Applications of Word Embeddings and Neural Networks in Implicit Discourse Sense Labelling

Refining methodologies utilized in Schenk et al. (2016)

Atreya Shankar, Luis Glaser
{shankar,luglaser}@uni-potsdam.de
AM1: Shallow Discourse Parsing, WiSe 2018/2019
Prof. Dr. Manfred Stede, Dr. Tatjana Scheffler
University of Potsdam

19. April 2019

Abstract

Implicit discourse label parsing is considered to be a challenging and largely unsolved classification task in Natural Language Processing (NLP). This task has seen a recent surge of interest in 2015 and 2016 through the CoNLL Shared Tasks in discourse annotations, resulting in many researchers sharing their algorithms and developments. Schenk et al. (2016) proposed an interesting approach to implicit sense labelling by using a "simple" feedforward neural network architecture coupled with word embeddings. This paper focuses on improving the methodologies proposed in Schenk et al. (2016) by modifying the word embeddings and the downstream feed-forward neural network. With our modifications, in particular to the word embeddings, we were able to exceed the test F_1 baseline by 0.46%. Furthermore, we show that improvements in F_1 scores can arise by implementing negative sampling within word embeddings.

1 Introduction & Motivation

Discourse relations structure natural language within sentences or partial sentences. They express information that goes beyond what single token entities may represent, putting partial sentence units into a meaningful relationship with each other. E.g. the phrases "Alice left." and "Bob was mad." express only relatively atomic facts about the world. "Alice left, because Bob was mad" or "After Alice left, Bob was mad" however put those phrases into a relation and thus enable a reader to more fully understand the combined and even singular units and furthermore disambiguate the meaning of their singular parts. The possible relations between the two spans were made *explicit* by "because" and "after". However, these relations can also be made *implicitly* yet they will still be understood by any reader, depending on order or context of the connected spans.

As interesting as these connections between spans may be, they pose a challenge for automatic detection as they are implicit for our case. In order to make the challenge more feasible, annotations of the Penn Discourse Treebank (PDTB) (Prasad et al., 2008) reduce the complexity with different assumptions. E.g. connections can only exist between neighboring spans.

Shallow discourse parsing on the PDTB has drawn significant attention, most prominently being featured in the 2015-2016 CoNLL Shared Task (Xue, 2016) which was concerned with shallow discourse parsing on English and Chinese discourse treebanks. This paper will build on one submission to the CoNLL Shared Task by Schenk et al. (2016) which features a rather lightweight neural network architecture, drawing the necessary linguistic information from pretrained GoogleNews word embeddings. This makes it a good fit for a student project, due to the limited on-boarding cost.

In this paper we present our work and findings. All relevant code can be found on our forked GitHub[1] repository. We only worked on sense labelling in implicit discourse relations given gold arguments, to keep our work succint and focused.

2 Previous Work

Our main aim was to improve the disambiguation of implicit shallow discourse annotations in the PDTB. Implicit Disambiguation is one partial task of shallow discourse parsing, if following the pipeline design of Lin et al. (2014) and others.

Discourse Parsing in general and implicit discourse sense labeling have seen a surge in 2015 and 2016 with both year having CoNLL Shared Tasks in discourse annotations (Xue, 2016). However, implicit sense labeling cannot be regarded as a solved problem yet and still attracts interest and further innovation (Weiss and Bajec, 2018; Xu, 2017). In line with that development, we will as well focus on implicit sense labeling for discourse

[1] http://github.com/AtreyaSh/shallow-discourse-parser

annotations. For that, we based our work on the previous research of Schenk et al. (2016) and tried to further improve the metrics. The general idea of their and other work is to keep a parser pipeline as resource-lean as possible (Rönnqvist et al., 2017). This makes it less dependent on the availability of additional resources; a prototype can be put together for any language if high quality word embeddings are available. This is also due to not needing to heuristically craft features that might be dependent on language or language family. More sophisticated approaches in a linguistic sense can perform better and even provide linguistic insight (Oepen et al., 2016) which our approach lacks. Recent research has shown, that NLP-specific architectures like LSTMs can perform better than our simple feedforward network (Hochreiter and Schmidhuber, 1997). This however is leads to a stark increase of hyper parameters and design choices that need to be taken into account. Furthermore, Rutherford et al. (2017) argued that the limited amount of data in shallow discourse parsing does not meet the requirements for training of LSTMs. We thus argue, that extending the existent feedforward approach with our two part improvement is a feasible and relevant contribution within a student project.

