DeepCause: Hypothesis Extraction from Information Systems Papers with Deep Learning for Theory Ontology Learning

Roland M. Mueller
Berlin School of Economics and Law
roland.mueller@hwr-berlin.de

Sardor Abdullaev
Berlin School of Economics and Law
sardorabdullaev91@gmail.com

Abstract

This paper applies different deep learning architectures for sequence labelling to extract causes, effects, moderators, and mediators from hypotheses of information systems papers for theory ontology learning. We compared a variety of recurrent neural networks (RNN) architectures, like long short-term memory (LSTM), bidirectional LSTM (BiLSTM), simple RNNs, and gated recurrent units (GRU). We analyzed GloVe word embedding, character level vector representation of words, and part-of-speech (POS) tags. Furthermore, we evaluated various hyperparameters and architectures to achieve the highest performance scores. The prototype was evaluated on hypotheses from the AIS basket of eight. The F1 result for the sequence labelling task of causal variables on a chunk level was 80%, with a precision of 80% and a recall of 80%.

1. Introduction

There is an exponentially increasing number of published papers in journals and conferences [4, 22]. This makes it more and more difficult to get an integrated view of the different theories and their relationships [30].

Figure 1 shows the traditional way of scientific publication at the bottom of the diagram and the potential role of theory ontologies and theory ontology learning at the top of the diagram. In the traditional scientific publishing system, an author would describe his or her mental model or theory in 5 to 30 pages of text, and the reader would have to reconstruct the mental model when reading the paper. For a few papers this is feasible but becomes nearly impossible with thousands of papers from different scientific disciplines. Literature databases support the reader only with full-text search and co-citation analysis. Theory ontologies might offer further, more effective ways for the reader to reconstruct the mental models of other authors. Theory ontologies might be used for creating internomological networks [25], inter-theory relationships with theory evolution graphs [30], or theory-data maps [31]. Theory ontologies can be manually created by authors, readers, and editors or semi-automatically by analyzing scientific texts through natural language processing methods in a theory ontology learning step. In the behavioral sciences in general, and for information systems in particular, theory ontology learning is suggested to overcome the lack of theory integration [21, 32].

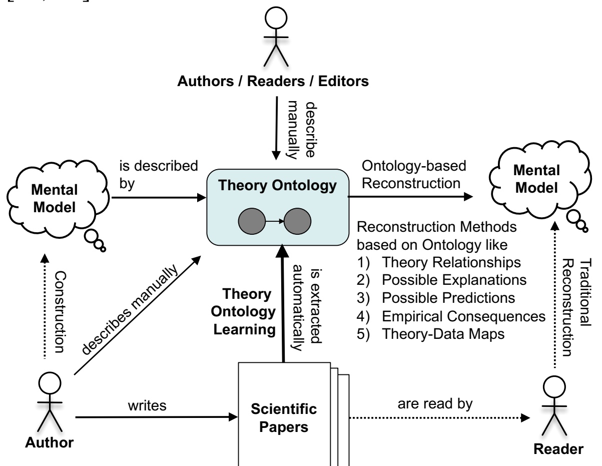

Figure 1. Role of theory ontology learning for theory meta-analysis

One example of a system for theory ontology learning is presented by Mueller and Huettemann [32]. They introduced the prototype CauseMiner for causal relationship extraction from hypotheses and propositions of information systems papers. CauseMiner used a number of natural language processing rules and cues for extracting causes, effects, signs, mediators, moderators, conditions, and interaction signs from hypotheses (see Figure 2). However, they did not use any machine learning approach for extracting the different elements of a hypothesis [32]. With this paper we want to build upon these finding and analyze the efficacy of machine learning, especially deep learning methods, for the

URI: https://hdl.handle.net/10125/60059
ISBN: 978-0-9981331-2-6

cause-effect extraction task. We present a deep learning-based prototype called DeepCause for theory-ontology learning.

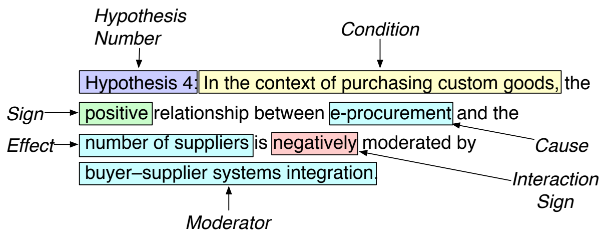

Figure 2. Example of a hypothesis [8] and the extracted parts [32]

Consequently, this paper tries to answer the following research questions:

RQ 1: *Are deep learning methods for sequence labelling capable to extract cause–effect relationships for theory ontology leaning?*

RQ 2: *What are the most effective deep learning architectures for the cause–effect extraction tasks?*

The next chapter will discuss relevant related literature. The developed DeepCause artifact will be discussed in Chapter 3. Here, we also elucidate the experiment setups and the design rationale behind the design decisions. Chapter 4 explains the evaluation criteria, metrics, and the results of the experiments. Finally, the conclusion illustrates the results of the work, answers the research questions, and discusses the limitations of this study and the possibilities for future work.

2. Related work

The existing literature about the automatic extraction of causal relationships from natural language can be classified into rule-based methods and machine learning-based methods [1]. Except for CauseMiner [32], no other system tries to identify additional elements to cause and effect, like moderating variables, mediating variables, or conditions. Also, no paper is using deep neural networks for causal relationship extraction, yet [1:8]. However, Asghar [1:8] suggested this might be effective because of the feature abstraction capabilities of deep learning. This paper tries to fill this research gap by examining the possibilities of deep learning for causal relationship extraction.

