A model for jointly tagging and classifying text

Pau Batlle Franch

This text is the continuation of [1], both of which explain my research as an intern of BBVA Analytics under the supervision of José A. Rodríguez-Serrano.

1 Tag prediction and classification

The problem that this article aims to solve is the joint classification and tag prediction for short texts. It arises in practical applications such as categorizing product descriptions in ecommerce sites [2], classifying names of shops in business datasets or organizing titles / questions in online discussion forums [3]. In applications related to banking, this problem can appear in transaction categorization engines, for instance as part of personal finance management applications, where the problem is to assign a category, and possibly tags, to an account transaction based on the words appearing in the transaction.

In all those cases, entities such as products, shops or post titles normally have a main category and can have many tags. Predicting categorical outputs based on a short text can be tackled with deep learning techniques [4]. However, both problems are not fully independent, since one may want to impose consistency on category and tags. Therefore, a challenge is how to design a model able to jointly learn tags and categories, given some input sentences (and possibly some known tags).

Additionally, there are situations where *explainability* is required, and thus we need insights of *why* a short text was assigned a given class. In the case of text classification, one example is pointing to the words that carried most of the information for classification. In the case of text classification, this is typically achieved through attention models [5, 6]. Again, a challenge would be how to add explainability to joint category and tag prediction.

The final goal was to explore a neural network architecture that takes as input a sentence and optionally set of known tags, and is able to

- Classify the sentence, taking into account the observed words and tags
- Predict missing tags, based on observed words and the other tags
- Score the input words and tags by importance; i.e. quantify or *explain* how much each observed word or tag contributes to the decision.

We were not aware of any existing short sentence classification method with the three above properties.

The model presented here uses a first attention model to build an embedding of a sentence in terms of the input words and tags. This embedding is used to score concepts from a fixed concept vocabulary. Finally, the embeddings of the scored concepts are pooled into a final sentence representation that can be used for scoring. The concept scoring can be interpreted as a second attention mechanism, where the input sentence is "reconstructed" in terms of a vocabulary of known concepts. Therefore, with this model we obtain classification, but also interpretability in terms of input words and tags, and induced tags.

2 Problem statement and proposed model

The interest relies in solving a double problem: text classification and tag predicition, given an input sentence.

Therefore, datasets suitable for this problem include text pieces $\{S_j\}$, a set of tags for each text piece $\{T(S_j)\}$, which includes a variable number of tags (or even no tags), and a single label for each text piece, $\{L(S_j)\}$, where j ranges from 1 to the number of elements of the dataset. We define a *class label* (in the following, shortly *label*) as a single, coarse-grain category name the sentence belongs to; and we define a *concept tag* (shortly, *tag*) as a set of zero, one or multiple descriptive concepts the sentence can refer to.

Our goal is to simultaneously learn T, which predicts tags given one of the text pieces, and L, which predicts the label of the piece of text. Throughout this text, the following notation will be used: we refer to the j^{th} word in the vocabulary as w_j and to the k^{th} word in a particular sentence as $w_{(k)}$. This same convention applies for tags as well, taking into account that the vocabularies of words and tags are disjoint sets.

2.1 Model specification

The model proposed to solve the aforementioned task of learning L and T consists of a joint attention mechanism and will be explained in this section. A representation of the model can be found in figure 2.

The inputs of the model during the learning stage are a sentence S, consisting of a set of words $w_{(1)}, ..., w_{(|S|)}$ and its tags, $t_{(1)}, ..., t_{(|T(S)|)}$, which are padded to $w_{(1)}, ..., w_{(M)}$ and $t_{(1)}, ..., t_{(N)}$ with M and N fixed for all the dataset. Every word and tag is now embedded into a $\mathbb{R}^d$ vector using a word embedding function $e(\cdot)$, which can either be learned during training or pre-trained using an unsupervised embedding method like Word2Vec [7] or GloVe [8]. It is possible for a particular word to appear both in the word vocabulary and the tag vocabulary but its embedding as a word and as a tag are learned independently and may differ.

These inputs now go through a self-attention architecture, similar to the one described in [5]. Here, we define a vector $\lambda \in \mathbb{R}^{M+N}$ as $\lambda = \text{softmax}(U^t E)$, where E is the $d \times (N+M)$ matrix with $E = [e(w_{(1)}), ..., e(w_{(M)}), e(t_{(1)}), ..., e(t_{(N)})]$ and U is a $d \times 1$ matrix of trainable parameters. The vector λ represents the first learned attention weights of the model. The value of λ_j is interpreted as the importance of the word (j) (or the tag $(j - M)$ if $j > M$) in the sentence.