3 Methodologies

Task 1: ⇒ Baseline embeddings ⇒ Baseline network	**Task 2:** ⇒ Modified embeddings ⇒ Baseline network
Task 3: ⇒ Baseline embeddings ⇒ Modified network	**Task 4:** ⇒ Modified embeddings ⇒ Modified network

Table 1. Overview of our approach

In our work we try to improve the architecture Schenk et al. (2016) proposed. For our improvements, we only focus on improving the effectiveness of implicit sense classification given gold arguments from the PDTB. We go about this process in two distinct ways. The first will be improving the word embeddings, the second will be improving the neural network itself. Since we needed to split our work for two students, we split the architecture into 4 tasks; highlighting the distinct embedding and neural network parts (See table 1). We tried all combinations of the architecture parts in order to soundly improve the it. This makes our work immune to mere implementation differences between the frameworks.

This section will describe our approach. First we will shortly summarize the data we and others have extensively studied (Section 3.1), then explain our intended improvements

sense	label	blind-test	dev	test	train
Comparison	17	0	1	0	145
Comparison.Concession	6	30	5	5	196
Comparison.Contrast	4	27	88	127	1657
Contingency	14	0	0	0	3
Contingency.Cause	5	0	0	0	1
Contingency.Cause.Reason	7	42	73	116	2098
Contingency.Cause.Result	11	33	52	89	1389
Contingency.Condition	8	0	0	0	2
EntRel	10	200	215	217	4133
Expansion	13	0	6	3	77
Expansion.Alternative	3	4	0	0	12
Expansion.Alternative.Chosen alternative	16	0	2	15	142
Expansion.Conjunction	18	106	122	147	3308
Expansion.Exception	15	0	0	0	1
Expansion.Instantiation	2	37	48	69	1134
Expansion.Restatement	9	141	103	190	2514
Temporal	19	0	0	0	1
Temporal.Asynchronous.Precedence	12	10	26	8	433
Temporal.Asynchronous.Succession	1	0	3	5	125
Temporal.Synchrony	0	3	19	5	212

Table 2. Distribution of Implicit/EntRel discourse relations

to the word embeddings (Section 3.2) and finally the contributions to the neural network (Section 3.4).

3.1 Data

The data we used our project stems from the Penn Discourse Tree Bank (PDTB) created from Wall Street Journal articles which already has been studied extensively for numerous tasks beyond discourse annotations (Prasad et al., 2008). More precisely we used the data that has been released for the CoNLL 2015/2016 Shared task. Table 2 shows the distribution of Implicit/EntRel discourse annotation over the dataset. We can already see, that some of the annotations are likely invalid as the class temporal should not be annotated but rather their subtypes precedence or succession. The forthcoming release of PDTB 3.0 should hopefully fix this issue. As the CoNLL Shared Task asked for a model for all given implicit discourse relations, we did not remove those relations despite its potential toll on model quality.

3.2 Word Embeddings

Word embeddings are used to represent textual data in a vector space in order to do computations with it like any other numeric data. More traditional approaches represent words

via *one-hot-encoding*, mapping each word to a distinct dimension. This causes the number of dimensions D being equal to the size of the vocabulary V. This makes computation with them rather expensive as most algorithms are at least $\in \mathcal{O}(D)$.

Word embeddings however can reduce D to an arbitrarily chosen number of dimensions (usually 50-300), which makes computation of and with them cheaper. Furthermore, distributional semantic information is encoded for each word within the embedding, giving metrics like cosine distance meaning[2], although their effect is limited (Iacobacci et al., 2016).

