Event Nugget Detection, Classification and Coreference Resolution using Deep Neural Networks and Gradient Boosted Decision Trees

Nils Reimers† and Iryna Gurevych†‡
†Ubiquitous Knowledge Processing Lab (UKP-TUDA)
Department of Computer Science, Technische Universität Darmstadt
‡Ubiquitous Knowledge Processing Lab (UKP-DIPF)
German Institute for Educational Research
`http://www.ukp.tu-darmstadt.de`

Abstract

For the shared task of event nugget detection at TAC 2015 we trained a deep feed forward network achieving an official F_1-score of 65.31% for plain annotations, 55.56% for event mention type and 49.16% for the realis value.

For the task of Event Coreference Resolution we prototyped a simple baseline using Gradient Boosted Decision Trees achieving an overall average CoNLL score of 70.02%.

Our code is publicly available on GitHub[1].

1 Introduction

The Event Nugget Detection task at NIST TAC KBP 2015 aimed to identify the mention of events in text. Every instance of a mention for the defined event types must be identified. An Event Nugget is defined as the smallest extent of text (usually a single word or a few words) that expresses the occurrence of an event.

For this shared task, 360 documents were annotated using the Rich ERE Annotation Guideline v2.5.1 (Zhiyi et al., 2015). 158 of those documents were provided for training and the systems where evaluated on the remaining 202 documents. Both, newswire articles as well as posts from discussion forums were annotated. The Rich ERE Annotation Guideline specifies eight main event types (Business, Contact, Conflict, Justice, Life, Manufacture, Movement, Personnel, Transaction) with each up to 13 subtypes totaling to 38 distinct type and subtype combinations. Table 1 depicts the distribution of the event types in the provided training and test sets. In addition to the mention type, systems were supposed to identify three realis values (ACTUAL, GENERIC, OTHER).

Some examples for the Event Nugget Detection task are:

- President Obama will `nominate` [realis:Other type:Personnel.Nominate] John Kerry for Secretary of State.
- He carried out the `assassination` [realis:Actual type:Life.Die].

The same event nugget may be tagged more than once for different event types/subtypes, for example when the nugget instantiates different events. As this was the case for less than 5% of the events in the training dataset, we decided to ignore possible multi-tags and simplified this task to a single-tag classification problem.

Our proposed tagger works in three phases: The first phase determines whether a token is part of an event nugget or not. The second and third phases determine consecutively the event type and realis value for all identified event nuggets. For all three phases we used the same neural network based on the design of Collobert et al. (2011) with the same preprocessing. The best run achieved in the official metric an F_1-score of 65.31% for plain annotations, i.e. detecting event nuggets without determining their mention type. For the mention type, we achieve an F_1-score of 55.56% and an F_1-score of 49.16% for

[1] `https://github.com/UKPLab/tac2015-event-detection`

the realis value. We also experimented with directly tagging the event type, however, this resulted in an F_1-score for plain annotations of 60.16%, a drop of around five percent points in comparison to the three phase approach.

For the task of event coreference resolution, we implemented a simple prototype using Gradient Boosted Decision Trees (Breiman, 1997; Friedman, 2000; Friedman, 2002) with some basic features. This implementation achieves for task 3 an overall average CoNLL score of 70.02% on the evaluation data.

This work is organized as follows. Section 2 describes our experience with re-annotating the corpus. Sections 3 and 4 describe the used preprocessing and extracted features for event nugget detection and classification, and section 5 describes the used neural network for this task. Section 6 presents our results. Section 7 presents our baseline and the results for the event coreference resolution.

2 Manual Re-Annotation

No inter-annotator-agreement was reported for the dataset, therefore we decided to perform a manual re-annotation on a subset. This re-annotation also allows us to estimate an upper-bound for the automatic approach. The annotator was trained on some documents of the training data and then annotated randomly sampled sentences. He annotated around 4500 tokens containing 190 event nuggets and, when compared to the provided gold labels, achieved an F_1-score of 76.57%. We observed a low agreement on words like *told* or *said*. In some parts of the dataset, these are annotated as *Contact* events, in other parts they are not annotated as event nuggets. We were not able to discover a consistent annotation for these words, and it appears that the annotation mainly depends on the original annotator who annotated the document. This type account for 27% of the missed event nuggets, while only 14% of the event nuggets were *Contact* events.

