Concept Tagging for the Movie Domain by using Transfer Learning and Named Entity Recognition.

Giovanni De Toni (197814)
University of Trento
Via Sommarive, 9, 38123 Povo,Trento TN
giovanni.detoni@studenti.unitn.it

Abstract

Neural Networks have produced remarkable results in many Natural Language Processing tasks, for example, when tasked to assigning concepts to words of a sentence. Their successes are made possible by employing good word representations (embeddings) which a Neural Network can understand. This work evaluates several newly developed pre-trained embeddings (ELMo, BERT and ConceptNet) on the task of tagging sequences from the movie domain. We then compare the measurements with previous results of the literature.

1 Introduction

Concept Tagging sequences is a classical problem in the NLP field. It consists of assigning to each word of a sentence a given concept which represents the meaning of that word (e.g., "*star wars*" is a movie title, hence the concept " *movie.title*"). Over the past years, it was extensively studied and several techniques were developed such to efficiently tag words given a pool of concepts. The most basic methods are statistical language models (e.g., Weight Finite State Transducers) which are easy to train and yield quite impressive results with a little tuning. However, with the rise of Recurrent Neural Networks, we can now develop models which have better performances than their statistic counterparts. Moreover, thanks to the use of specific word embeddings (e.g., word2vec, GloVe, ELMo etc.), we can boost the performances even more. This work takes as a starting point a previous paper on the topic by Gobbi et al. (2018) and tries to improve the results of some of the neural models presented by adding new features (NER and POS tags) and by trying to employ more intelligent embeddings (ELMo, BERT and ConceptNet). We conducted several experiments with a few selected architectures and we tried to outperform the performances presented (with respect also to the classical WFST methods). The report is structured as follows: the second section describes the models used for the experiments, the third section gives a brief explanation about the possible types of embeddings and then it describes the chosen embeddings for our project. The fourth section describes the analysis of the dataset and of the embeddings. Ultimately, the fifth and the sixth sections report the experiments performed and the discussion over the results.

2 Models

To evaluate how the performance changes with respect to the previous results we selected two models from the original work. We concentrated on the Neural Networks approaches by selecting the ones which held the best results over the test set. We chose the following architectures: LSTM and LSTM-CRF. Moreover, we extended the features used by the various models by providing also the **Part-of-Speech tags (POS)** and the **Named Entities Recognition (NER)** entities as one-hot-encoded vectors. We now describe the two selected architectures.

Long-Short Term Memory (LSTM) (Hochreiter and Schmidhuber, 1997) is a model which extends and improve the previous RNN by solving the vanishing/exploding gradients problem. We tested a simple LSTM which receives in input the embeddings of the words. Moreover, we tested another LSTM model which uses also the characters embeddings (CHAR) obtained using a convolutional layer concatenated with the new features.

We experimented also with the **LSTM-CRF** model (Yao et al., 2014; Huang et al., 2015) in which the LSTM provides the class scores for each token and then the Viterbi algorithm is used to determine the labels at a global level. Moreover, an

LSTM-CRF version extended with the character embeddings and the new features was also evaluated.

3 Embeddings

As a first step to improve the performances of the previous work we concentrated on evaluating several recently proposed pre-trained language models to produce words representations.

Generally speaking, language representations can be either **context-free** or **contextual**. Context-free means that the word representation is generated without looking at the context in which the word is used. For instance, the word "*season*" may have different meanings with respect to the context in which it is placed (e.g., "*Summer is my favourite season*" and "*Season the chicken and then serve*"), but in a context-free model it will get the same representation (embedding). A contextual representation takes care also of the context in which a word is used. Moreover, the contextual representations can be further divided into two other categories: unidirectional or bidirectional.

A **unidirectional model** contextualizes a given term by just looking at the words on the right (or on the left) of it.