We compute these word embeddings using the open source package `gensim` (Řehůřek and Sojka, 2010). Intuitively, a word embedding with `Word2Vec` is computed by fitting a single layered neural network with dimensions D *on the corpus itself*. Based on a context, a given word is to be predicted when using the continuous bag of words architecture or vice-versa for continuous the skip-gram architecture (Mikolov et al., 2013). The resulting $|V| \times D, |V| < D$ word embeddings stem from the learned weight matrix within the neural network. The vocabulary is still represented by one-hot-encoded vectors. Multiplying the one-hot-encoded vector of a word v_w with the word-embedding will result in a single-column vector with D dimensions. Within this work, a word w will always be represented by this vector representation v_w with dimensions $D = 300$. Note that the varying number of dimensions in table 3 stems from further aggregating them as suggested by Schenk et al. (2016).

3.3 Word Embeddings: Tasks 1/2

For tasks 1 and 2, we firstly needed to define a baseline. To do this, we had to port the open-source code from Schenk et al. (2016) from Python2 to Python3, as the source-code was published several years ago. We then defined the baseline from Schenk et al. (2016) as model M_0. In order to introduce our own changes to the embeddings for task 2, we considered the following parameters included in the `gensim` python3 module:

1. **negative**: integer; refers to the number of negative samples used per positive-sampled word for training
2. **sg**: 1 if skip-gram architecture is used, 0 if common-bag-of-words (CBOW) architecture is used
3. **hs**: 1 if hierarchical softmax is used, 0 if negative sampling is used
4. **final dimension**: final dimensionlity after concatenating aggregated word vectors

[2]Inspired by Firth's "You shall know a word by the company it keeps." (1957, quoted from Sadeghi et al., 2015)

5. **aggregation**: 0 for baseline approach using averaging and products, 1 for replacing baseline products with variance, 2 for removing baseline products

6. **context**: 1 if contexts of implicit arguments are used in aggregation, 0 if contexts are ignored

7. **stop**: 1 if stopwords are excluded, 0 if stopwords are untouched

Model	negative	sg	hs	final dimension	aggregation	context	stop
M_0	0	1	1	600	0	0	0
M_1	10	1	0	600	0	0	0
M_2	0	1	1	600	1	0	0
M_3	0	1	1	600	0	0	1
M_4	0	1	1	1200	1	0	0
M_5	0	1	1	600	0	1	0
M_6	0	1	1	600	2	1	0
M_7	0	1	1	1200	1	1	0
M_8	10	0	0	1200	1	1	0
M_9	5	0	0	1200	1	1	0
M_{10}	0	1	1	600	1	1	0
M_{11}	20	1	0	600	0	0	0

Table 3. Overview of various word-embedding models

Next, we defined 11 additional models, specifically M_1 to M_{11} by varying the above-mentioned parameters pertaining to word-embeddings. A complete overview of the properties of these models is described above in Table 3. After defining these models, we trained each individual word-embedding corresponding to these models and tested them with the baseline neural network from Schenk et al. (2016). We did this by performing a grid-based search for optimal hyperparameters inherent to the neural network. These parameters include decay rate, momentum and learning rate. As a result, each word-embedding was tested on 72 variants of the baseline neural network and the corresponding training, development and test accuracies were recorded.

3.4 Neural Network: Tasks 3/4

After having represented our corpus within a vector space with D dimensions, we use those embeddings to classify our data. The classes we are trying to fit stem from the discourse relations that the implicit discourse relations express. The PDTB annotation allows for up to two relations between two arguments, however, we do only predict one single relation for each argument pair.

Since we want to add to the work of Schenk et al. (2016), we will first port their work to a newer framework than `theanets`. In hope of increasing the longevity of our work, we decided to port it to `keras 2.0` (Chollet et al., 2015) with a `theano` backend (Theano Development Team, 2016). Furthermore we intend to increase the quality of the architecture and potentially increase the metrics as well. This goal will be reached by first porting the `theanets` architecture to `keras` in order to immunize against simple implementation differences between both packages. We then further add to the architecture of Schenk et al. (2016), by trying out different aggregation methods and by further adapting the neural network. Due to time limitations we mostly focused on improving the new network instead of working with the old architecture. We consider this a feasible cutback as the result we aim for is a usable extension to the existing parser without legacy code limiting it's usability.