Another anomaly detected in the dataset was the tagging of pronouns like *this* or *it*. In some documents, they are tagged with the same tag as the event they refer to. In other documents, they are not tagged, even when they were referring to an event.

Type	**Subtype**	**#Train**	**#Test**
Business	Declare Bankruptcy	33	44
Business	End Org	13	6
Business	Merge Org	28	33
Business	Start Org	18	35
Conflict	Attack	800	591
Conflict	Demonstrate	200	149
Contact	Broadcast	417	510
Contact	Contact	337	587
Contact	Correspondence	95	110
Contact	Meet	244	272
Justice	Acquit	30	31
Justice	Arrest-Jail	37	69
Justice	Appeal	287	348
Justice	Charge-Indict	190	155
Justice	Convict	222	96
Justice	Execute	66	97
Justice	Extradite	63	60
Justice	Fine	55	45
Justice	Pardon	239	51
Justice	Release-Parole	73	124
Justice	Sentence	144	158
Justice	Sue	55	72
Justice	Trial-Hearing	196	155
Life	Be Born	19	17
Life	Die	514	408
Life	Divorce	45	49
Life	Injure	133	87
Life	Marry	76	83
Manufacture	Artifact	22	90
Movement	Transport.Artifact	70	66
Movement	Transport.Person	517	439
Personnel	Elect	97	71
Personnel	End Position	209	291
Personnel	Nominate	35	63
Personnel	Start Position	77	94
Transaction	Transaction	51	63
Transaction	Transfer-Money	551	554
Transaction	Transfer-Ownership	280	265
		6538	6438

Table 1: Distribution of event types in the provided training and test datasets.

3 Preprocessing

Given the existent segmentation of the documents, we performed only minimal preprocessing. We decided to remove all HTML tags. The href-attribute of some HTML <a> tags contained event nugget annotations. We decided to ignore all HTML tags, even though this might decrease the system performance for this shared task. This is because we think that in typical use cases, the user is not interested to tag tokens inside HTML tags. After removing HTML tags, the document was split into sentences at every single period, exclamation mark, or question mark.

To simplify processing, we decided to convert the documents and annotations to a format as it has been used for many different CoNLL shared tasks on tagging problems, including POS-tagging and Named Entity Recognition. A sentence is encoded as one token per line, with information provided in tab-separated columns. Spans are encoded in the BIO-scheme (Sang and Buchholz, 2000).

Some event nuggets might trigger multiple events, for example the event nugget *bought* in the sentence *"I bought a car for $5.000"* triggers the events *Transaction_Transfer-Money* and *Transaction_Transfer-Ownership*. Such multiple events were annotated in the dataset. This transforms the task to a multi-label tagging task, which is a lot more complicated to model than a single-label tagging task. As only around 5% of the event nuggets triggered more than one event, we decided to ignore the multi-label setup and only extracted a single label per token. An alternative approach would be to transform the multi-label task to a single label task by introducing new labels for tokens that trigger more than one event.

4 Feature Extraction

The main strength of our classifiers originates from pre-trained word embeddings using methods like word2vec (Mikolov et al., 2013) or GloVe (Pennington et al., 2014). Given only the target token and the neighboring tokens, our classifier achieves an F_1 score of 61.73% on the test set.

To test the impact of different features on the classification results, we used DKPro-Core (Eckart de Castilho and Gurevych, 2014) to extract some features. More specifically, we extracted the lemma of each word, the POS-tag and, if present, the subject and object linked to the token based on the parse tree. For these steps, we used Stanford CoreNLP[2]. We also extracted the initial and final 2 to 6 characters of the token, the word-shape[3] given by `wordShapeChris2` from Stanford CoreNLP, and a feature to indicate that tokens are inside quotes. The documents came either from newswire or from discussion forums. We also experimented with including the document source type into the classifier.