A **bidirectional model** looks both on the left and on the right of the target word before producing a representation. There are also **shallow bidirectional models** which combine two unidirectional models (one for the left side and one for the right side) to give a more complete representation. In our work, we tested three different embeddings, one for each category (context-free, contextual shallow bidirectional and contextual bidirectional): ConceptNet, ELMo and BERT. We then compared their performances against the one used in the original work called **w2v_trimmed** (a standard word2vec representation).

3.1 ConceptNet

ConceptNet Numberbatch (Speer et al., 2017) is a context-free language representation. It is a set of word embeddings and it was made by combining previously existing models, namely: GloVe (Pennington et al., 2014), word2vec (Mikolov et al., 2013) and OpenSubtitles 2016 (Lison and Tiedemann, 2016). Moreover, it leverages the data of the ConceptNet knowledge graph to enrich its representation. The authors reached this goal by using a retrofitting procedure to adjust the word embedding matrix by using their knowledge graph. Thanks to this procedure, ConceptNet is also intrinsically multilingual since this method pushes the model to use the same embedding space both for English words and words in different languages (since they share the same meaning). This work uses the Numberbatch 19.08 English-only model [1] which contains 516782 embeddings with a dimension of 300.

3.2 ELMo

ELMo (Embeddings from Language Models) (Peters et al., 2018) is a deep contextualized word representation which captures both syntactic, semantic and polysemy characteristics of a given word. It uses a deep shallow bidirectional language model (biLM) trained on a large corpus by looking at the entire input sentences. The generated representations are deep because they are a function of all the layers of the biLM. Moreover, each layer models certain features of the given word. The original work states that the higher-level LSTM layers capture context-dependent aspects of the word meaning, while lower-level states capture aspects of the syntax. We used the pre-trained small ELMo model [2] with the embedding dimension of 1024. In this work, the the final embeddings were produced by **linear combination** from the ELMo hidden layers. If $\mathbf{h_i}$ denotes an ELMo hidden layer, the final embeddings $\mathbf{e}$ can be computed by doing $\mathbf{e} = \sum_{i=0}^{N} \mathbf{h_i} \cdot w_i$, where N is the number of layers. We devised two strategies to estimate the scalar weights w_i:

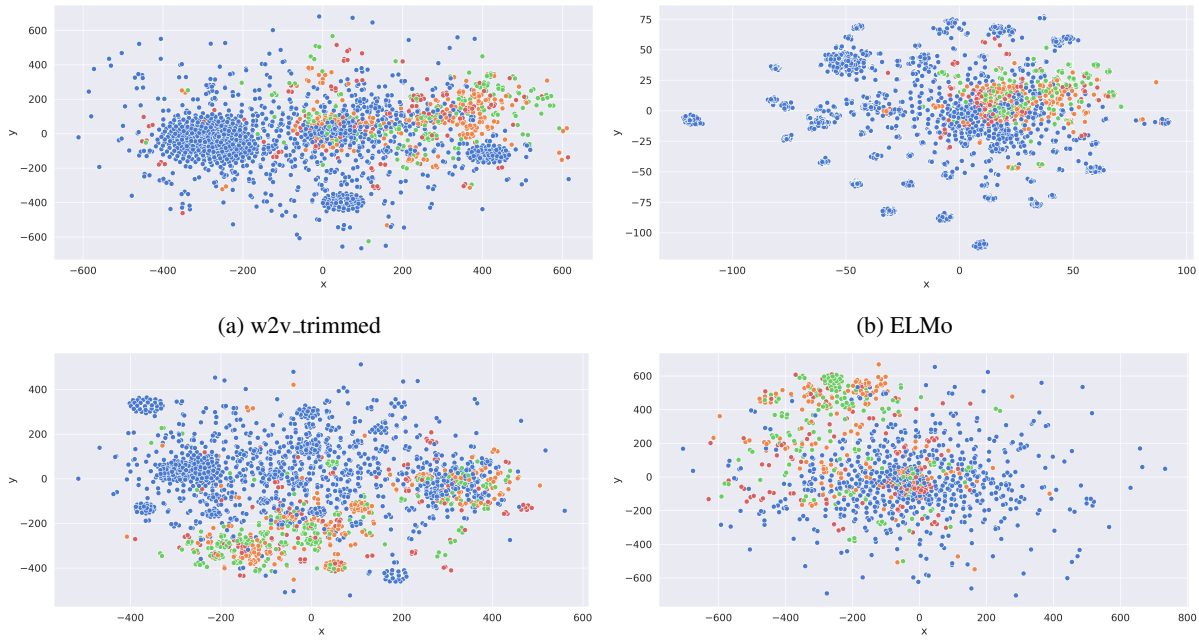
- **Averaged**: $w_i = \frac{1}{N}$ to give an equal weighting;
- **Fine-tuned**: it learns directly all w_i during the training of the final concept tagger.