Deep learning describes neural network architectures with multiple hidden layers [23]. Besides dense networks that connect all neurons from one layer to all other neurons of the next layer, also other special deep learning architectures were developed. Convolutional neural networks (CNN) [23] try to better learn translation invariant features by using a shared moving 1-dimensional (for text) or 2-dimensional (for images) window (also called stride). Recurrent neural networks (RNN) are used for analyzing sequential data like the sequence of words in natural language. An RNN processes the input sequence one token at a time but also maintains a kind of hidden state that captures parts of the history of the past elements. However, a simple RNN has problems to learn long-term dependencies in the sequences because it suffers from the so-called vanishing gradient problem. Long short-term memory (LSTM) networks [15] are special RNNs that use forget gates to learn how long past elements of the sequence are relevant and therefore are better in learning long-term dependencies in sequences. In written text, the meaning of a word is not just dependent on the preceding words but also on the succeeding words. Bidirectional LSTMs (BiLSTMs) split the neurons of a regular LSTM into forward states that are connected with past states (positive time direction) and backward states that are connected with future states (negative time direction).

Word embedding improved the performance of many NLP tasks. In word embedding, individual words are represented as a vector of, for example 300 dimensions, that are learned in an unsupervised manner on a large corpus [29]. These word-embedding vectors capture the semantics of words and can be used to calculate the similarity of words. Popular word embedding methods, which have also pretrained models that are trained on billions of tokens, are word2vec [29], GloVe [35] and fastText [3].

A deep learning model can have different possible outputs [23]. In a simple classification task, like sentiment mining, a sequence of words would be classified as a whole into one class (e.g., as positive or negative sentiment). In sequence labelling, for each input token an output label is generated. Hence, the output has the same length as the input. Typical sequence labelling tasks are part-of-speech (POS) tagging or named entity recognition (NER). We want to use sequence labelling for the extraction of the parts in a sentence that constitutes a cause, effect, moderator, or mediator. For sequence labelling, conditional random fields (CRF) are often used. CRFs are discriminative undirected probabilistic graphical models that take the neighboring input into account [27].

For finding best practices for deep learning-based sequence labelling, we looked at recent studies on the topic (see Table 1). The studies in Table 1 were analyzed according to the extent a method is suitable for our problem and whether the selected method is state-of-the-art for the time being.

Table 1. Related studies of sequence labelling and LSTM

Task / Paper	Methods	Data Inputs	Setting
End-to-end sequence labelling via bidirectional LSTM-CNNs-CRF [27]	CNN for character-level representation, BiLSTM, CRF	Penn Treebank WSJ corpus (for POS tagging), CoNLL 2003 corpus	Stanford's publicly available GloVe 100-dimensional embedding
Natural language processing (almost) from scratch. Part-of-speech tagging, chunking, named entity recognition and semantic role labelling [7]	CNN and SLNN (sentence-level likelihood neural nets) with CRF with unary potential (conditional tag path probability). CNN for the whole sentence and for the window, convex setup	Wikipedia, Brown cluster, CoNLL 2003 Reuters, Wall Street Journal	Substring features, word features, and gazetteer features
Bidirectional LSTM-CRF models for sequence tagging [17]	LSTM, BiLSTM, CRF, LSTM-CRF and BiLSTM-CRF	CoNLL2000, CoNLL2003	SENNA embedding, gazetteer features, spelling features, context features, batch size of 100, BIO2 tagging schema, learning rate of 0.1, hidden layer size of 300
Named entity recognition with bidirectional LSTM-CNNs [5]	Hybrid BiLSTM and CNN architecture. CNN for character level representation	CoNLL-2003, OntoNotes 5.0	Character features using a convolutional neural network, 50-dimensional word embedding (50 Dims.) based on Wikipedia and the Reuters RCV-1 corpus, GloVe and word2vec on Google News, additional word and character features
Effect of non-linear deep architecture in sequence (named entity recognition and syntactic chunking) [40]	CRF and SLNN models	CoNLL-2003, MUC, ACE	L-BFGS optimization algorithm, L2-regularization, word embedding trained over Wikipedia text, BIO2 tagging
Empower sequence labelling with task-aware neural language model (chunking and POS tagging) [26]	LM-LSTM-CRF (multitask learning with highway layers)	CoNLL 2003, CoNLL 2000, Wall Street Journal	Pre-trained word embedding (GloVe 100-dimension pre-trained), hidden state size of LSTM 300, mini-batch, depth of highway layer is 1
Neural architectures for named entity recognition [20]	LSTM, CRF and a transition-based approach inspired by shift-reduce parsers (stack LSTM)	CoNLL-2002, CoNLL-2003	Character-based word representations learned from the supervised corpus and unsupervised word representations learned from unannotated corpora, IOBES tagging schema, character-based features, pre-trained embedding, LSTM-CRF model with a single LSTM layer
Optimal hyperparameters for deep LSTM-networks for sequence labelling tasks [36]	LSTM, BiLSTM-CRF, BiLSTM-CNN-CRF, BiLSTM-LSTM-CRF	CoNLL 2003, Reuters, Wall Street Journal, ACE 2005, TempEval3	Experiments with more than 50,000 combinations of hyperparameters for Sequence labelling tasks were conducted to estimate the influence of each hyperparameter
Unified DL architecture for NLP with multitask learning (SRL, NER, POS) [6]	Time-delay neural networks (TDNNs similar to CNN)	English Wikipedia	Word embedding, word and sentence level features
Deep semantic role labelling: what works and what's next [13]	Deep highway BiLSTM architecture with constrained decoding, ensemble model	CoNLL 2005, CoNLL 2012	8 BiLSTM layers (4 forward LSTMs and 4 reversed LSTMs) with 300-dimensional hidden units, and a softmax layer, orthonormal initialized weight matrices, GloVe embedding (100 dim) pre-trained, updated during training
Named entity recognition with long short-term memory [12]	LSTM with 2 passes	Reuters corpus, English language, volume 1, and European corpus initiative multilingual corpus 1	Each sentence is presented word by word in two passes; first pass to accumulate information for disambiguation in second pass; in second pass network is trained to output vector representation of relevant output tag. no momentum and direct connections from input to output layers for 100 iterations.
Multi-task cross-lingual sequence tagging from scratch [41]	GRU, word-level GRU, CRF, character-level GRU, hierarchical GRU	Penn Treebank (PTB), POS tagging, CoNLL 2000 chunking, CoNLL 2003	One-hot gazetteer features, pre-trained word embedding (SENNA embedding trained on Wikipedia), polyglot embedding with fine-tuning, hidden state dimensions to be 300 for the word-level GRU, BIOES.