[2] `http://nlp.stanford.edu/software/corenlp.shtml`

5 Neural Network Design

Collobert et al. (2011) propose a unified neural network architecture that can be applied to various natural language processing tasks. The presented deep neural network architecture uses only features based on minimal preprocessing and several layers of abstractions are learned.

The first layer is a *lookup operation* which maps each word and its associated features (e.g. POS) to a d-dimensional vector. The second layer makes the assumption that the entity of a word can be predicted from its neighboring words. The vectors from the lookup operation for the target word and the neighboring words are concatenated and fed through an affine transformation followed by a non-linear activation function like the hyperbolic tangent function.

There are two different approaches for the last layer of the network, depending on whether the *isolated tag criterion* or the *sentence tag criterion* is used. For the *isolated tag criterion*, each word in the sentence is considered independently. The probabilities of the different tags for each word are computed by the softmax-function.

The *sentence tag criterion* optimizes the label sequence over the entire sentence. Tag probabilities from each window are concatenated and the dependencies between tags are factored into the model by learning initial probabilities and transition probabilities between tags. The Viterbi algorithm is used during inference.

We compared the performance of both variants for the task of event nugget detection and did not find an improvement from the more complex *sentence tag criterion*-model. Event nuggets are in most cases isolated within a sentence with non-event-tokens before and after the event nugget. Also, event nuggets

[3] The word-shape feature transforms a word to a string representing the shape of the word. Upper-case characters are for example represented by X, low-case characters by x and digits by #. The token CoNLL'03 would be transformed to XxXXX'##.

rarely span over multiple tokens. Optimizing the label sequence does therefore not make a difference in our task, and we decided to use the simpler and more efficient isolated tag criterion for further experiments.

We experimented with different context window sizes, hidden layer sizes and feature combinations. Our final submission, achieving an F_1-score of 65.31% on the plain annotations, used the token, the lemma, the POS, the capitalization, the initial / final 2 to 6 characters, and the subject / object as features.

We used a context window size of 3. For the lookup of the lemmas, we used the pre-trained word embeddings by Levy and Goldberg (2014) on the English Wikipedia using dependency links as context. As Levy and Goldberg demonstrated, word embeddings trained on dependencies favor syntactic functions over semantic similarity. For the task of event nugget recognition, we were mainly interested in tokens with similar syntactic functions. As Table 2 shows, those embeddings also resulted in the highest F_1 score.

For the tokens, the idea was to use different embeddings than for the lemmas. We used there the pre-trained word embeddings by Mikolov et al. (2013) on the Google News dataset. The embeddings by Levy et al. capture syntactic similarity well, while the vetors trained on the Google News dataset capture semantic similarity well. The combination of both embeddings allows the neural network to choose the more suitable embeddings for the task. For the subject and object we used the pre-trained embeddings by Levy and Goldberg. For the other features, i.e. capitalization, POS, initial and final characters of the token, we randomly initialized the embeddings and updated them during training. We chose a dimension of 5 for bi- and trigrams, as there is only a limited number of bi- and trigrams, and a dimension of 10 for four- to sixgrams.

It is known that updating the word embeddings helps to fit those for the specific task. However, updating bears the risk that the updated embeddings overfit on the training data and result in a worse test performance. We decided therefore to update those only for the first four epochs of the training and stopped updating after that to prevent overfitting.

For the hidden layer, we achieved the best result with 100 hidden neurons and the tangent hyperbolic activation function. For training, we used stochastic gradient descent with an initial learning rate of 0.1. The cross entropy error with no weight regularization was used. We used a mini-batch size of 35.

We used the same features and setup to detect first the plain event nuggets in the text, and we then labeled the event type and realis value for each detected event nugget. For the plain event nugget detection, the system was trained for 8 epochs, 243 epochs for the event type classification and 8 epochs for the realis classification. All hyperparameters were derived by training the network on 75% of the provided documents and computing the performance on the remaining 25% documents.