We then form a first vector representation of the sentence, s using these attention weights as $s = \sum_{i=1}^{N} \lambda_i e(w_{(i)}) + \sum_{j=1}^{M} \lambda_{N+j} e(t_{(j)})$. s mantains the embedding dimension d of the words and tags.

If we only had to classify the sentence, we could feed s directly to a dense layer outputting class weights. But the crucial part of the model is that *we will force an intermediate representation in terms of concept embeddings.* Therefore, s is then fed into a dense layer of input dimension d and output dimension $|\mathcal{T}|$, this is, the number of tags in the tag vocabulary $\mathcal{T}$. These tags can undergo another self-attentive layer, which after a softmax layer outputs μ, a vector in $\mathbb{R}^{|\mathcal{T}|}$, whose components μ_k can be interpreted as the attention that this sentence gives to tag t_k in the vocabulary. Therefore, all tags in the tag vocabulary are scored with a probability of them appearing as a tag for the input sentence. A second reconstruction of the sentence s' is now created by adding the embeddings of all the tags of the tag vocabulary weighted by the second attention vector μ, such that $s' = \sum_{j=1}^{|\mathcal{T}|} \mu_j e(t_j)$.

This reconstruction is only made using the tags of the tag vocabulary, i.e. is an interpretation of the input sentence in terms of previously known concepts. This means the model will be "forced" to activate some tags during test time that did not appear in the input (e.g. the word "pizza" might activate the concept "food" even if not explicitly tagged so), even in the case of zero input tags (note however, that some tags are necessary at training time to capture word-tag-label relations.

Finally, we can predict class labels based on the intermediate concept representation, by feeding s' to another dense layer which, after a softmax layer, outputs the class probabilities p_k in a vector $\hat{y}$.

As a summary, for a given input sentence, our model will output the prediction scores in each category, the attention scores input words and tags, and the attention scores of the intermediate concepts.

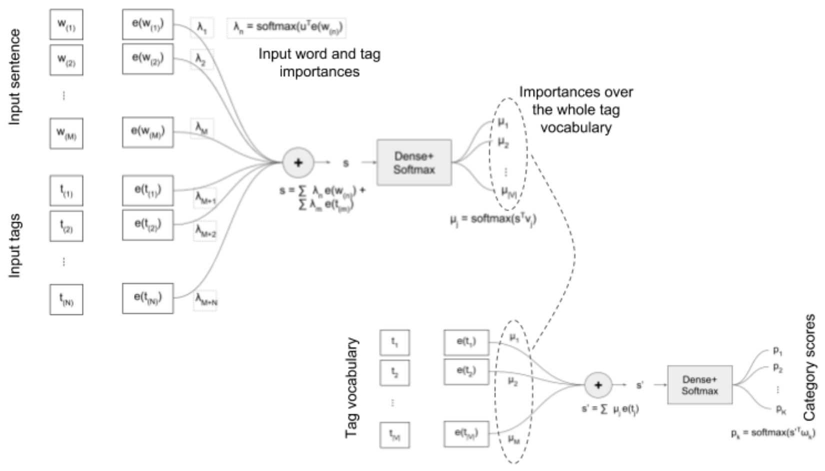
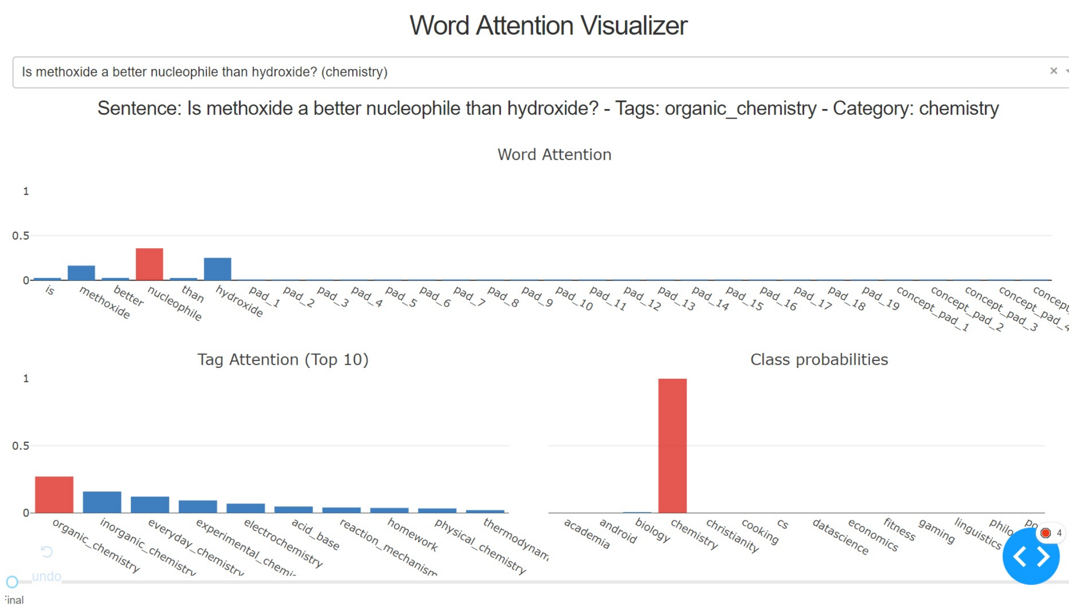