Table 2. Best practices for LSTM and sequence labelling

1	Optimizer	Clip the gradients of LSTM weights so that their norm is bounded by value 1.0 [34]
2	Bias	Adding a bias of 1 to the forget gate of LSTM closes the gap between LSTM and GRU. LSTM with the large forget bias outperformed both LSTM and GRU on almost all tasks [18]
3	Dropout	Regularizing using dropout is ineffective for RNN networks and [42] shows the ways to avoid it; similar but more advanced solution utilized in [10]
4	Neural Network	Implement dense concatenation through the layers [16]
		Sequence Labelling Best Practices
5	Neural Network	Adding an extra dense layer between BiLSTM and output layers [20]
6	Neural Network	Connect some features directly to an output layer, skipping RNN layers [17]
7	Neural Network	For the initialization of the matrix weight parameters in the neural network use "Xavier" initialization [11]
8	Optimizer	Use Nesterov momentum with Adam optimizer [9]
9	Neural Network	Use two BiLSTM layers [36]

Based on the studies (Table 1), the following design decisions might improve the performance of sequence labelling tasks: use of conditional random fields (CRF) classification, part-of-speech (POS) tags as part of input, use of character level embedding, and use of bidirectional LSTM models. Several additional best practices and tweaks are summarized in Table 2.

3. DeepCause artifact

3.1. Training, development and test data

We collected propositions and hypotheses from information system papers from the AIS basket of eight. For each hypothesis we labelled causes, effects, moderators, and mediators. Table 3 shows the training and test data. All datasets have a tabular structure with the columns Hypotheses, Cause, Effect, Moderator, and Mediator. Each row represents a sentence. Dataset 1 is manually constructed. In contrast, Dataset 2 (pseudo labelled training data) contains pseudo-labelled data produced by CauseMiner [32]. Lee [24] suggested that pseudo-labelled data might improve results.

Table 3. Datasets

Dataset	Explanation	Source	Rows
Dataset 1 (training/test)	Manually labelled sentences	Information System papers from the AIS basket of eight	1975
Dataset 2 (pseudo labelled training data)	Automatically labelled sentences	Sentences labelled by CauseMiner from Information System papers from the AIS basket of eight	15179

We tested the impact of pseudo labels by creating two different splits with three data sets each: a training set, a development set, and a testing set.

First, for testing pseudo labels, we used 100% of Dataset 2 (pseudo-labelled data) and 50-60% of Dataset 1 for the training set, 20-25% of Dataset 1 for the development set, and 20-25% of Dataset 1 for the test set. We used the development set for the hyperparameter tuning procedure.

Second, for the experiments without pseudo labels we used only Dataset 1: 40-50% for training, 25-28% for development and 22-25% for testing.

The assignment to the training, development, and test sets were randomized and hypotheses from the same paper might not be kept in the same set.

3.2. Input features

We represented each sentence from the input dataset with three feature types: word embedding, part-of-speech (POS tags), and character embedding.

3.2.1. Word embedding. Before generating the features, we cleaned the raw data by trimming extra whitespaces and quotes. To quantify the input sentences, we used the pre-trained word embedding of GloVe [35], which is pre-trained on 6 billion tokens from Wikipedia and news articles with a vocabulary size of 400,000 words. The word embedding was not retrained by us, which means that it was not updated during the training. In addition, we initialized the padding words with zeros. Padding was used to align the input data in each batch.

3.2.2. POS tags. Khoo et al. [19] and Pakray and Gelbukh [33] illustrated diverse use-cases for POS in cause-effect extraction. In natural language, cause and effect could correlate to a particular POS tag sequence. Considering these facts, we added information about POS tags as a feature. POS tags were generated using the Python library spaCy [28].

3.2.3. Character embedding. Papers about POS and NER labelling tasks often use character embedding features because these features are good at detecting morphologic clues in words. Huang et al. [17], Ma and Hovy [27], Liu et al. [26], Lample et al. [20], and Chiu and Nichols [5] demonstrated significant improvements of their results. However, Reimers and Gurevych [36] stated that these features do not influence the accuracy substantially. We decided to use character embedding features.