Layer type	input dimensions	output dimensions
Input	600	600
Dense	600	1000
PReLU Activation	-	-
Dense	1000	1250
PreLu Activation	-	-
Dense	1000	20

Table 4. Network Architecture

Most prominently we will introduce a second hidden layer h_2 as depicted in table 4. We denote the parametric rectifier linear unit (PReLU) as a layer as well since it takes a parameter a into account which is updated during back-propagation as well and has already seen adaptation in previous work (Schenk et al., 2016).

Model	worst dev-F_1	best dev-F_1	worst test-F_1	best test-F_1
M_0	0.3933065	0.4319465	0.3286365	0.3593190
M_1	0.3819941	**0.4395718**	0.3278879	**0.3639238**
M_2	0.3923425	0.4256447	0.3151798	0.3543410
M_3	0.3631321	0.4128714	0.3051548	0.3309730
M_4	0.3947183	0.4291707	0.3254213	0.3501314
M_5	0.3861839	0.4379196	0.3145549	0.3570653
M_6	0.3137239	0.3429012	0.2639836	0.2886952
M_7	0.3230136	0.3800511	0.2724814	0.3134248
M_8	0.3169306	0.3998472	0.2842099	0.3419556
M_9	0.3140485	0.3737086	0.2698399	0.3317736
M_{10}	0.3946776	0.4321273	0.3099193	0.3514591
M_{11}	0.3791169	**0.4396398**	0.3327401	0.3590001

Table 5. Best and worst dev/test F_1 scores

4 Results

4.1 Tasks 1/2

After running the word-embeddings for the models M_0 to M_{11}, we were able to observe and record the performance of these models with respect to the baseline neural network from Schenk et al. (2016). Table 5 shows the best and worst dev/test F_1 scores. Model M_1 showed an improvement in the test F_1-score over the baseline by $\approx$0.46%. This shows the importance and utility of negative sampling in word embeddings, since model M_1 had negative sampling included while model M_0 did not.