Input sentence
Input tags
Input word and tag importances
Importances over the whole tag vocabulary
Dense+ Softmax
Tag vocabulary
Category scores

Figure 1: Overview of the model

3 Discussion and Training insights

In this section some examples of results of the work are visualized with a tool which was coded as part of the internship. In particular, what is shown here is the output of the different layers of the model for some unseen text. The dataset used was composed of different questions from the StackExchange network, with their own categories and tags and therefore what we use as an input are questions from the same site as well. In the top row one can see the attention the model gives to each of the words of the sentence, the bottom left shows the activation values of different tags and the bottom right shows the activation for each of the classes.

In the first example, see how the model puts attention almost entirely on ”methoxide”, ”nucleophile” and ”hydroxide”. These are indeed the most relevant words in the sentence, and are used to correctly classify the sentence into ”chemistry” and to correctly predict the tag ”organic chemistry”.

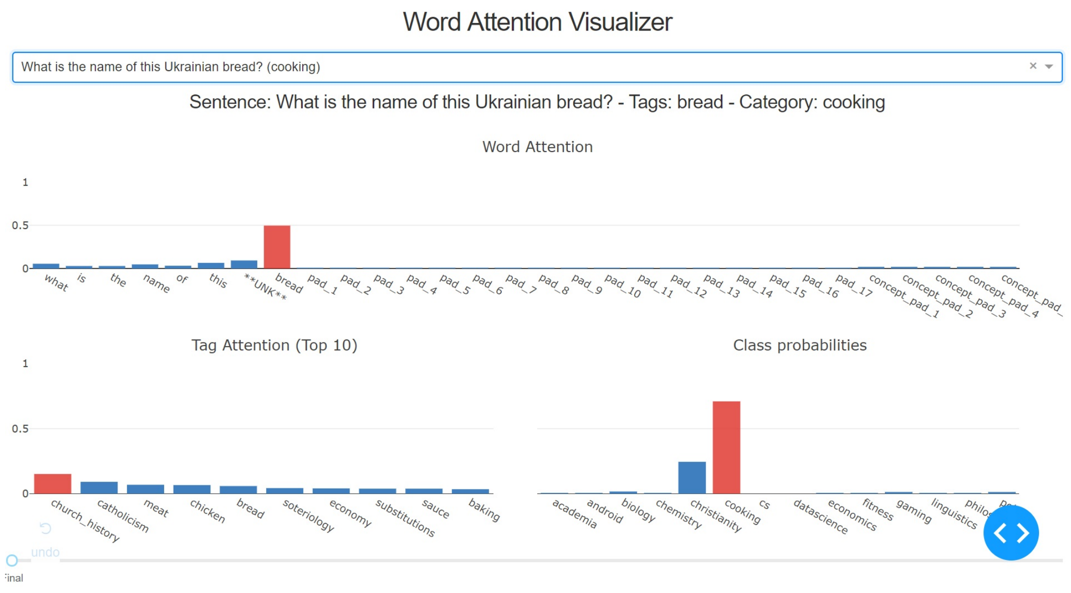

Word Attention Visualizer

Is methoxide a better nucleophile than hydroxide? (chemistry)

Sentence: Is methoxide a better nucleophile than hydroxide? - Tags: organic_chemistry - Category: chemistry

Figure 2: First example of unseen sentence

The second example is a bit more interesting. Since the word ”Ukranian” was not part of the vocabulary, it is replaced by a wildcard token, and most of the attention is therefore put on the word “bread”. While the predicted class is correctly “cooking”, note the activations of the class “Christianity” and religion-related tags, presumably due to the learned connection between bread and Christianity as observed in some training samples. A longer sentence with other significant words (e.g oven) could have made the model more sure about the sentence being about cooking and not about Christianity.