For character embedding, we used the architecture of Ma and Hovy [27]. An embedding matrix was used so that each column corresponds to a single character. The width of the matrix was equal to the number of words in the dictionary and the height was a hyperparameter which could take a value of 15 or 30. In addition, a dropout layer was applied. The dropout rate was also a hyperparameter. Initial values of the matrix were drawn from the uniform distribution of the range [-0.05; 0.05]. Moreover, our character embedding features were trainable, in contrast to the word embedding. Next, we used one-dimensional convolutional layers with hyperbolic tangent as the activation function and the number of filters and the window size as hyperparameters. To retrieve the compactable features, we applied pooling layers on top of the character embedding. The max-pooling layer had a pool size of 35 (the length of the longest word). Finally, we stacked all three features for each word and sorted the entire input by the length of each row to reduce the padding during the training with minibatches.

3.3. Output layer

We labelled each word according to a selected tagging schema (see Table 4). In place of padding values, we used the 'out' tag 'O' (from BIO and IOB tagging) labels. We used one hot encoding for CRF classifiers and a compact encoding for softmax. These tags are the output that the sequence labelling algorithm should learn for each token in the input sequence.

Table 4. Used tagging schema

Tag	Meaning
B-Cause	Beginning of Cause Chunk
I-Cause	Cause Chunk (Inner or End)
B-Effect	Beginning of Effect Chunk
I-Effect	Effect Chunk (Inner or End)
B-Moderator	Beginning of Moderator Chunk
I-Moderator	Moderator Chunk (Inner or End)
B-Mediator	Beginning of Mediator Chunk
I-Mediator	Mediator Chunk (Inner or End)
O	Out (all other words)

3.4. Model

We tested different deep learning architectures for the recurrent layers: SimpleRNN, LSTM, BiLSTM, and GRU. Depending on the tested hyperparameters, the architectures can be modified significantly. For example, adding an extra dropout layer, duplicating a layer, removing a layer and partitioning of input, and so on. Figure 3 shows the used example of a deep learning model with a bidirectional LSTM, an extra dense layer, and a CRF output layer.

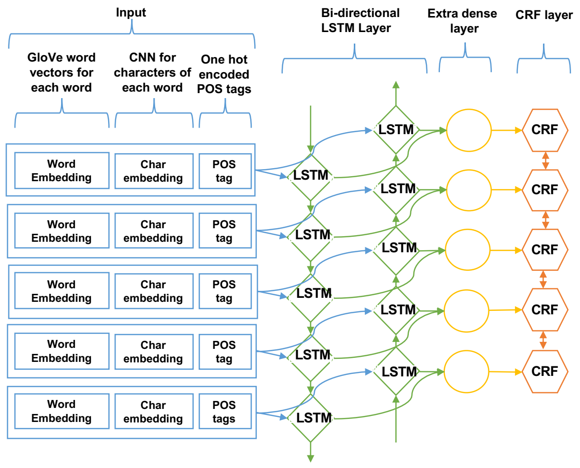

Figure 3. Exemplary tested model

We implemented most of the best practices from Table 2. However, we did not implement number 4 (dense concatenation through the layers) and 6 (connect some features directly to an output layer, skipping RNN layers) because of high implementation complexity with the used Keras framework and the relatively low expected impact. All other suggestions were implemented and tested, like the Xavier initialization [11] or adding an extra dense layer with a ReLU activation before the output layer [20]. For the output, we tested two different layers: a CRF classifier layer to catch joint pairwise dependencies between adjacent labels, and a softmax classification layer.

3.5. Hyperparameter tuning

We tested a list of values for different hyperparameters (Table 5). Also, the type of neural network was a hyperparameter for the model.

Table 5. List of hyperparameters

Category	Parameter	Values
NN model	Architecture	SimpleRNN, LSTM, BiLSTM, GRU
NN model	Output layer	CRF, CRF with an extra dense layer, Softmax
NN model	Minibatch size	1, 2, 4, 8, 16
NN input	Whether to use POS tags	True, False
NN input	Whether to use char embedding	True, False
Char embedding	Number of filters	15, 30
Char embedding	Number of dimensions	15, 30
Char embedding	Dropout	0, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9
Char embedding	Window length	3, 6
Word embedding	Dropout	0, 0.2, 0.4, 0.6
Word embedding	Number of dimensions	100, 200, 300
Recurrent layer	Number of recurrent units	100, 300
Recurrent layer	Number of recurrent layers	1, 2
Recurrent layer	Dropout	0.1, 0.2, 0.3, 0.5, 0.7, 0.9
Optimizer	Learning rate (only for SGD optimizer)	0.001, 0.005, 0.01, 0.05, 0.1, 0.5, 1.0
Optimizer	Decay	0, 1e-4, 1e-6
Optimizer	Momentum	0, 0.9
Optimizer	Nesterov optimization	True, False
Optimizer	Optimizer type	SGD, Adam, Nadam, RMSProp, Adagrad, Adadelta
Optimizer	Patience	4

Grid search became impossible due to a huge number of possible combinations. Similarly, random search [2] and Bayesian search [37] turned out to be too time-consuming. That is why for hyperparameter tuning we used an approach that combined ideas from both methods.

During the training we iterated over each group of hyperparameters to find the value producing the highest F1 score independently from the rest of parameters. On each step the value of the yet untraversed group of hyperparameters was just assigned to the first element. We shuffled the list of hyperparameters for each parameter several times to eliminate sticking to a local minimum and thus adding randomization, similarly to random search [2]. Before each shuffling, we modified the values for the hyperparameters based on the results.

