MXX@FinSim3 - An LSTM–based approach with custom word embeddings for hypernym detection in financial texts

Nadine Kroher[1*] , **Aggelos Pikrakis**[1] , **Simon White**[1] , **Joe Lyske**[1]
[1]MXX
{nadine, aggelos, simon, joe}@mxx.ai,

Abstract

In this paper, we present our submission to the FinSim3 Shared Task, which introduces several improvements to a common approach to term classification. Specifically, in order to learn a robust and representative word embedding on domain specific text, we employ a series of established text preprocessing stages and increase the scope of training data to cover a plurality of financial text corpora. Then, instead of training a shallow classifier on word embeddings that are subsequently averaged over all words over the input text, we train a stacked recurrent neural network that takes the full sequence of word embeddings as input and associates it with a hypernym. The official task results show that these modifications yield a significant performance improvement compared to the baseline and outperform all other competing algorithms.

1 Introduction

The third edition of the *FinSim* task ([El Maarouf *et al.*, 2021], [Mansar *et al.*, 2021]) provides a framework for the comparative evaluation of hypernym detection methods targeting the financial domain. More specifically, participating systems are tasked with automatically associating a given financial term with the most relevant hypernym from a predefined set of concepts.

While text classification and hypernym detection are well-studied problems in the Natural Language Processing (NLP) community ([Nguyen *et al.*, 2017], [Aggarwal and Zhai, 2012]), this particular task poses a series of additional challenges which exclude a trivial application of off-the-shelf methods. Most importantly, the domain-specific language of the financial sector, with its unique terminology, results in poor generalisation of pre-trained word and sentence embeddings. While state of the art deep learning based embedding networks, such as Google's *Universal Sentence Encoder (USE)* [Cer *et al.*, 2018], *BERT* [Devlin *et al.*, 2018] or *SBERT* [Reimers and Gurevych, 2019], are generally powerful tools that are capable of encoding semantic similarity between words and sentences, they are trained on large generic text corpora which lack generalisation capabilities to financial language. In addition, the manually curated training data for the task is rather sparse compared to other NLP data sets which can often be obtained in a fully automatic manner by scraping appropriate online resources (i.e. [Krapivin *et al.*, 2009], [Dunn *et al.*, 2017] or [Wang *et al.*, 2020]).

Our approach to this task involves two stages. First, we train a custom *Word2Vec* [Mikolov *et al.*, 2013] embedding on a large amount of financial text. Specifically, we augment the training data by combining prospectuses supplied by the task organisers with text scraped from domain-specific web resources and terms from the training dataset. Furthermore, we apply a series of standard text preprocessing methods to the data to increase coverage and robustness of the resulting tokens. At a second stage, we train a recurrent neural network to associate a given term with its corresponding hypernym. The network takes as input the sequence of word embeddings extracted from the term to be classified and outputs a class probability for each of the target hypernyms. The official task results show that our method outperforms the baseline methods and all other submissions to the task.

The rest of this paper is organised as follows: Section 2 provides a brief overview of related approaches to the task. A detailed technical description of our method is given in Section 3 and the our task results are summarised in Section 4. Finally, are drawn in Section 5.

2 Related work

The first edition of the *FinSim* hypernym detection task [El Maarouf *et al.*, 2021] resulted in a variety of approaches to the problem, including a method that operated on sparse word representations [Berend *et al.*, 2020] and a system that used off-the-shelf *USE* embeddings as input [Anand *et al.*, 2021]. The system that won [Keswani *et al.*, 2020] had broadly followed the two-stage approach described in the previous section. In particular, for the first stage, the use of custom context-free *Word2Vec* embeddings and context-dependent *BERT* embeddings was investigated. For the second stage, they compared the performance of different shallow classifiers. In all setups, the input to the classifier was the averaged embedding over all words contained in the term. The second best performing system [Saini, 2021], augmented the training data with additional resources scraped from the