6 Evaluation

Pre-trained word embeddings have a major impact on the performance of the classifier (Collobert et al., 2011). We therefore examined as the first step the performance for different existent word embeddings. Table 2 depicts the F_1-score for a context window size of 3 and a hidden layer of 100 for different, publicly available word embeddings. During the shared task, we computed the scores on our internal train-test-split. Throughout this section, we will report the F_1 score on the provided test dataset.

As depicted in Table 2, we get the worst F_1-score with the pre-trained embeddings from the GloVe website[4]. The best results are achieved with the word2vec embeddings from Levy's website[5]. Detecting an event nugget is in the first step a syntactic challenge and classifying it to one of the 38 types is in the second step a semantic challenge. The embeddings by Levy and Goldberg that were trained on dependency links are especially suitable to capture syntactic relatedness of words which we think gave this small increase in performance.

Next we tested the impact of the window size. As shown in Table 3, there is only a slight difference in terms of performance for different context window sizes. With a window size of 1, i.e. only using the embedding and capitalization information for the target word, we achieve an F_1-score of 60.04%.

[4]http://nlp.stanford.edu/projects/glove/

[5]https://levyomer.wordpress.com/2014/04/25/dependency-based-word-embeddings/

Embeddings	Token	Lemma
word2vec, Google News Corpus	58.93%	59.32%
word2vec, Wiki., dep. links	60.70%	59.76%
word2vec, Wiki., word context	60.16%	59.23%
GloVe - 6B - 100d	58.38%	58.62%
GloVe - 6B - 300d	57.76%	58.23%
GloVe - 840B - 300d	58.12%	59.43%

Table 2: F_1 score with different pre-trained word embeddings. 1) word2vec on 100 billion token Google News Corpus (Mikolov et al., 2013), 2) word2vec with dependency links on Wikipedia (Levy and Goldberg, 2014), 3) word2vec with context window size of 5 on Wikipedia (Levy and Goldberg, 2014), 4) GloVe (Pennington et al., 2014) on Wikipedia 2014 and Gigaword 5 (6 billion tokens) with 100 dimension, 5) GloVe on Wikipedia and Gigaword with 300 dimensions, 6) GloVe on Common Crawl with 300 dimensions (840 billion tokens).

Hence, local context information appears to be of minor importance for the task of event nugget detection.

Window Size	Precision	Recall	F_1
1	82.02%	47.35%	60.04%
3	81.92%	48.21%	60.70%
5	81.94%	48.68%	61.08%
7	81.32%	49.75%	61.73%

Table 3: Performance using different window sizes. Window size 1 means that only the target token is included. The word embeddings by Levy and Goldberg (2014) and a 100 dimensional hidden layer is used.

As described in section 4, we extracted different additional features. Table 4 compares the performance for different feature combinations. As the table shows, adding further local features like POS or initial/final characters, does not improve the performance. Adding more long distance and global feature, for example the subject and object linked to the word (if existent) or the document source type (newswire vs. discussion forum), can have a slight impact on the performance. Instead of modeling this information as additional features, a network that automatically captures information on sentence level, for example by using a convolutional layer that convolves over all tokens in the sentence, might be a more suitable approach.

In all our experiments, we noticed a comparatively high precision of around 80% while only a comparatively small recall of around 50%. Further, we noticed a large performance difference between our internal development set[6] and the test set. While our system achieves an F_1 score of 73.00% on the development set, it only achieves 65.31% on the provided test set. In fact, when training on the complete training set, the performance drops on the test set to 62.26%. It appears that this effect is due to some inconsistent annotations in the datasets. As noted in section 2, the annotation for *Contact* events is fairly noisy and in the test set, *Contact* events appear nearly twice as much as in the train set which could be due to an inconsistent annotation of those.

Features	F_1
Token	61.73%
Token + POS	60.84%
Token + initial / final characters	61.67%
Token + Lemma	61.15%
Token + subject & object	62.10%
Token + document source type	62.24%
Token + quote detector	61.84%

Table 4: Performance using different feature combinations with window size 7, 100 dimensional hidden layer and the word embeddings presented by Levy and Goldberg (2014).