3.6. Training

For each set of tuning parameters, we trained our model on the dataset as described in Section 3.1. The model was trained a maximum of 200 epochs with early stopping, if the loss function was not improving for 4 epochs (*patience)*. One epoch is one forward and one back propagation with all the training data. The number of epochs before the interruption (*patience)* was also used as a hyperparameter (see Table 5). We tried to achieve balance between underfitting and overfitting, as well as to meet performance requirements. The model was trained with batches. We forward propagated the input over the hidden layers, calculated the gradients, back propagated errors, and updated the weights of the neural network. As an output layer, we tested softmax and CRF.

4. Evaluation

4.1. Evaluation metrics

To evaluate the efficacy of our artifact and the impact of the different deep learning architectures, we analyzed all results and compared the performances of our models. Each configuration is a combination of hyperparameter values (Table 5) and is considered one experiment. We used the following metrics to evaluate the artifacts:

- Precision: The percentage of detected sequences (chunks) that are correct [38, 39]. If only one token was not correctly included or excluded to a sequence that describes a construct, then the sequence is not correctly detected.
- Recall: Percentage of sequences in the data that were found by the model [38, 39].
- F1 score: Harmonic mean of precision and recall [38, 39]. We used the F1 score to find the best hyperparameters because it incorporates both recall and precision.

4.2. Results of hyperparameter testing

4.2.1. Pseudo labels. Figure 4 shows a swarm plot of all experiments with and without pseudo labels. The output layer is color coded. We see that pseudo labels are not improving the results.

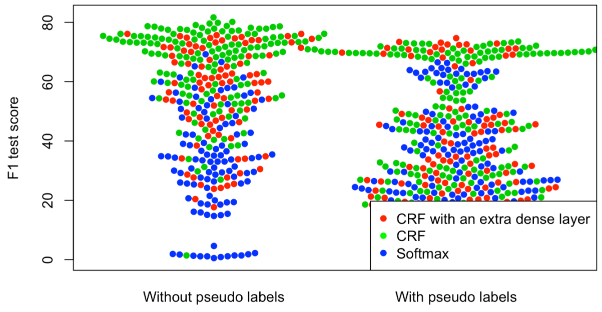

Figure 4. F1 scores with and without pseudo labels (color=output layer)

4.2.2. RNN architecture. Figure 5 shows the best performing experiments without pseudo labels for the different RNN architectures (Bidirectional LSTM, LSTM, GRU, SimpleRNN) and the different output layers. It shows that the best development F1 score was by bidirectional LSTM.

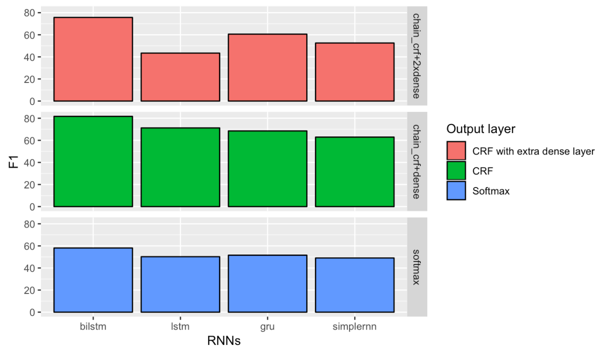

Figure 5. Best F1 score by RNN type and output layer (without pseudo labels)

4.2.3 Output layer. Figure 4 shows a swarm plot of all experiment with the output layer (CRF with extra dense layer, CRF, and softmax) encoded in the color. Figure 5 shows the impact of the output layers on the best accuracy. Both figures show that CRFs are beneficial. CRF without an extra dense layer showed slightly better results.

4.2.4. Optimization algorithm. Figure 6 shows the performance by the used optimization algorithm (SGD, Adam, Adagrad, Rmsprop, Nadam) and the RNN architectures. Figure 7 shows the performance of the used optimization algorithm and the used output layer. SGD performed worse than the other optimizers. The Adam algorithm outperforms the others. Unfortunately, we were not able to conduct a deeper investigation on all optimizers, especially Nadam, because our hyperparameter optimization algorithm did not include enough experiments with all optimizers.

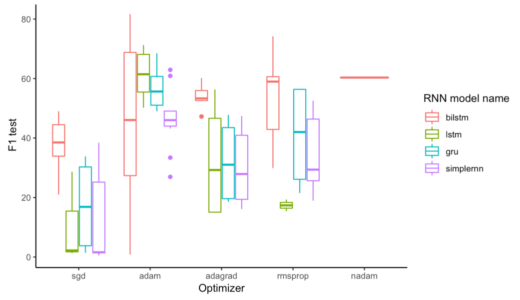

Figure 6. F1 score by optimization algorithm and RNN types

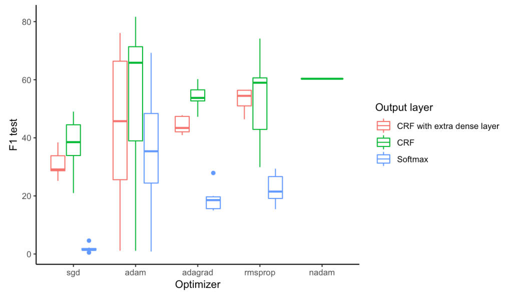

Figure 7. F1 score by optimization algorithm and output layer classifier

4.2.5. POS tags. Figure 8 shows the kernel density distribution (KDE) of all experiments with their F1 test scores for the experiments with POS tags (With) and without POS tags (Without). The data suggests that using POS tags can improve the model. However, we are not claiming that this effect is significant, because, as stated before, data may not be balanced enough.