3.3 BERT

BERT (Bidirectional Encoder Representations from Transformers) (Devlin et al., 2018) is a relatively new language representation model released in 2019. It pre-trains deep bidirectional representations from unlabeled texts by exploiting both the left and right context. The model is a multi-layer bidirectional Transformer encoder. The authors state that this new representation solves the previous limitations of pre-trained models (e.g., ELMo). They argue that the standard

[1] https://github.com/commonsense/conceptnet-numberbatch

[2] https://allennlp.org/elmo



(a) w2v_trimmed

(b) ELMo

(c) BERT

(d) ConceptNet

Figure 1: These plots show the embeddings representation of four major concepts of the training dataset. Namely, *movie.name* (blue), *actor.name* (orange), *producer.name* (red) and *director.name* (green). We can clearly see how some embeddings enable nicer clusters for some concepts.

LMs are unidirectional, thus limiting the kind of architectures which can be used during training. They also state that other pre-trained embeddings like ELMo are just shallow bidirectional since they use several separate models to capture the left and right context. There are two steps in the BERT framework: *pre-training* and *fine-tuning*. During the first phase, the model is trained on unlabeled data over two separate unsupervised training tasks: *Masked LM* and *Next Sentence Prediction (NSP)*. The Masked LM task consists of predicting some words of the given input phrase which are masked beforehand. This would enable to train a real deep bidirectional representation since bidirectional conditioning on standard LM is not possible. The Next Sentence Prediction (NSP) consists of predicting relationships between sentences which are usually not captured by LM. During the fine-tuning phase, the model is initialized with the pre-trained parameters obtained in the previous stage and it is fine-tuned on the desired task. In our work, we used a pre-trained BERT model[3] to get the word embeddings directly without doing the fine-tuning phase. More specifically, the BERT model used was pre-trained on the BookCorpus [4] and English Wikipedia. The embedding dimension was 768.

4 Data Analysis

We evaluated our solutions by using the NL2SparQL4NLU dataset [5]. It corresponds to the MOVIE dataset of the original paper. It includes sentences taken from the movie domain (e.g., "*trailer for star wars a new hope*"). The dataset contains 3338 and 1084 sentences for the train and test set respectively. The dictionary size (# of unique tokens) is of 1728 (train) and 1039 (test) with an OOV rate between the datasets of 24%. The average sentence length is 6.42 (train) and 6.52 (test). We have a total of 23 concepts (with 43 concepts in IOB format). Each word was also POS tagged and there are 50 different tags between the training and test set (taken from the Tree Tagger POS list[6]). We generated a new dataset by analyzing each sentence using a Named Entity Recognition (NER) tool, spaCy[7]. It found 398 (1,86%) and 169 (2,37%) entities inside the training and test set respectively. These entities

