (the metric we optimised with the cross-validation) decreases from training to test, but it remains the same from test to validation test, thus we have a robust model. Regarding the quality of the predictions, we have a 42% precision at 1 on the validation set, and only 15% of the subjects have the exact correct label. But for more than 70% of the subjects the model predicted at least one good label. To improve the predictive power of the model, we could consider choosing only the label predicted with the biggest probability.

	Gross Loss	Precision	Recall	Correct LR	One Correct LR	WoLabel rate
Train Set	0.0	1.0	0.0	1.0	1.0	0.0
Test Set	0.003	0.42	0.001	0.13	0.71	0.15
Validation Set	0.003	0.42	0.001	0.15	0.72	0.14

Table 2: Results of the log-regression

3 Recurrent neural network

3.1 Words embedding - pretrained model

The embedding could be calculated directly by initializing at random the embedding layer and calculating the weights while doing the task. Notwithstanding, this require an extremely big dataset and a lot of computation to be really efficient. That is why, as for most of NLP related tasks since Tomas Mikolov et Dean [2013], we chose to use pre-trained embeddings. We picked a classic one, the GloVe 6B Pennington *et al.* [2014], with an embedding dimension of 100.

3.2 Bi-directional LSTM with Attention mechanism

We first started with a very classical neural network for this kind of task : the first layer use the embedding from GloVe Pennington *et al.* [2014] and send the word representations in a bidirectional LSTM. We added some dense layers and use a sigmoid final activation with as many units as classes to predict. Indeed, as the objective is a multi-class multi-label problem, we can not operate with functions such as softmax. If the probability of "wine" is getting higher it would decrease the probability of "french-cooking", while both are totally possible at the same time.

Then we added self attention on sentence level after the recurrent layers as proposed in Ashish Vaswani [2017] with only 1 head at first. It is quite computationally intensive as it has a complexity of n^2. Nevertheless it yields good results in the literature and the dataset is relatively small, so it remained a totally acceptable computational cost.

Also, we saw in Ashish Vaswani [2017] that they used multiheads layers for attention mechanism, therefore we increased the number of heads of our self attention from one to five (initializing as recommended in Glorot et Bengio [2014] with different seeds and concatenating the outputs). The results were then quite conclusive as for 2 heads as we can see on Table **??**

3.3 Results

	Gross Loss	Precision	Recall	Correct LR	One Correct LR	WoLabel rate
Train Set	0.0026	0.29	0.0004	0.11	0.55	0.31
Test Set	0.0027	0.25	0.0004	0.09	0.48	0.35
Validation Set	0.0028	0.25	0.0005	0.08	0.50	0.35

Table 3: Results of the LSTM

4 Conclusion

The task of multi-label classfication here is hard, because for not only we have to find several labels for each sentences, but moreover they are hundreds of labels possible (with our intersection we had 731 labels). Many of those labels don't appear in the test and validation set, and some labels do not exist in the train while they exist in the validation and test set. To answer this task, we can observe on the results that the logistic regression performs better than the LSTM in all three of the training, testing and validation phases. There are a few explanations, we didn't optimize very well the LSTM, the architecture is not really optimized for the task, and the hyper parameter are not fined tuned as much as possible. In addition, once we vectorized the words, the task is relatively easy to answer and a logistic regression can totally outperform a neural network with a dataset of this size. For all this reason, we can conclude that for our setup, the logistic regression seems to answer better the problem.

References

ASHISH VASWANI, e. a. (2017). Attention is all you need.

GLOROT, X. et BENGIO, Y. (2014). Understanding the difculty of training deep feedforward neural networks.

PENNINGTON, J., SOCHER, R. et MANNING, C. D. (2014). Glove: Global vectors for word representation. *In Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

SALTON, G. et BUCKLEY, C. (1988). Term-weighting approaches in automatic text retrieval.

TOMAS MIKOLOV, Kai Chen, G. C. et DEAN, J. (2013). Efcient estimation of word representations in vector space.