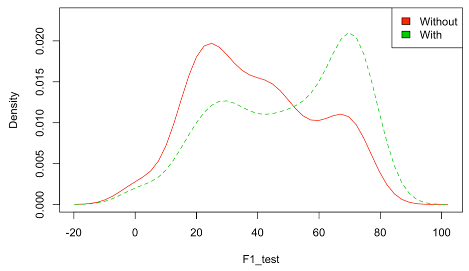

Figure 8. Influence of POS tags (With=with POS Tags) on F1 score

4.2.6. Character embedding. Figure 9 shows the kernel density distribution (KDE) of the F1 test score with character embedding (With) and without character embedding (Without). We observe a performance improvement with character embedding. However, the impact is smaller than for the POS tags.

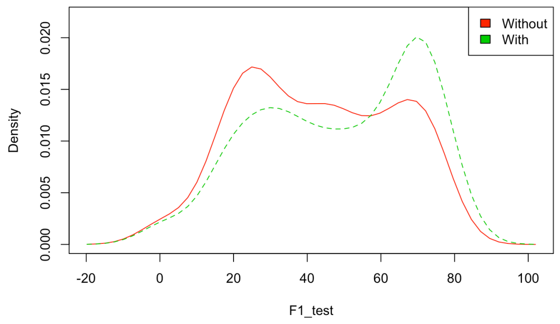

Figure 9. Influence of character embedding (With=with character embedding) on F1 score

4.2.7. Tagging scheme. The distributions of the F1 score for each tagging schema (Figure 10) suggest that the choice between BIO and IOB had no significant effect on the performance of the model.

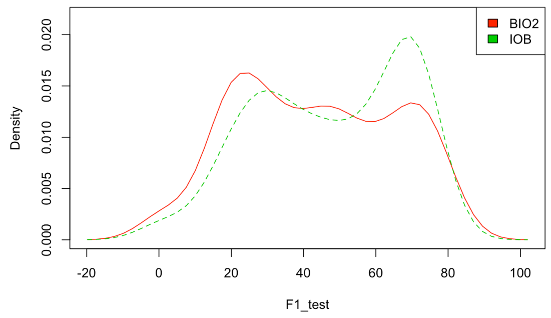

Figure 10. Influence of tagging schema (BIO2 or IOB) on F1 score

4.2.8. Patience. We used an early stopping rule so if the loss function was not improving for a number of epochs (*patience),* we would stop the learning process. A patience of 3 and 4 seemed to perform better but the impact was not very strong. We achieved the best results with a patience of 4.

4.2.9. Minibatch size. Smaller batch sizes (1, 2, and 4) seemed to perform better than larger batch sizes, but the influence was not very strong. We achieved the best results with a batch size of 1.

Table 6. Top three best results with the different configurations

Dev. Set	Test set			Archi-tecture	Output Layer	Embedding dim.	Optimizer	RNN dropout	Number of recurrent layers	Tagging schema	Pseudo labels	POS tags	char. CNN embedding
F1	Precision	Recall	F1										
79.15	80.32	80.0	80.16	BiLSTM	CRF	300	Adam	0.3	2	BIO2	FALSE	TRUE	TRUE
78.49	78.42	79.79	79.1	BiLSTM	CRF	100	Adam	0.4	1	BIO2	FALSE	TRUE	TRUE
76.85	80.14	79.58	78.86	BiLSTM	CRF	100	Adam	0.0	1	BIO2	FALSE	TRUE	TRUE

4.3. Results

Table 6 shows the top three results based on the F1 score of the development set with the used hyperparameters. All of the top three results did not use pseudo labelling. For the results without pseudo labelled data, the training set size was 987 sentences and the development and training set size was 494 sentences.

The best F1 score was approximately 80.2% with precision of 80.3% and recall of 80.0%. The confusion matrices of cause and effect chunks are depicted in Table 7 and Table 8 (based on the best result). Because the total number of chunks of moderators and mediators were only around 30, we did not include a confusion matrix for these tags.

Table 7. Confusion matrix for cause on a chunk level

		Predicted	
		Cause chunks	Not cause chunks
Actual	Cause chunks	291	61
	Not cause chunks	65	-

Table 8. Confusion matrix for effect on a chunk level

		Predicted	
		Effect chunks	Not effect chunks
Actual	Effect chunks	288	86
	Not effect chunks	82	-

5. Conclusion

Table 9 shows the design science process and contributions of our paper, structured based on the design science guidelines of Hevner et al. [14].

Research Question 1 asked if deep learning methods for sequence labelling were capable to extract cause-effect extraction. The best model achieved a F1 score of 80% on the chunk level. Table 10 shows the results of the rule-based system CauseMiner [32] for cause and effect tested on 564 sentences. When we compare the performances of DeepCause and CauseMiner, we see that the F1 score of CauseMiner is slightly smaller (79% on a chunk level).

Rule-based systems like CauseMiner and machine learning-based systems like DeepCause have both their benefits and drawbacks for cause-effect extraction. Rule-based systems are often brittle, they need a huge effort and special domain knowledge to be built, they are optimized only for a specific use case, and they are not leveraging the possibilities of more training data. However, they do not need so much training data and complex patterns can be described by the rules. By contrast, machine learning systems need a lot of annotated data and even for end-to-end models like deep learning, complex feature engineering steps are necessary. However, with more data their performance will improve up to certain a point. Moreover, the same architecture can also handle slightly different annotation tasks.