[3] https://pypi.org/project/bert-embedding/

[4] https://yknzhu.wixsite.com/mbweb

[5] https://github.com/esrel/NL2SparQL4NLU

[6] https://courses.washington.edu/hypertxt/csar-v02/penntable.html

[7] https://spacy.io/

Model	Hidden	Epochs	Batch	Lr	Drop rate	Emb	Min F_1 / Mean F_1 / Best F_1
LSTM	200	15	20	0.001	0.7	w2v_trimmed	81.88 83.17 84.35 82.23 **83.35** 84.22
LSTM	200	15	20	0.001	0.7	ConceptNet	81.58 82.49 83.75 81.78 **82.94** 84.11
LSTM	200	30	20	0.001	0.7	BERT	77.06 79.10 81.39 80.51 **82.13** 83.66
LSTM	200	30	20	0.001	0.7	ELMo	81.77 82.65 83.48 83.77 **84.40** 85.35
LSTM	200	30	20	0.001	0.7	ELMo (fine tuned)	75.19 82.29 85.14 82.88 **84.47** 85.60
LSTM-CRF	200	10	1	0.001	0.7	w2v_trimmed	84.85 **85.70** 86.83 84.66 85.35 86.30
LSTM-CRF	200	10	1	0.001	0.7	ConceptNet	85.46 **85.94** 86.52 85.49 85.83 86.51
LSTM-CRF	200	20	1	0.001	0.7	BERT	82.33 83.73 84.95 83.10 **83.83** 84.98
LSTM-CRF	200	20	1	0.001	0.7	ELMo	84.50 **85.53** 86.39 84.12 85.29 86.29
LSTM-CRF	200	10	1	0.001	0.7	ELMo (fine tuned)	84.34 85.72 86.56 83.96 **85.81** 86.81

Table 1: Evaluation results of the various models. Each model is bidirectional. The performances were obtained by recording the results for 25 runs by varying the initialization parameters of the models and by keeping their hyperparameters fixed. The rightmost column is divided as such: the first line shows the results without any improvements and the second line shows the results by adding the character convolution (CHAR) and the NER and POS tagging features. The **bold** values indicate the best result for that particular embedding.

were of 9 different types[8].

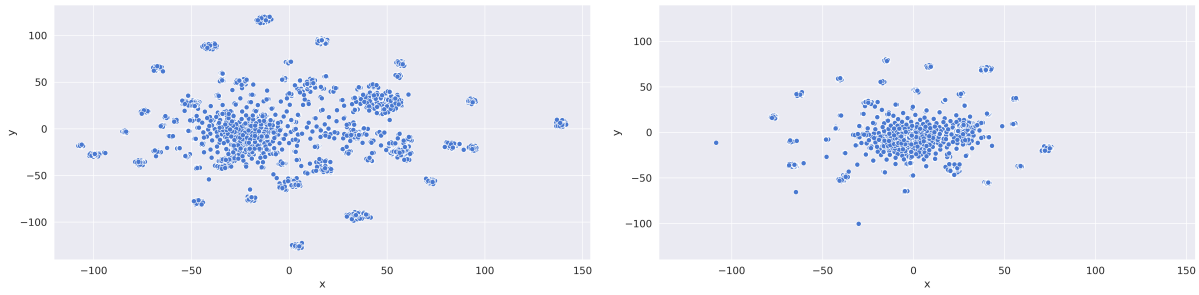
4.1 Embeddings Analysis

We evaluated also the various embeddings using the *embedding coverage* metric (namely, how many words in the dataset dictionary can be found in the embedding matrix) and we compared it with the embeddings used in the original work. With the default embeddings (w2v_trimmed), the coverage is 96.18% (66 words not recognized) while with ConceptNet the coverage is 91.90% (140 words not recognized). For BERT and ELMo models we performed a visual analysis instead by using dimensionality reduction techiques. More specifically, we used *t-SNE* (van der Maaten and Hinton, 2008) to reduce the dimensionality of the BERT/ELMo embeddings down to 2-dimensional data points and then we plotted the results for the four most represented concepts in the dataset. See Figure 1 for the results. It is possible to notice how the default embeddings, ELMo and BERT produce nice clusters for the *movie.name* category. The other categories are slightly mixed together which is expected. For instance, *actor.name* and *director.name* may end up with the same embedding since a person name can be referring both to an actor and to a movie director. BERT and ELMo should be able to discriminate between them since they can exploit the context, but the lack of clusters could be caused by the *t-SNE* reduction. Ultimately, ConceptNet does not produce significant clusters (apart from a small one composed by *director.name* elements). This however could be again caused by the *t-SNE* reduction.