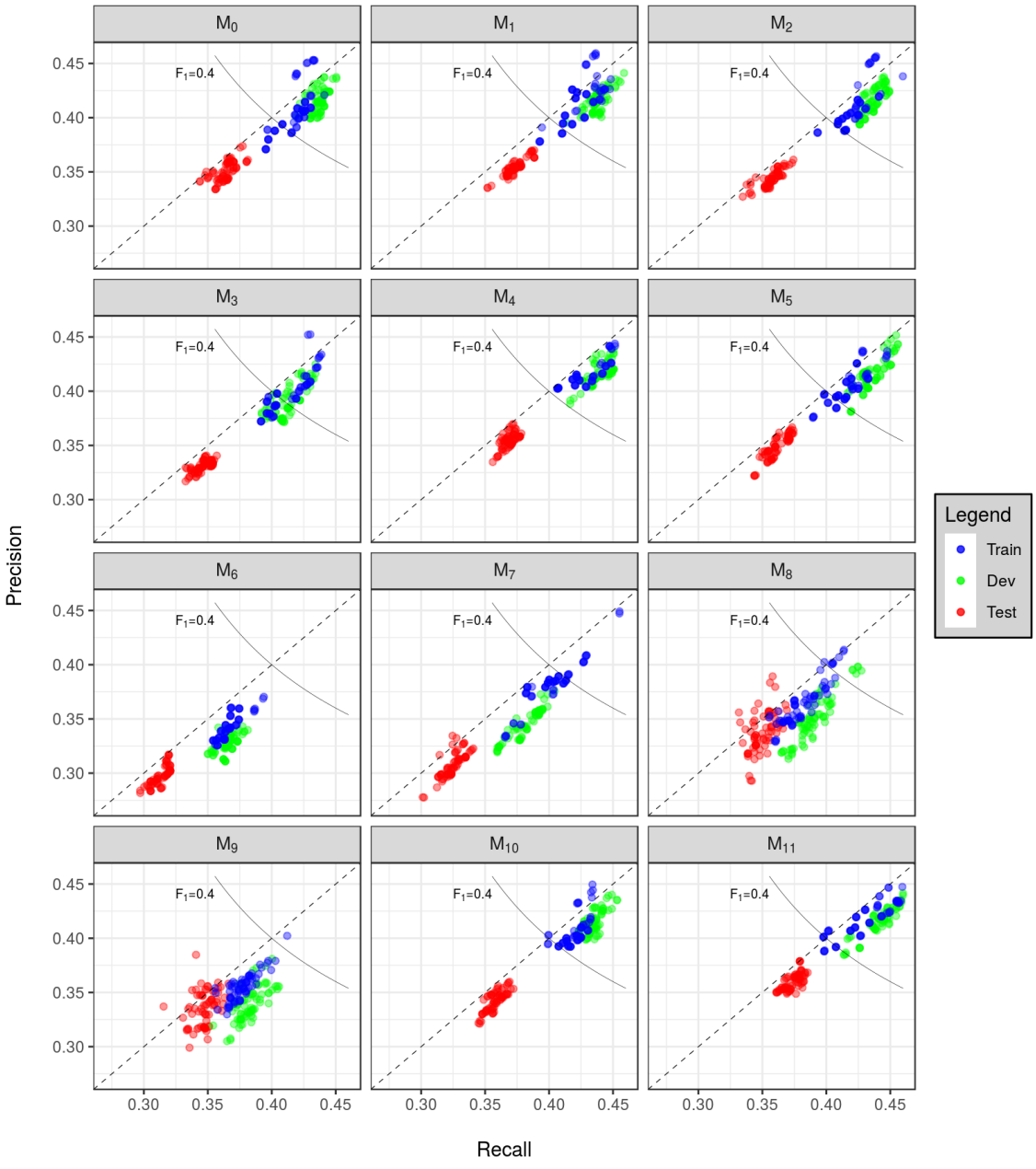

Figure 1. Precision-Recall plots of applied word-embedding models

For the purpose of visualization, we compiled all of the weighted precisions and recalls from each of the 72 runs per word-embedding model. We then plotted the precision-recall curves of all the corresponding models in Figure 1. We also plotted an approximate contour line for the F_1 score of 0.4 relative to precision and recall.

We can observe interesting trends above. For example, we can observe the phenomenon of overfitting in cases where the train and development clusters lie significantly further away from the test clusters. In this regard, we can observe that M_8 and M_9 are models with a smaller tendency to overfit, since all three train, development and test clusters are close to each other. Other models appear to have a greater overfitting tendency.