After detecting the event nuggets, we used the same network to determine the event type and the realis value. Those networks were trained only on event mentions, ignoring all other non-event tokens. For determining the event type, we achieve an accuracy of 71.38% for all event nuggets on the test set and an accuracy of 73.00% for the realis value. When combining it with the automatic event nugget detection, we achieve in the official evaluation an F_1 score of 55.56% for event mention type and 49.16% for the realis value.

6.1 Error Analysis

Table 6 depicts the precision, recall and F_1-score per event type. A predicted event nugget was considered correct, if the span, the type, and the subtype were correctly identified. As the table depicts, identifying and classifying *Justice*-events resulted in the highest F_1-score of 70.98%. Events of type *Manufacture* were not retrieved by our classifier, which we

[6]We hold out 25% of the provided training documents to optimize hyper-parameters.

System	**Plain**	**Type**	**Realis**
Human upper bound	76.57%	-	-
Our system	65.31%	55.56%	49.16%
Rank 1	**65.31%**	58.41%	**49.16%**
Rank 2	63.66%	57.18%	48.70%
Rank 3	62.49%	55.83%	47.05%
Rank 4	60.77%	**55.56%**	45.54%
Median	58.52%	48.79%	39.33%

Table 5: Official event nugget detection results (micro F_1-score) for Plain=detecting event nuggets, Type=detecting event nuggets with correct event type, Realis=detecting event nuggets with correct realis value. Human upper bound was determined as described in section 2. Our system ranks numer 1, 4, and 1 out of 14 submitted systems for the categories Plain, Type, and Realis.

believe is mainly due to the small number of training instances. Only 22 out of 6538 event nuggets are tagged as a *Manufacture*-event. As noted in section 2, *Contact*-events are fairly noisy and the annotations appear to be inconsistent. The submitted system achieves for those events an F_1 score of just 32.70%.

Event Type	**Precision**	**Recall**	F_1
Business	77.94%	44.92%	56.99%
Conflict	61.18%	49.19%	54.53%
Contact	34.81%	30.83%	32.70%
Justice	75.19%	67.21%	70.98%
Life	72.08%	48.91%	58.28%
Manufacture	–	0.00%	0.00%
Movement	51.47%	38.22%	43.86%
Personnel	74.50%	50.67%	60.32%
Transaction	54.19%	32.99%	41.01%

Table 6: Overview of performance per event type. Each event nugget must match exactly on the type and subtype.

Table 7 shows the confusion matrix for the submitted system and for the different event types. We can observe that the most errors come from not detecting an event as an event (the right *No*-column). Those count for 63% of the errors. The second largest source of error is that a token is wrongly classified as an event nugget (the bottom *No*-row). Those count for 29% of the errors. A wrong event type counts only for 8% of the errors. Wrongly classified *Contact*-events count for 28% of the errors. We can conclude that distinguishing between the event types is comparatively easy, while detecting that a token or a phrase is an event nugget is much more challenging.

That 63% of the errors originate from the fact that an event nugget was not identified as an event indicates that generalization is a big challenge in the detection of event nuggets. The system works well for event nuggets that it has seen in the training data, but it fails to learn the general concept of how to spot an event in a text. That in the training data only events of a certain type were annotated is an additional challenge, as the system must not only learn how to detect the textual representation of an event, but also how to spot only the textual representation of certain events. As distinguishing between event types appears to be much easier, a more suitable approach might be to first learn the general concept of events and then in the next step to decide whether the event matches one of the predefined event types or not.

7 Event Coreference Resolution

For the task of Event Coreference Resolution we decided to implement a simple baseline by using Gradient Boosted Decision Trees (Breiman, 1997; Friedman, 2000; Friedman, 2002). We model this task as a binary mention pair classification problem, predicting whether two event triggers belong to the same coreference chain or not. Such models have been very effective for coreference resolution (Soon et al., 2001; Bengtson and Roth, 2008). When combined with a secondary clustering algorithm, such approaches can achieve state-of-the-art results (Clark and Manning, 2015).

For the gradient boosted decision trees, we use the existent library XGBoost [7]. We decided to use a simple clustering strategy and to merge all clusters with a positive link between them, i.e. if an event e_1 in cluster 1 and an event e_2 in cluster 2 refer to the same event (according to our classifier), the two clusters are merged together. This approach achieves an overall average CoNLL score of 70.02% for task 3 with given event nuggets.