[*]Contact Author

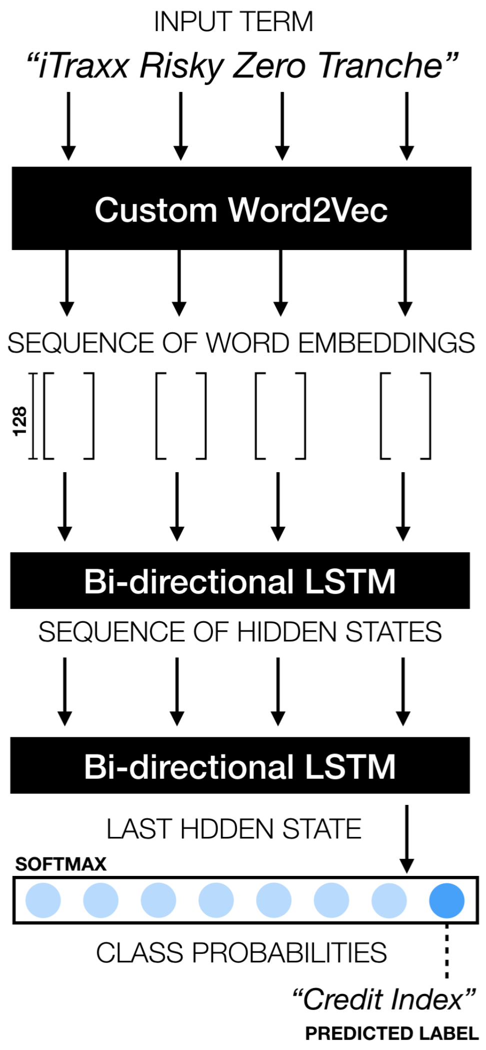

Figure 1: System overview

web and used hand-crafted features together with bi-gram TF-IDF features as input to a linear support vector machine.

3 Method

Our method follows the two-stage framework employed by the winning system of the previous edition of the task described in Section 2. At a first stage, we train a custom word embedding which maps words to vectors and is specifically trained to cover the unique terminology and use of language encountered in financial texts. The resulting word embeddings are then used to train a stack of *Long Short Term Memory (LSTM)* [Greff *et al.*, 2016] networks to map a sequence of word embeddings to the corresponding hypernym. An overview of the method is shown in Figure 1.

We selected this approach for two important reasons. Firstly, because it can operate on embedding sequences of arbitrary lengths rather than being restricted to a singled input vector, as it is the case with most shallow classifiers and simple feed-forward neural networks. In related approaches (see Section 2), the word embeddings extracted from individual words of a term are usually averaged to obtain a single embedding vector. Averaging is often, however, a simplification. For example, if only one out of five words is crucial to make a classification decision, its importance will be demoted by the averaging procedure. Furthermore, LSTMs are capable of learning temporal sequences and can thus leverage important information contained in the ordering of words within the term representation. In addition, LSTMs have relatively few trainable parameters compared to other fully-connected deep networks with many dense layers and can therefore be trained on smaller data sets with reduced over-fitting risk.

Below, the two stages of our method are described in more detail.

3.1 Custom word embedding

In order to generate a custom word embedding, we train the well-known Word2Vec algorithm, which can learn a mapping from a word to a vector in an unsupervised manner using unstructured text.

As training data, we use all sentences from the set of prospectuses and the flat *FIBO* ontology definitions supplied by the task organisers. In addition, we scraped all term definition from *Investopedia* [1], an online lexicon for financial terms and added those sentences to the training data. We furthermore added all terms from the training data of the classification task to the training set. The latter was done to ensure that frequently occurring words and expressions will be added as tokens in the embedding model, as it is for example the case with the word "iTraxx" which appears frequently in terms of the "Credit Index" class.

We furthermore applied a cascade of standard NLP preprocessing steps to the raw text to obtain more representative and robust word embeddings:

- we converted all characters to lower case
- we removed common stopwords
- we excluded terms which contain non-alphabetic characters
- we applied lemmatization to retain only the root form of each word

The resulting sentences were then used to train a Word2Vec model using the skip-gram algorithm and a 128-dimensional embedding vector.

3.2 Data augmentation

Even though the parameter count of our network architecture (see Section 3.3) is relatively low (approx. 150k parameters) compared to other deep, fully-connected architectures, the availability of training instances is also rather limited (approx. 1000 annotated terms were provided in the context of the task). We therefore augmented the training data using two mechanisms.