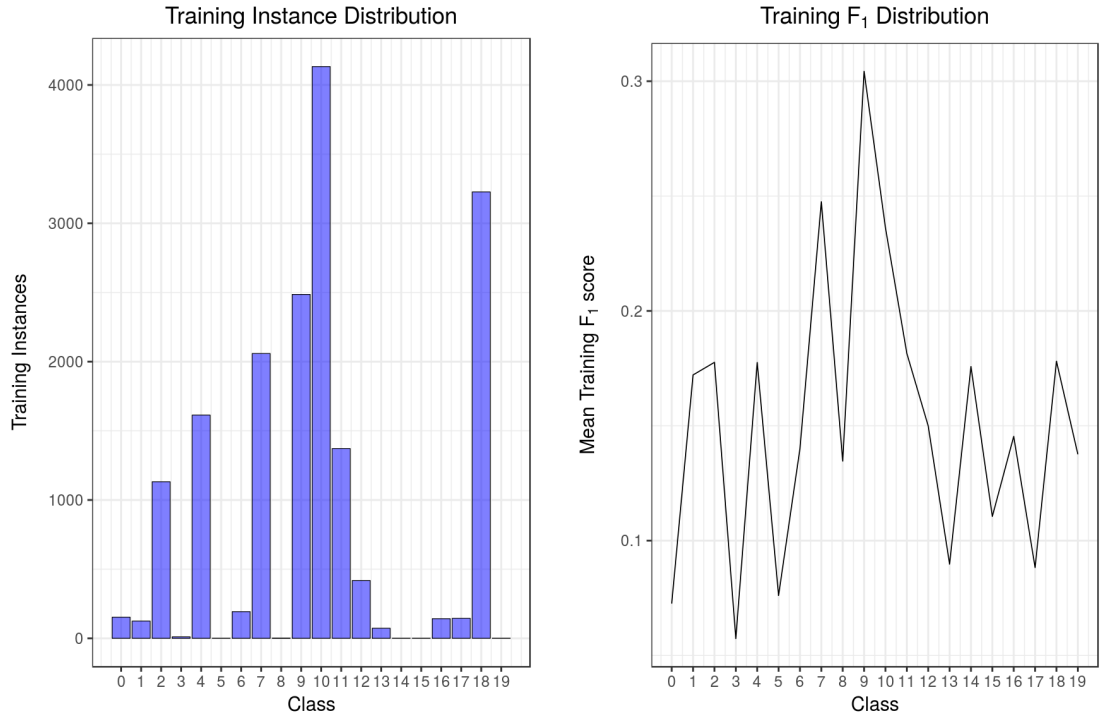

Figure 2. Distribution of implicit sense training data and training F_1-scores

As a final step, we also wanted to investigate the relationship between training sample size and training F_1 scores. We did this by macro-averaging the class-wise training F_1 scores and visualized the results in Figure 2. One can observe that if the training sample size is large, the training F_1 score is generally also large. This also stresses the importance of posessing training data, which is a limitation for this implicit sense labelling task due to the data-related constraints of the PDTB.

4.2 Tasks 3/4

We were able to fully reconstruct the `theanets` architecture in `keras`. Thus, we can safely compare our results between the new and old embeddings and new and old net-

work without confusing our results with differences in implementation. Stopword filtering decreased F1 by .02, suggesting that they do carry relevant information for the task of implicit discourse classification.