5 Experiments

We evaluated the models by doing a hyperparameter search by starting from the values used in the original work. We tried to find the hyperparameter combination which maximizes the mean F_1 score over the test dataset. The final results can be seen in Table 1. When using a classical LSTM model, it is clear that by combining the character representation (CHAR) and the additional features (POS+NER) we obtain better results than the default baseline. This happens regardless of the embeddings employed. Moreover, the ELMo embeddings provide better performances than the other competitors. The BERT embeddings give instead less performing results (more than 3% difference between the F_1 mean

[8] https://spacy.io/api/annotation#named-entities



(a) Default ELMo Embeddings (*movie.name* only)

(b) Fine-Tuned ELMo Embeddings (*movie.name* only).

Figure 2: These plots show how the embeddings clustering for the *movie.name* concept changes between the averaged version and the fine-tuned one. Fine-tuning the weights gives importance on the higher layers which capture context-dependent aspects of word meaning and therefore they help to produce better clustering.

baseline in some cases). *However, it seems that using different embeddings does improve the baseline.* When using the LSTM-CRF we obtained slightly different results. The LSTM-CRF generally outperforms the simple LSTM model, but here adding the CHAR+NER+POS features gives good results only when using the fine-tuned ELMo embeddings or BERT. By employing ConceptNet or the default embeddings we obtain a better F_1 score without using those features (even if the difference is minimal). By using LSTM-CRF and ELMo (fine-tuned) we were able to reach the highest F_1 score of 86.81 over the various embeddings (the baseline model reaches a maximum of 86.83). Ultimately, the best combination overall with a mean F_1 value of 85.94 is obtained by using the ConceptNet embeddings without additional features and the LSTM-CRF model. All these results surpass the previous baseline obtained with the WFST (83.73). The models were written in Python and PyTorch by forking from the previous project[9] and both the code and the dataset are available on Github[10].

5.1 Effects of Embeddings Fine-Tuning

Next we focused on analyzing the used embeddings by focalizing on ELMo which gave the most interesting results. We studyied the visual differences between the fine-tuned and averaged ELMo embeddings. We also evaluated the SSE (Sum of Squared Errors) of the generated clusters for certain concepts. By fine-tuning the ELMo weightings, it is possible to generate slightly better "concept clusters" in the embedding space. Figure 2 shows an example. We also saw an SSE error reduction of almost 50% with the fine-tuned version (from 13490936.61 to 7267435.40) when considering the *movie.name* concept. This is beneficial because a Neural Network learns how to divide the features space such to separate with hyperplanes each concept. Then we just need to check where the representation of my token lies to assign a concept. However, we noticed that, since our dataset is imbalanced (the concepts *O* and *movie.title* takes 80% of the total) only the clusters of the majority classes will be improved. This is corroborated by the fact that by analyzing the *confusion matrices* of the fine-tuned ELMo models (both LSTM and LSTM-CRF) it is possible to see how the accuracy of the majority concepts increases slightly while the accuracy of the other concepts degrades. See Table 2 for a comparison between the ELMo versions. We can conclude that fine-tuning pre-trained contextual embeddings is always advisable.