Table 9. Design science contributions

Guideline [14]	Contribution
Design as an Artifact	The research outcomes: 1. DeepCause artifact for extracting elements of a hypothesis 2. Influence of different model parameters on the performance
Problem Relevance	Because of the exponentially increasing number of scientific papers, we need methods for extracting causal claims for theory ontology learning
Design Evaluation	Evaluation of the artifact based on a test set, measured on F1 score, accuracy, precision, and recall
Research Contributions	Compared to the rule-based CauseMiner tool, this paper shows the potential of deep learning for hypothesis extraction
Research Rigor	Creating a gold standard, split of the data into training, development, and test sets
Design as a Search Process	Extensive test of different hyperparameters
Communication of Research	Detailed information of the best deep learning architectures

Table 10. Performance of CauseMiner [32]

Variable	Prec.	Recall	F1	Acc.
Cause	0.79	0.79	0.79	0.66
Effect	0.77	0.81	0.79	0.66

Research Question 2 asked for the most effective deep learning architectures for the cause-effect extraction task. From an extensive literature review we distilled best practices in the field and carried out a range of experiments with possible deep learning configurations.

We found that pseudo labelling did not improve the results. One reason is that pseudo-labelled data is error-prone as Table 10 shows, so that DeepCause also learns partially wrong patterns when using pseudo labels.

We found a bidirectional LSTM (BiLSTM) as the most effective RNN-architecture. For the output layer we found conditional random fields (CRF) without an extra dense layer as most promising. Also, using POS tags and character embedding improved the results. We used pre-trained GloVe word embedding vectors.

One of the limitations of the study is the size of the dataset which comprised of only 1,975 manually labelled sentences and 15,179 pseudo labelled sentences. Therefore, in future work we will improve the quantity and quality of our data.

We did not analyze if the publication year of the hypotheses would have an influence on the accuracy of the model. However, we do not expect that the way hypotheses are phrased in English language changed dramatically over the years. Therefore, we would not expect a big influence of the publishing year.

Deep learning models involve a lot of parameters and analyzing all of them was an intractable task for this paper. For example, stacked bidirectional LSTM models with CRF classifiers require high processing power. We could not test all combinations of the hyperparameters and used a special hyperparameter tuning approach instead.

There are multiple opportunities for further research. An interesting possible solution regarding the layer connections are highway networks [43] that demonstrated to be beneficial for sequence modelling tasks. Also, we might try multitask learning [6] to extract cause, effect, moderator, and mediator, as well as word level features as different tasks. Also, a trained word embedding on a large number of information systems papers might be promising. The effect of dropout should be studied more detailed to regularize the models properly.

One problem of our artifact was the muddling of cause and effect chunks. A possible solution would be to use sequence labelling just for construct chunk extraction and an additional classifier to predict which of the different chunks are cause, effect, etc. An ensemble-based system that combines the rule-based method of CauseMiner and the machine-learning method of DeepCause might be an opportunity to increase accuracy even further.

6. References

[1] Asghar, N., “Automatic Extraction of Causal Relations from Natural Language Texts: A Comprehensive Survey”, *arXiv preprint arXiv:1605.07895*, 2016.

[2] Bergstra, J., and Y. Bengio, “Random search for hyper-parameter optimization”, *Journal of Machine Learning Research 13*(Feb), 2012, pp. 281–305.

[3] Bojanowski, P., E. Grave, A. Joulin, and T. Mikolov, “Enriching word vectors with subword information”, *arXiv preprint arXiv:1607.04606*, 2016.

[4] Bornmann, L., and R. Mutz, “Growth rates of modern science: A bibliometric analysis based on the number of publications and cited references”, *Journal of the Association for Information Science and Technology 66*(11), 2015, pp. 2215–2222.

[5] Chiu, J.P., and E. Nichols, “Named entity recognition with bidirectional LSTM-CNNs”, *arXiv preprint arXiv:1511.08308*, 2015.

[6] Collobert, R., and J. Weston, “A unified architecture for natural language processing: Deep neural networks with multitask learning”, *Proceedings of the 25th international conference on Machine learning*, ACM (2008), 160–167.

[7] Collobert, R., J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa, “Natural language processing (almost) from scratch”, *Journal of Machine Learning Research 12*(Aug), 2011, pp. 2493–2537.

[8] Dedrick, J., S.X. Xu, and K.X. Zhu, “How Does Information Technology Shape Supply-Chain Structure? Evidence on the Number of Suppliers”, *Journal of Management Information Systems 25*(2), 2008, pp. 41–72.

[9] Dozat, T., “Incorporating nesterov momentum into adam”, *ICLR Workshop*, (2016).

[10] Gal, Y., and Z. Ghahramani, “A theoretically grounded application of dropout in recurrent neural networks”, *Advances in Neural Information Processing Systems 29*, (2016), 1019–1027.

[11] Glorot, X., and Y. Bengio, “Understanding the difficulty of training deep feedforward neural networks”, *Proceedings of the 13th International Conference on Artificial Intelligence and Statistics (AISTATS)*, (2010), 249–256.

[12] Hammerton, J., “Named entity recognition with long short-term memory”, *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003-Volume 4*, Association for Computational Linguistics (2003), 172–175.

[13] He, L., K. Lee, M. Lewis, and L. Zettlemoyer, “Deep semantic role labeling: What works and what’s next”, *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, (2017).

[14] Hevner, A.R., S.T. March, J. Park, and S. Ram, “Design science in information systems research”, *MIS Quarterly 28*(1), 2004, pp. 75–105.

[15] Hochreiter, S., and J. Schmidhuber, “Long short-term memory”, *Neural computation 9*(8), 1997, pp. 1735–1780.