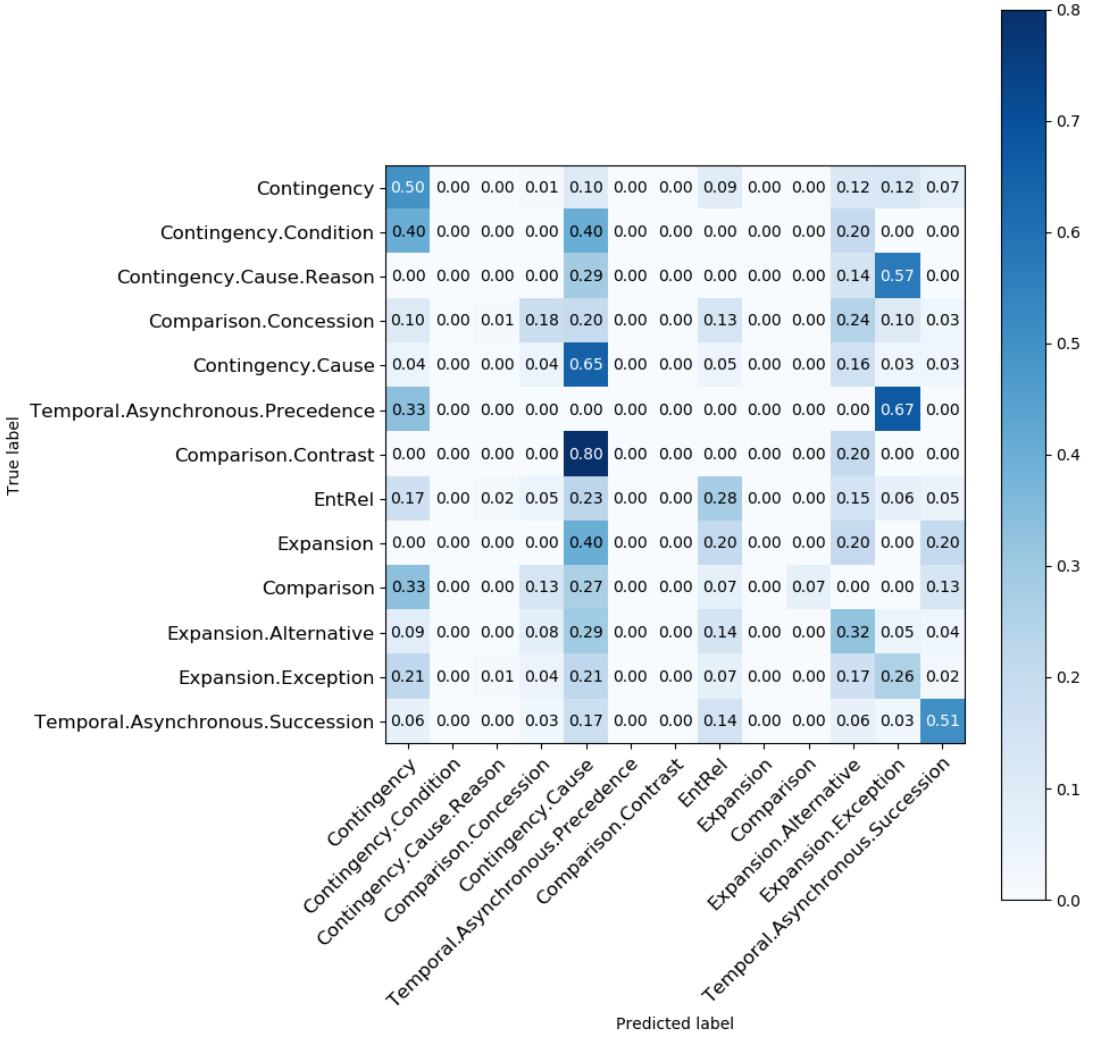

Figure 3. Confusion matrix for test set

Figure 3 Shows the confusion matrix for our top `keras` model with new embeddings for the test set. The x axis represents prediction made by the model. The y axis represents true annotations in the training data. Hits on the principal diagonal represents a true positive. Cells differing from the principal diagonal visualize missed predictions and can give an intuition of potential biases within a model.

On first glance we see, that our model does not solve the problem fully, as it is expected for implicit discourse annotations. Only `contingency.cause` and `temporal.asynchronous.succession` see $> .5$ correct predictions. However the column of `contingency.cause` already shows the model has a bias towards this class, as it predicted this class for all relations at some point.

As this class has only occurred once in the training set, never in development and train and five times in blind test (cf. table 2) an improvement in F1 could have been achieved by removing rare relations. However this was forbidden by the CoNLL Shared Tasks, we thus kept those relations.

We see a similar phenomenon in `expansion.exception` which only occured once in training but had a fairly strong impact on prediction, most prominently being mistaken for `temporal.asynchronous.precedence`.

5 Conclusion

In this work we aimed to further improve on the implicit sense labeling that Schenk et al. (2016) began. We improved their classifier in two ways: We first continued developing on the word embeddings by improving the training routine and changing the aggregation functions that feed into the neural network. By implementing negative sampling in word embeddings, we arrived at a $\approx$0.46% increase in test F_1 score.

Second we improved the architecture of the network by porting it to python3 and keras 2.0, adding another hidden layer and increasing the number of hidden nodes. However the theanets legacy code did still outperform the updated code. This is most likely due to small implementation differences between both packages. However the improved word embeddings are promising and should be independent of further changes.

In order to build on this approach, surely improving the network architecture further is feasible. Another promising way would be to try and fit the most recent advances in NLP classifiers like open AIs transformer to this task. Nevertheless the updated architecture allows to easily adapt new network designs to improve the performance.

References

François Chollet et al. Keras. 2015.