Word Attention Visualizer

What is the name of this Ukrainian bread? (cooking)

Sentence: What is the name of this Ukrainian bread? - Tags: bread - Category: cooking

Figure 3: Second example of unseen sentence

Finally, the third example shows a wrongly classified sentence. Note how the presence of the word ”sentence” tricks the model into thinking that the sentence is about linguistics (and tags it as phonetics). While it wrongly gives high confidence about the linguistics class, reading only the sentence (which is the only model input) it is not clear whether the original poster wanted a linguistics or data science answer, in my opinion, so it's an acceptable mistake.

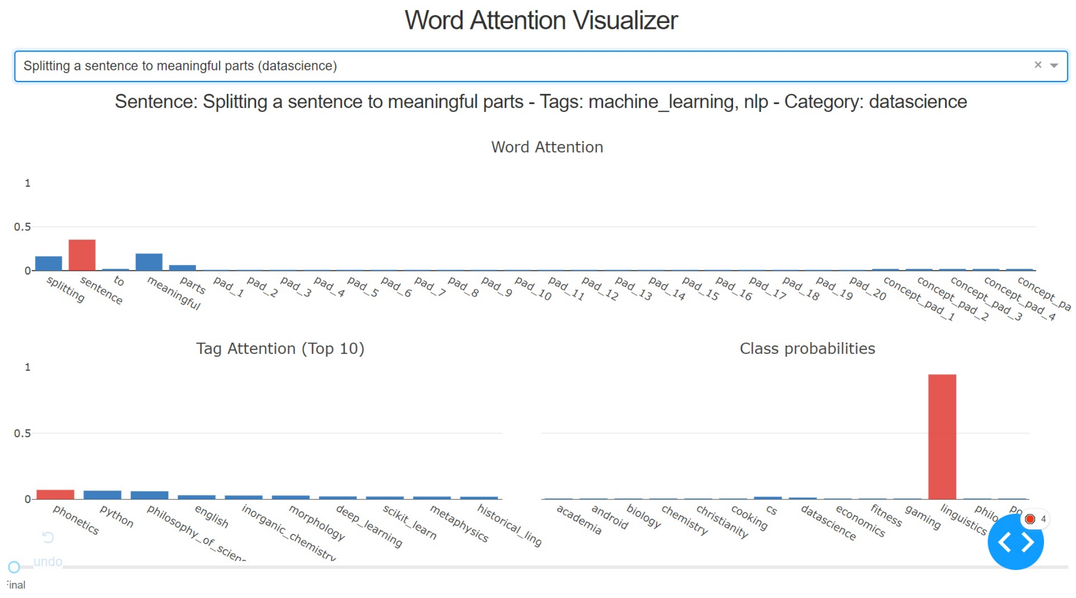


Figure 4: Third example of unseen sentence

4 Conclusions and Acknowledgments

In this article we have discussed attention models as a possible solution for tackling text classification and tag prediction at the same time. We have proposed a model and shown some examples of its performance in unseen text. However some more tweaks to parameters and benchmarking should be performed, which could not be performed during the duration of the internship due to time constraints. As in my previous post, I would like to thank my supervisor José Antonio Rodríguez for its patience and time supervising both of my internships, and for the opportunity of doing the internships in the first place.

References

[1] P. B. Franch, "Financial text classification: an analysis of different methods for word embedding."

[2] J.-W. Ha, H. Pyo, and J. Kim, "Large-scale item categorization in e-commerce using multiple recurrent neural networks," in *KDD*, 2016.

[3] C. Stanley and M. D. Byrne, "Predicting tags for stackoverflow posts," in *ICCM*, 2013.

[4] J. Y. Lee and F. Dernoncourt, "Sequential short-text classification with recurrent and convolutional neural networks," in *NAACL-HLT*, 2016.

[5] Z. Lin, M. Feng, C. Nogueira dos Santos, M. Yu, B. Xiang, B. Zhou, and Y. Bengio, "A structured self-attentive sentence embedding," *arXiv preprint arXiv:1703.03130*, 2017.

[6] A. Giannakopoulos, D. Antognini, C. Musat, A. Hossmann, and M. Baeriswyl, "Dataset construction via attention for aspect term extraction with distant supervision," in *ICDMW*, pp. 373–380, 2017.

[7] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," 2013.

[8] J. Pennington, R. Socher, and C. D. Manning, “Glove: Global vectors for word representation,” in *Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1532–1543, 2014.