First, we identified a few web resources which contain additional terms that belong to a particular class

[1]https://www.investopedia.com

with a high degree of certainty and can thus be easily scraped. For example, the website www.advfn.com/nyse/newyorkstockexchange.asp contains a list of companies that are listed in the NYSE index and therefore belong with a high degree of confidence to the class ”stock corporation”. Similarly, the website www.bis.org/regauth.htm lists numerous terms belonging to the ”regulatory agency” class. In this way, we managed to gather additional instances for some of the target classes.

Secondly, we applied the text data augmentation method in [Marivate and Sefara, 2020] to expand the amount of training instances even further. More specifically, we created several variants of each term by replacing one of its words with the most similar word according to the custom Word2Vec model. For example, one augmentation of the term *”S&P Global Energy Sector Index”*, results in *”S&P Global Agriculture Sector Index”*. The example shows that the augmentation strategy yields additional instances that are mostly fictional but preserve characteristic word patterns and sentence structures. A more detailed description of this process can be found in [Marivate and Sefara, 2020].

After this procedure was completed, a total of 8000 training instances were available. We used 100 of the original terms supplied by the organisers as a validation set during the training procedure.

3.3 LSTM stack

The sequences of word embeddings are fed as input to a stack of two Bi-directional LSTMs, 64 units each. The first layer returns all intermediate hidden states (i.e., one per input embedding) and passes them to the second layer which returns the last hidden state only as a 64-dimensional output vector. This output is further connected to a dense softmax layer with one output node per class (17 output nodes in total).

The model is trained by minimizing the categorical cross-entropy loss using the *Adam* [Kingma and Ba, 2014] optimiser with an initial learning rate of 0.0001. We employ an early stopping criterion which stops the training process if the validation loss fails to improve over a patience period of 5 epochs.

At run time, we can interpret the output of the softmax layer as class-specific probabilities and rank our predictions accordingly, e.g. the predicted class is the class with the highest output value, followed by the one with the second highest, etc.

4 Results

The methods submitted to the competition are evaluated based on two metrics, i.e., *accuracy*, which describes the percentage of correctly predicted tags and *mean rank*, which stands for the average rank of the correct class in the list of predicted labels.

On the held-out validation set, our method achieves an accuracy of 0.96 and an average rank of 1.06. The official results of the actual challenge test set show that our method obtained an accuracy of 0.941 and a mean rank of 1.113 on the test set, thus outperforming the two baseline methods (baseline 1: accuracy= 0.564 and mean rank= 1.941; baseline 2: accuracy= 0.669 and mean rank= 1.75) as well as all other submitted approaches. It can be observed that the results of our method on the test set are only slightly inferior to the training stage results, which can be interpreted as an indication of our method’s robustness to over-fitting phenomena.

For the sake of completeness, the results of the baseline and all participating methods are summarised in Table 1.

team	**mean rank**	**accuracy**
Baseline 1	1.941	0.564
Baseline 2	1.75	0.669
dicoe 1	1.180	0.889
dicoe 2	1.162	0.904
lipi 1	1.257	0.886
lipi 2	1.22	0.895
lipi 3	1.156	0.917
MiniTrue 2	1.315	0.865
MiniTrue 1	1.346	0.855
MiniTrue 3	1.337	0.825
MXX	**1.113**	**0.941**
yseop 1	1.236	0.883
yseop 2	1.141	0.917

Table 1: Mean rank and accuracy across different algorithms from the official task results

5 Conclusions

We described a two-stage scheme for the classification of terms from the financial domain with respect to hypernym affinity. At a first stage, we extracted a custom Word2Vec embedding trained on domain-specific text that was pre-processed using a cascade of standard NLP methods. At a second stage, a stack of LSTMs was designed to take as input the sequences of word embeddings of a term and predict class probabilities for each of the possible hypernyms. We have also demonstrated that simple methods can be used to automatically gather additional data and augment the existing training corpus, thus permitting more complex networks to be trained with a reduced over-fitting risk. The official results of the task show that our approach outperforms both baseline methods as well as all other participating systems.