[7] https://github.com/dmlc/xgboost

Predicted

Actual

	Bu	Co	Ca	Ju	Li	Ma	Mo	Pe	Tr	No
Bu	**53**	0	0	0	0	0	0	0	1	64
Co	0	**368**	2	0	33	0	8	4	0	325
Ca	0	4	**784**	13	7	0	15	0	2	654
Ju	0	3	7	**1017**	13	0	3	0	4	414
Li	0	72	1	32	**319**	0	0	7	1	212
Ma	0	1	1	0	0	**0**	0	0	0	88
Mo	0	5	8	37	1	0	**202**	1	9	242
Pe	0	4	2	0	3	0	9	**274**	5	222
Tr	1	4	0	4	1	0	9	0	**386**	477
No	14	134	505	203	60	0	129	67	129	**0**

Table 7: Confusion matrix for the extracted event nuggets split by the different types. Bu=Business, Co=Conflict, Ca=Contact, Ju=Justice, Li=Life, Ma=Manufacture, Mo=Movement, Pe=Personnel, Tr=Transaction, No=Not annotated

7.1 Setup of the Training

For a certain event class, for example `Conflict_Attack`, all possible event pairs in a document were considered. We only matched to compatible event classes, i.e. the other event class must be of the same subtype (except for the `Contact_*` and `Transaction_*`, according to the annotation guideline those can be matched against any other event of the same main type). The pairs were ordered, the event appearing first in the document was put to the first position.

The classification of the pairs was done using Gradient Boosted Decision Trees (Breiman, 1997; Friedman, 2000; Friedman, 2002). Gradient Boosted Decision Trees are well known for their fast speed and their accurate predictive power. The model uses hundreds of decision trees in combination with a gradient boosting step. We hold out 25% of the training documents and tuned the hyperparameters on those documents. For the final model we set the parameters to 250 decision trees with a maximal depth of 3. In order to generate the final coreference chain, we merged all events together that were positively classified.

7.2 Feature Extraction

The features we extracted can be distinguished between features for each event and those that are computed for the pair. The features we tested for the individual events are: token, lemma, POS, subject & object (based on dependency links), semantic arguments (A0-A4), event mention type, and realis values.

Besides features for individual event nuggets, we also computed some features for the pair of two events: binary decision if the token, lemma, or POS are identical, cosine similarity of the two tokens/lemmas using the word embeddings by Levy and Goldberg (2014), cosine similarity between subjects / objects, number of tokens/sentences between the events, binary decision if the two lemmas appear in the same wordnet synset, and sentence similarity based on a bag-of-words representation and the Jaccard-index.

7.3 Evaluation

We hold out 25% of the training documents and tuned the hyper-parameters and the features on those documents. The accuracy for the binary pair classification task is depicted in Table 8. With all features enabled, the system achieves an accuracy of 87.03% on our development set. Removing the sentence similarity feature decreases the performance to 85.32%. The baseline, classifying all pairs with the majority class, achieves an accuracy of 82.90%.

Feature	**Accuracy**
Baseline: Majority Class	82.90%
Final System	87.03%
without distance features	86.78%
without token similarity	86.54%
without sentence similarity	85.32%

Table 8: Accuracy on our development set for the binary pair classification task if two events corefer.

For our final system we used the following features: lemma, binary decision if lemma or tokens are identical, cosine similarity between the two tokens / lemmas, binary decision if the synsets overlap, distance measured in tokens and sentences between the two events, and the sentence similarity based on the Jaccard-index (Lyon et al., 2001). The official overall average CoNLL score is 70.02% for this proposed system.

Type	**Avg CoNLL score**
Rank 1	75.69%
Rank 2	74.28%
Rank 3	72.60%
Our system (rank 4)	70.02%

Table 9: Official score for the event coreference task.

We could observe that the discussion forum posts contained a lot of quotes from previous posts. Event nuggets in these quotes obviously refer to the same event as in the original post and it was straight forward to identify those. Detecting other coreferring event nuggets is much more challenging.