[16] Huang, G., Z. Liu, K.Q. Weinberger, and L. van der Maaten, “Densely connected convolutional networks”, *Proceedings of the IEEE conference on computer vision and pattern recognition*, (2017), 3.

[17] Huang, Z., W. Xu, and K. Yu, “Bidirectional LSTM-CRF models for sequence tagging”, *arXiv preprint arXiv:1508.01991*, 2015.

[18] Jozefowicz, R., W. Zaremba, and I. Sutskever, “An empirical exploration of recurrent network architectures”, *International Conference on Machine Learning*, (2015), 2342–2350.

[19] Khoo, C.S., J. Kornfilt, R.N. Oddy, and S.H. Myaeng, “Automatic extraction of cause-effect information from newspaper text without knowledge-based inferencing”, *Literary and Linguistic Computing 13*(4), 1998, pp. 177–186.

[20] Lample, G., M. Ballesteros, S. Subramanian, K. Kawakami, and C. Dyer, “Neural architectures for named entity recognition”, *arXiv preprint arXiv:1603.01360*, 2016.

[21] Larsen, K.R., S. Michie, E.B. Hekler, et al., “Behavior change interventions: the potential of ontologies for advancing science and practice”, *Journal of Behavioral Medicine 40*(1), 2017, pp. 6–22.

[22] Larsen, P.O., and M. von Ins, “The rate of growth in scientific publication and the decline in coverage provided by Science Citation Index”, *Scientometrics 84*(3), 2010, pp. 575–603.

[23] LeCun, Y., Y. Bengio, and G. Hinton, “Deep learning”, *Nature 521*(7553), 2015, pp. 436–444.

[24] Lee, D.-H., “Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks”, *Workshop on Challenges in Representation Learning, ICML*, (2013), 2.

[25] Li, J., and K. Larsen, “Establishing Nomological Networks for Behavioral Science: a Natural Language Processing Based Approach”, *ICIS 2011 Proceedings*, 2011.

[26] Liu, L., J. Shang, F. Xu, et al., “Empower Sequence Labeling with Task-Aware Neural Language Model”, *arXiv preprint arXiv:1709.04109*, 2017.

[27] Ma, X., and E. Hovy, “End-to-end sequence labeling via bi-directional lstm-cnns-crf”, *arXiv preprint arXiv:1603.01354*, 2016.

[28] Matthew, H., “Website - spaCy: Industrial-Strength Natural Language Processing”, 2017. https://spacy.io

[29] Mikolov, T., I. Sutskever, K. Chen, G.S. Corrado, and J. Dean, “Distributed representations of words and phrases and their compositionality”, *Advances in neural information processing systems*, (2013), 3111–3119.

[30] Mueller, R.M., “A Meta-Model for Inferring Inter-Theory Relationships of Causal Theories”, *48th Hawaii International Conference on System Science (HICSS)*, IEEE (2015), 4908–4917.

[31] Mueller, R.M., “Theory-Data Maps: A Meta-Model and Methods for Inferring and Visualizing Relationships between Causal Theories and Empirical Evidences”, *49th Hawaii International Conference on System Sciences (HICSS)*, (2016), 5288–5297.

[32] Mueller, R.M., and S. Huettemann, “Extracting Causal Claims from Information Systems Papers with Natural Language Processing for Theory Ontology Learning”, *Proceedings of the 51st Hawaii International Conference on System Sciences*, (2018), 5295–5304.

[33] Pakray, P., and A. Gelbukh, “An open-domain cause-effect relation detection from paired nominals”, *Mexican International Conference on Artificial Intelligence*, Springer (2014), 263–271.

[34] Pascanu, R., T. Mikolov, and Y. Bengio, “On the difficulty of training Recurrent Neural Networks”, *arXiv:1211.5063 [cs]*, 2012.

[35] Pennington, J., R. Socher, and C. Manning, “GloVe: Global vectors for word representation”, *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, (2014), 1532–1543.

[36] Reimers, N., and I. Gurevych, “Optimal Hyperparameters for Deep LSTM-Networks for Sequence Labeling Tasks”, *arXiv preprint arXiv:1707.06799*, 2017.

[37] Snoek, J., H. Larochelle, and R.P. Adams, “Practical bayesian optimization of machine learning algorithms”, *Advances in neural information processing systems*, (2012), 2951–2959.

[38] Tjong Kim Sang, E.F., and S. Buchholz, “Introduction to the CoNLL-2000 shared task: Chunking”, *Proceedings of the 2nd workshop on Learning language in logic and the 4th conference on Computational natural language learning-Volume 7*, Association for Computational Linguistics (2000), 127–132.

[39] Tjong Kim Sang, E.F., and F. De Meulder, “Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition”, *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003-Volume 4*, Association for Computational Linguistics (2003), 142–147.

[40] Wang, M., and C.D. Manning, “Effect of non-linear deep architecture in sequence labeling”, *Proceedings of the Sixth International Joint Conference on Natural Language Processing*, (2013), 1285–1291.

[41] Yang, Z., R. Salakhutdinov, and W. Cohen, “Multi-task cross-lingual sequence tagging from scratch”, *arXiv preprint arXiv:1603.06270*, 2016.

[42] Zaremba, W., I. Sutskever, and O. Vinyals, “Recurrent neural network regularization”, *arXiv preprint arXiv:1409.2329*, 2014.

[43] Zilly, J.G., R.K. Srivastava, J. Koutník, and J. Schmidhuber, “Recurrent Highway Networks”, *International Conference on Machine Learning*, (2017), 4189–4198.