Acknowledgements

The authors would like to thank Naomi Clarke for providing insightful discussions on data gathering and ontology interpretation procedures.

References

[Aggarwal and Zhai, 2012] Charu C Aggarwal and ChengXiang Zhai. A survey of text classification algorithms. In *Mining text data*, pages 163–222. Springer, 2012.

[Anand *et al.*, 2021] Vivek Anand, Yash Agrawal, Aarti Pol, and Vasudeva Varma. Finsim20 at the finsim task: Making sense of text in financial domain. In *Proceedings of the Second Workshop on Financial Technology and Natural Language Processing*, pages 104–107, 2021.

[Berend *et al.*, 2020] Gábor Berend, Norbert Kis-Szabó, and Zsolt Szántó. Prosperamnet at the finsim task: Detecting hypernyms of financial concepts via measuring the information stored in sparse word representations. 2020.

[Cer *et al.*, 2018] Daniel Cer, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St John, Noah Constant, Mario Guajardo-Céspedes, Steve Yuan, Chris Tar, et al. Universal sentence encoder. *arXiv preprint arXiv:1803.11175*, 2018.

[Devlin *et al.*, 2018] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[Dunn *et al.*, 2017] Matthew Dunn, Levent Sagun, Mike Higgins, V Ugur Guney, Volkan Cirik, and Kyunghyun Cho. Searchqa: A new q&a dataset augmented with context from a search engine. *arXiv preprint arXiv:1704.05179*, 2017.

[El Maarouf *et al.*, 2021] Ismail El Maarouf, Youness Mansar, Virginie Mouilleron, and Dialekti Valsamou-Stanislawski. The finsim 2020 shared task: Learning semantic representations for the financial domain. In *Proceedings of the Second Workshop on Financial Technology and Natural Language Processing*, pages 81–86, 2021.

[Greff *et al.*, 2016] Klaus Greff, Rupesh K Srivastava, Jan Koutník, Bas R Steunebrink, and Jürgen Schmidhuber. Lstm: A search space odyssey. *IEEE transactions on neural networks and learning systems*, 28(10):2222–2232, 2016.

[Keswani *et al.*, 2020] Vishal Keswani, Sakshi Singh, and Ashutosh Modi. Iitk at the finsim task: Hypernym detection in financial domain via context-free and contextualized word embeddings. *arXiv preprint arXiv:2007.11201*, 2020.

[Kingma and Ba, 2014] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[Krapivin *et al.*, 2009] Mikalai Krapivin, Aliaksandr Autaeu, and Maurizio Marchese. Large dataset for keyphrases extraction. 2009.

[Mansar *et al.*, 2021] Youness Mansar, Juyeon Kang, and Ismail El Maarouf. The finsim-2 2021 shared task: Learning semantic similarities for the financial domain. In *Companion Proceedings of the Web Conference 2021*, pages 288–292, 2021.

[Marivate and Sefara, 2020] Vukosi Marivate and Tshephisho Sefara. Improving short text classification through global augmentation methods. In *International Cross-Domain Conference for Machine Learning and Knowledge Extraction*, pages 385–399. Springer, 2020.

[Mikolov *et al.*, 2013] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.

[Nguyen *et al.*, 2017] Kim Anh Nguyen, Maximilian Köper, Sabine Schulte im Walde, and Ngoc Thang Vu. Hierarchical embeddings for hypernymy detection and directionality. *arXiv preprint arXiv:1707.07273*, 2017.

[Reimers and Gurevych, 2019] Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. *arXiv preprint arXiv:1908.10084*, 2019.

[Saini, 2021] Anuj Saini. Anuj at the finsim task: Anuj@ finsimívlearning semantic representation of financial domain with investopedia. In *Proceedings of the Second Workshop on Financial Technology and Natural Language Processing*, pages 93–97, 2021.

[Wang *et al.*, 2020] Yanshan Wang, Naveed Afzal, Sunyang Fu, Liwei Wang, Feichen Shen, Majid Rastegar-Mojarad, and Hongfang Liu. Medsts: a resource for clinical semantic textual similarity. *Language Resources and Evaluation*, 54(1):57–72, 2020.