8 Conclusion and Future Work

Using no other feature than the token and the neighboring tokens, a fairly high F_1 score of 61.73% for event nugget detection can be achieved using a deep feed forward network. Adding further local features did not impact the classification result, while more context aware features resulted in a small improvement. Our proposed system is publicly available [8].

For event coreference resolution, we showed that a simple pair-based classifier with a few features can achieve an average CoNLL score of 70.02% using gradient boosted decision trees. Especially measuring the similarity of the two sentences led to a big performance increase. Our current clustering strategy is suboptimal. It merges two event clusters if they have a positive link, i.e. the clusters are merged if an event e_1 in cluster 1 and an event e_2 in cluster 2 refer to the same event (according to the classifier). A single positive link will cause that two clusters are merged, even if all other pair combinations are in disfavor of merging the two clusters. A more advanced clustering algorithm, for example as used in (Clark and Manning, 2015), could significantly improve the results.

[8] `https://github.com/UKPLab/tac2015-event-detection`

Acknowledgement

This work has been supported by the Volkswagen Foundation as part of the Lichtenberg-Professorship Program under grant No. I/82806 and by the German Federal Ministry of Education and Research (BMBF) under the promotional reference 01UG1110D (DARIAH-DE). Additional support was provided by the German Federal Ministry of Education and Research (BMBF) as a part of the Software Campus program under the promotional reference 01-S12054.

References

Eric Bengtson and Dan Roth. 2008. Understanding the Value of Features for Coreference Resolution. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '08, pages 294–303, Stroudsburg, PA, USA. Association for Computational Linguistics.

Leo Breiman. 1997. Arcing the edge. Technical report.

Kevin Clark and Chris Manning. 2015. Entity-centric coreference resolution with model stacking. In *Proceedings of the 53nd Annual Meeting of the Association for Computational Linguistics, ACL 2015*.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel P. Kuksa. 2011. Natural Language Processing (almost) from Scratch. *Journal of Machine Learning Research*, 12:2493–2537.

Richard Eckart de Castilho and Iryna Gurevych. 2014. A broad-coverage collection of portable nlp components for building shareable analysis pipelines. In *Proceedings of the Workshop on Open Infrastructures and Analysis Frameworks for HLT*, pages 1–11, Dublin, Ireland, August. Association for Computational Linguistics and Dublin City University.

Jerome H. Friedman. 2000. Greedy function approximation: A gradient boosting machine. *Annals of Statistics*, 29:1189–1232.

Jerome H. Friedman. 2002. Stochastic gradient boosting. *Comput. Stat. Data Anal.*, 38(4):367–378, February.

Omer Levy and Yoav Goldberg. 2014. Dependency-Based Word Embeddings. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, ACL 2014, June 22-27, 2014, Baltimore, MD, USA, Volume 2: Short Papers*, pages 302–308.

Caroline Lyon, James Malcolm, and Bob Dickerson. 2001. Detecting short passages of similar text in large document collections. In *Proceedings of the 2001 Conference on Empirical Methods in Natural Language Processing*, pages 118–125.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems*, pages 3111–3119.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global Vectors for Word Representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP 2014)*, pages 1532–1543.

Erik F. Tjong Kim Sang and Sabine Buchholz. 2000. Introduction to the CoNLL-2000 Shared Task: Chunking. In *Proceedings of CoNLL-2000 and LLL-2000*, Lisbon, Portugal.

Wee Meng Soon, Hwee Tou Ng, and Daniel Chung Yong Lim. 2001. A Machine Learning Approach to Coreference Resolution of Noun Phrases. *Comput. Linguist.*, 27(4):521–544, December.

Song Zhiyi, Ann Bies, Strassel, Tom Riese, Justin Mott, Joe Ellis, Jonathan Wright, Seth Kulick, Neville Ryant, and Xiaoyi Ma. 2015. From Light to Rich ERE: Annotation of Entities, Relations, and Events. In *Proceedings of the 3rd Workshop on EVENTS at the NAACL-HLT 2015*, pages 89–98.