Sepp Hochreiter and Jürgen Schmidhuber. Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780, November 1997. ISSN 0899-7667, 1530-888X. doi: 10.1162/neco.1997.9.8.1735.

Ignacio Iacobacci, Mohammad Taher Pilehvar, and Roberto Navigli. Embeddings for Word Sense Disambiguation: An Evaluation Study. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 897–907, Berlin, Germany, 2016. Association for Computational Linguistics. doi: 10.18653/v1/P16-1085.

Ziheng Lin, Hwee Tou Ng, and Min-Yen Kan. A PDTB-Styled End-to-End Discourse Parser. *Natural Language Engineering*, 20(2):151–184, 2014.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient Estimation of Word Representations in Vector Space. *arXiv:1301.3781 [cs]*, January 2013.

Stephan Oepen, Jonathon Read, Tatjana Scheffler, Uladzimir Sidarenka, Manfred Stede, Erik Velldal, and Lilja Øvrelid. OPT: Oslo–Potsdam–Teesside. Pipelining Rules, Rankers, and Classifier Ensembles for Shallow Discourse Parsing. In *Proceedings of the CoNLL-16 Shared Task*, pages 20–26, Ann Arbor, Michigan, 2016. Association for Computational Linguistics. doi: 10.18653/v1/K16-2002.

Rashmi Prasad, Nikhil Dinesh, Alan Lee, Eleni Miltsakaki, Livio Robaldo, Aravind Joshi, and Bonnie Webber. The Penn Discourse TreeBank 2.0. In *Proceedings of the Sixth INternational Conference on Language Resources and Evaluation (LREC)*, Marrakesch, 2008.

Radim Řehůřek and Petr Sojka. Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 45–50, Valletta, Malta, 2010. ELRA. http://is.muni.cz/publication/884893/en.

Samuel Rönnqvist, Niko Schenk, and Christian Chiarcos. A Recurrent Neural Model with Attention for the Recognition of Chinese Implicit Discourse Relations. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 256–262, Vancouver, Canada, 2017. Association for Computational Linguistics. doi: 10.18653/v1/P17-2040.

Attapol Rutherford, Vera Demberg, and Nianwen Xue. A Systematic Study of Neural Discourse Models for Implicit Discourse Relation. In *Proceedings of the 15th Conference*

of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers, pages 281–291, Valencia, Spain, 2017. Association for Computational Linguistics. doi: 10.18653/v1/E17-1027.

Zahra Sadeghi, James L. McClelland, and Paul Hoffman. You shall know an object by the company it keeps: An investigation of semantic representations derived from object co-occurrence in visual scenes. *Neuropsychologia*, 76:52–61, September 2015. ISSN 00283932. doi: 10.1016/j.neuropsychologia.2014.08.031.

Niko Schenk, Christian Chiarcos, Kathrin Donandt, Samuel Rönnqvist, Evgeny Stepanov, and Giuseppe Riccardi. Do We Really Need All Those Rich Linguistic Features? A Neural Network-Based Approach to Implicit Sense Labeling. In *Proceedings of the CoNLL-16 Shared Task*, pages 41–49, Ann Arbor, Michigan, 2016. Association for Computational Linguistics. doi: 10.18653/v1/K16-2005.

Theano Development Team. Theano: A Python framework for fast computation of mathematical expressions. *arXiv e-prints*, abs/1605.02688, 2016.

Gregor Weiss and Marko Bajec. Sense classification of shallow discourse relations with focused RNNs. *PLOS ONE*, 13(10):e0206057, October 2018. ISSN 1932-6203. doi: 10.1371/journal.pone.0206057.

Jingjing Xu. Shallow Discourse Parsing with Maximum Entropy Model. *arXiv:1710.11334 [cs]*, October 2017.

Nianwen Xue, editor. *Proceedings of the CoNLL-16 Shared Task*. Association for Computational Linguistics (ACL), Stroudsburg, Pa, 2016. ISBN 978-1-932432-66-4. OCLC: 836423929.