6 Conclusion

The experiments gave us new insights about the usage of embeddings for obtaining state-of-the-art performances on concept tagging tasks. BERT and ELMo pre-trained models provide new embeddings obtained by looking at the entire phrase. However, they cannot be used directly out-of-the-box but they need to be fine-tuned on the task at hand to be effective. This can be clearly seen by looking at the difference between the ELMo embeddings created by mere averaging of the layers and by learning instead the weights of the linear combination. We showed how ELMo fine-tuned embeddings provides certain advantages with respect to their not-tuned version. We think this

[9] https://github.com/fruttasecca/concept-tagging-with-neural-networks

[10] https://github.com/geektoni/concept-tagging-nn-spacy

	O	movie.name	director.name	actor.name	producer.name	movie.language	movie.release_date
Averaged ELMo	0.99584898	0.94475655	0.85628743	0.89855072	0.96330275	**0.92592593**	**0.94117647**
Fine-Tuned ELMo	**0.99625468**	**0.96190476**	**0.87647059**	**0.91724138**	**0.98181818**	0.84745763	0.90410959

Table 2: Given the top-seven concepts from the test dataset, we compute the tagging accuracy when using the LSTM-CRF and the CHAR+POS+NER features. The first five concepts appear more than 2% over the entire dataset, while the last two appear less than 1%. It is possible to see how the performances increase for the first five when using the fine-tuned ELMo, while the other two accuracies decrease with respect to the averaged ELMo.

is also the reason why BERT underperformed in some cases. We used an implementation which extracts directly the embeddings without going through the fine-tuning phase described in the original BERT paper. Therefore, the embeddings are not "trained" to interpret the current concept tagging task and they lead to bad performances. We are aware of the issues which arise from our validation methods (in particular, *t-SNE* (Wattenberg et al., 2016)) but we believe they still provide useful information. We presented also the impact of adding more features to the original model, more specifically, the POS tag and the NER entities. However, we did not see any important improvement. *We thought that having extra information like the NER tags could have improved the performances of the models but it was not the case*. This happened for several reasons. For instance, one major problem we found with this method is related to the sparsity of the NER tagging. On over 21000 tokens, only 400 were tagged with an entity definition (just the 1.9%). This adds just too little discriminative power to the original model and therefore it does not produce any huge gain in performances. In conclusion, we were able to surpass classical WFST methods and we were able to almost replicate the results of Gobbi et al. (2018). *However, even by employing more advanced embeddings and new feature combinations, we were unable to generate significative improvements during the evaluation*. We think that the general issue that needs to be solved to get better results is the unbalanced dataset we are using. Some concepts are underrepresented (e.g., *actor.type*, *actor.nationality*, etc.) while others are present in large quantity (e.g., *O* and *movie.name*). The model will be pushed to learn to classify the majority classes while ignoring the rest. We believe that this problem can be solved by increasing the size of the dataset. This can be done either by gathering real-world examples or by generating *synthetic sentences* by sampling from a statistical model trained on the original dataset. However, this task was left for future work.

References

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding.

Jacopo Gobbi, Evgeny Stepanov, and Giuseppe Riccardi. 2018. Concept tagging for natural language understanding: Two decadelong algorithm development.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Comput.*, 9(8):17351780.

Zhiheng Huang, Wei Xu, and Kai Yu. 2015. Bidirectional LSTM-CRF models for sequence tagging. *CoRR*, abs/1508.01991.

Pierre Lison and Jörg Tiedemann. 2016. Opensubtitles2016: Extracting large parallel corpora from movie and tv subtitles. In *LREC*.

Laurens van der Maaten and Geoffrey E. Hinton. 2008. Visualizing data using t-sne.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proc. of NAACL*.

Robyn Speer, Joshua Chin, and Catherine Havasi. 2017. ConceptNet 5.5: An open multilingual graph of general knowledge. pages 4444–4451.

Martin Wattenberg, Fernanda Vigas, and Ian Johnson. 2016. How to use t-sne effectively. *Distill*.

Kaisheng Yao, Baolin Peng, Geoffrey Zweig, Dong Yu, Xiaolong(Shiao-Long) Li, and Feng Gao. 2014. Recurrent conditional random field for language understanding. In *ICASSP 2014*. IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP).