Causal Expression Identification

CHRISTOPH SCHATZ, Graz University of Technology, Austria
MUHAMED RAMIĆ, Graz University of Technology, Austria

This research project had the main goal of comparing different algorithms like Random Forest, LSTM, Decision Trees and BERT-based models in the context of Natural Language Processing. The specific task was to find the purpose of annotations within text, and different methods were used in order to analyse the results and come to conclusions which are discussed in the report. The main findings and conclusions were that not only models that have the specific purpose of processing text perform good, but that also other models which were applied on the dataset had a good performance, both when it comes to the accuracy and the runtime. The research project did not include the surrounding of every annotation, which is something that could be included in further research in order to improve the accuracy of the models and provide us with a better conclusion.

ACM Reference Format:
Christoph Schatz and Muhamed Ramić. 2023. Causal Expression Identification. In . ACM, New York, NY, USA, 8 pages. https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

As natural language processing was slowly shifting from just detection to understanding the text content, the following questions emerged. Can NLP methods actually understand text? And if they can, can they perform reasoning and inference tasks?

Causality span detection and identification is a task where the events are extracted from text. These events contain three spans that represent cause, signal and effect.

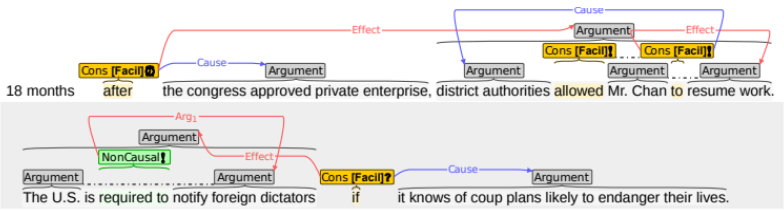


Fig. 1. Examples of annotated sentences with causality in them [Dunietz et al. 2017]

Causality extraction is a key to understanding relations in text. Agent and effect detection methods have a use in political, social and economic sciences, in cases such as improving trading strategies. These methods can be use to improve inference, classification and summarization tasks.

Manuscript submitted to ACM

The remaining content of this paper is structures as follows. In section 2 we review some related work for causality identification. In section 3 we talk about data preparation and the methods that we use to tackle the task. In section 4 we show how we evaluate our methods, after which we discuss our findings in section5. This paper ends with conclusion and suggestions for future work in section 6.

1.1 Problem definition

Is it possible to predict the span label based on its content without the content of the whole causal sentence?

2 RELATED WORK

When it comes to causal span identification the common approach is to detect if the pair of events has a cause-effect relation. Such a method is CATENA [Mirza and Tonelli 2016] which for a sentence with causality in the form of $< e_{cause}, e_{signal}, e_{effect} >$. It then forms pairs $< e_{cause}, e_{signal} >$, $e_{signal}, e_{effect} >$ and $< e_{cause}, e_{effect} >$, and then tries to identify which of these pairs is made of cause and effect spans. It is based on SVM using l_2 loss.

In [Rehbein and Ruppenhofer 2020] the method is based on BERT [Devlin et al. 2018] sequence classifier and they try to classify individual causal spans. On the other hand, LTRC for Causal News Corpus [Adibhatla and Shrivastava 2022] see this task as a token classification in a named entity recognition task. They also use BERT based model adjusted for token classification tasks.

However there are approaches like [Chen et al. 2022] where they make prediction on sentences where they try to find start and end positions of the causality spans. Output format of this method is $< s_{cause}, e_{cause} >, < s_{effect}, e_{effect} >, < s_{signal}, e_{signal} >$. Their detection method is based on BERT but they do not use just one architecture. They construct a Signal Classifier that first detects if the sentence contains causality and helps by providing additional information. Here they also augment data by paraphrasing sentences using a pre-trained model.

3 METHODOLOGY

3.1 Dataset

We used the BECAUSE Dataset [Dunietz et al. 2017] to train our model. It is a set of fully annotated texts that attempts to capture the enormous variety of constructions used to express cause and effect. We used our own parser for these .ann files to extract exactly the parts of these files that we are interested in. In these file we can find information about causal statements and events between them.

These tokens can be classified into six classes: Argument, Consequence, Motivation, NonCausal, Note and Purpose. There is class imbalance in the dataset since there are more Argument tokens then of any other class, but since this also represents the imbalance that exists in the normal text this should not create any issues.

3.2 Preprocessing

3.2.1 Data loading. From all the annotations that the dataset provides we were only interested in the entries beginning with either "T" or "E" which represent text and event respectively. Besides text in the tokens and the token information

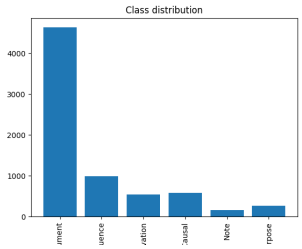


Fig. 2. Class distribution

int the events there are addition information the entries such as begging and start position of the token.

We then use these samples to put together a data frame that contains the sentences and their labels. Here we were thinking about also using more information like the other parts of the sentences and feeding them into the methods. However, we decided to only train the models on the text sequences since our goal was to create a tool that classifies tokens, where the additional information might not be available.

3.2.2 Text normalisation. Three steps of text cleaning were performed. We removed contractions from text. By contractions we mean words or combinations of words where some letters are removed, eg. **don't** is a contraction of **do not**. The second pre-processing step in text normalisation that we performed was to remove any special characters. Lastly, we transformed all text to lower-case.

```
you're looking to go from colonel to general, and it's a 2 percent selection rate
you are looking to go from colonel to general and it is a 2 percent selection rate
```

Fig. 3. Removing contractions

```
``Guys, you are going to get shot coming off the helicopter,'' he told them
guys you are going to get shot coming off the helicopter he told them
```

Fig. 4. Removing special characters

```
Secretary of State Condoleezza Rice telephoned Mr. Abbas
secretary of state condoleezza rice telephoned mr abbas
```

Fig. 5. Transformation to lowercase

3.2.3 Text tokenization. Since the ML models use numerical values for computation we need to transform text to numerical representations. We have done experiments with two tokenizers. First we tried using **tf.keras.preprocessing.text.Tokenizer** that performs a vectorization of said texts by constructing a dictionary and then representing a word by its index in the dictionary.

We also used BERT based tokenizers, specificaly **BertTokenizer** and **DistilBertTokenizer**, that are based on Google's WordPiece embeddings. These tokenizers work by using a whole word as a token or splitting a word into several distinct pieces. As an example the word **unable** would be split into 2 tokens by splitting it to **un** and **#able**.

3.2.4 Label encoding. Class labels are in textual form and therefore also need to be transformed to numerical representations. For this we encode each into an integer with **sklearn.preprocessing.LabelEncoder**. For LSTM models, we also did one-hot-encoding on the labels.

3.3 Model selection

Let us first look into baseline models. Our first baseline model is the **rounded mean** which predicts the mean of all training labels rounded to the closest integer which is in our case 1, meaning that it will always predict the Consequence. Our second baseline model is the **median** which always predicts 0, which predicts the class Argument. The third baseline model is the **discrete uniform** and similar to that our last baseline is the **chainsaw** model which predicts by repeating the following sequence 0, 1, 2, 3, 4, 5.

Fast and simple models that performed well were the Decision Tree Classifier, as well as the Random Forest Classifier. These models do not take much time to train and have an efficient performance when it comes to runtime.

We also try a simple bidirectional LSTM model with the architecture described in the table below. This model is trained with Adam optimizer with a default learning rate of $1e-3$ and categorical crossentropy as the loss function.

Table 1. Model summary of the LSTM model

Layer	Output Shape	Param #
Embedding layer	(None, 67, 100)	666400
Bidirectional LSTM	(None, 67, 128)	84480
Bidirectional LSTM	(None, 64)	41216
Dense layer	(None, 64)	4160
Dense layer	(None, 6)	390

Then we use the pre-trained BERT model with the sequence classification head on top. This model has an additional dense layer on top of it and is fitted for the classification task. We train this model with the Adam optimizer with PolinomialDecay learning scheduler with initial learning rate $5e-5$. This model uses sparse categorical crossentropy as a loss function.

We added the DistilBERT to our project to see if the smaller network can have the same performance as BERT since it has 40% less parameters.

4 EVALUATION

To train and test models data was split by ratio 80:20. For LSTM and BERT based models 20% of the training data was used as validation data.

As evaluation metrics we use precision, recall, f1-score and accuracy. For the models except BERT we will only mention test accuracy and average f1-score. These results are shown in table 2.

Table 2. Performance results

	f1-score	Accuracy
Rounded mean	0.03	0.14
Median	0.50	0.64
Discrete uniform	0.19	0.15
Chainsaw	0.22	0.18
Decision tree	0.84	0.85
Random forest	0.84	0.85
LSTM	0.85	0.86
BERT	**0.87**	**0.87**
DistilBERT	0.86	**0.87**

Here we can see the machine learning models achieve significantly higher accuracy and f1-score then the naive baseline models. We notice that BERT has the highest results considering both metrics. We provided a full BERT classification report in table 3. DistilBERT proved that it can keep up with the BERT model regardless to the parameter number reduction.

Table 3. BERT classification report

	precision	recall	f1-score	support
Argument	1.00	0.99	1.00	914
Consequence	0.73	0.68	0.70	197
Motivation	0.58	0.47	0.52	113
NonCausal	0.57	0.91	0.70	117
Note	0.17	0.03	0.05	35
Purpose	0.89	0.83	0.86	60
accuracy			0.87	1436
macro avg.	0.66	0.65	0.64	1436
weighted avg.	0.87	0.87	0.87	1436

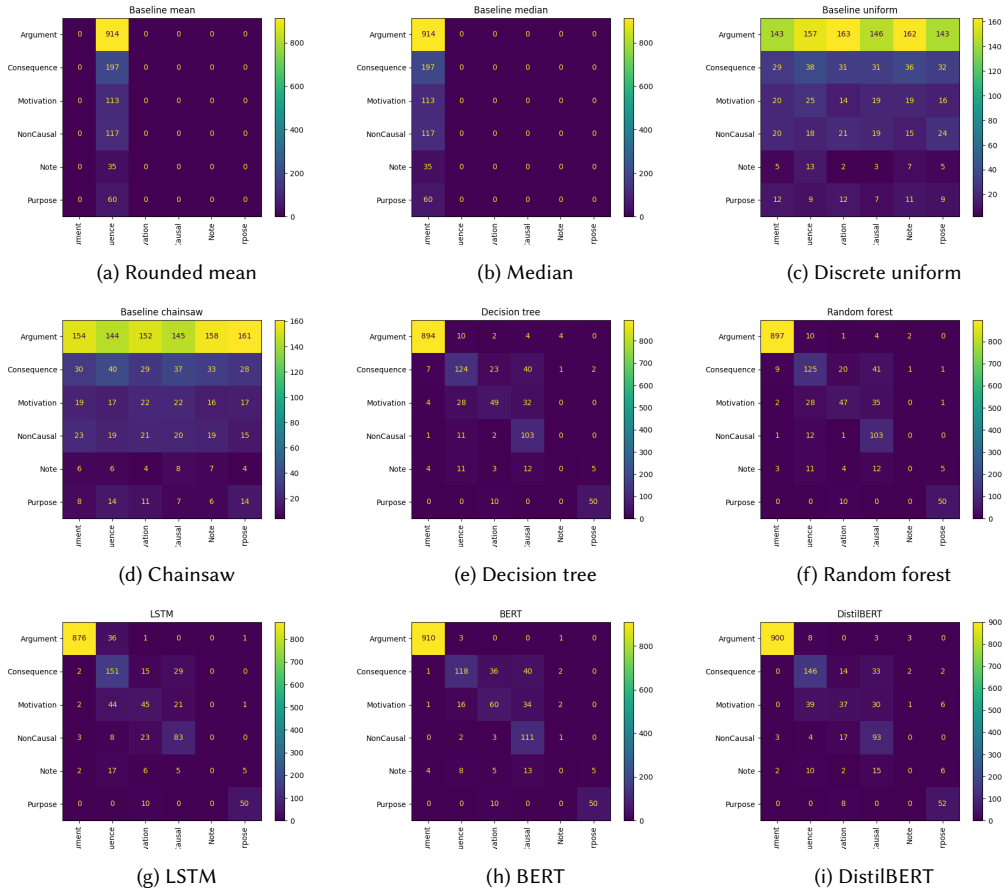
To visually represent the performance of the models we used confusion matrices. See figure 6.

5 DISCUSSION

We can see that even if there is a class imbalance in the dataset, our models still have a decent accuracy. All our models, forest classifier, decision tree classifier, LSTM, BERT for sequence classification and DistilBERT for sequence classification, have similar performances.

We also notice that the models do not overfit since the test accuracy is lower by 2-5% then the train accuracy. By taking a look into the training history we can see there that the models do not overfit.

While inspecting token texts we wanted to know if there were some tokens with the same text and if they could be removed. We have found 827 duplicates in the dataset. However we found that that even if the entries have the same textual content they can have different role in different events. In the table 4 we have a few examples.

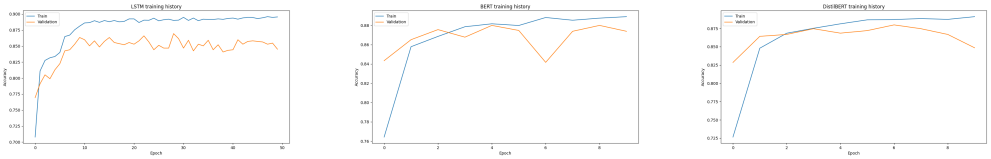


(a) Rounded mean (b) Median (c) Discrete uniform

(d) Chainsaw (e) Decision tree (f) Random forest

(g) LSTM (h) BERT (i) DistilBERT

Fig. 6. Confussion matrices



(a) LSTM (b) BERT (c) DistilBERT

Fig. 7. Training history

This hints that the role of the token is highly related to the context within the sentence. This tells us that if we consider using a whole sentence or pairs of tokens as inputs in the classification method.

When performing tokenization, we tried using vectorization method and BERT based tokenizers. Here we notice that change in tokenization method does not influence model performance.

Table 4. DistilBERT classification report

Text token	Role 1	Role 2
to	Purpose	Note
led to	Consequence	Motivation
after	NonCausal	Consequence
allowing to	Consequence	Motivation

6 CONCLUSION

Based on our finding we can conclude that language models such as LSTM and BERT have decent performance in the token identification task. We found that other machine learning methods, such decision trees and random forest, can do these predictions faster without loosing in performance.

For our future work we would consider other approaches that take surrounding text of the token into consideration. Words have different meanings depending on the position in the sentence and in what context they are used. We that this also applies to causal spans within a sentence.

We believe that we should not restrict ourselves to using a single architecture for predictions but diving the detection into smaller tasks and specializing models for each task. Such task would be causality presence detection in sentence and using that information for the final predictions.

REFERENCES

Hiranmai Sri Adibhatla and Manish Shrivastava. 2022. LTRC@ Causal News Corpus 2022: Extracting and identifying causal elements using adapters. In *Proceedings of the 5th Workshop on Challenges and Applications of Automated Extraction of Socio-political Events from Text (CASE)*. 50–55.

Xingran Chen, Ge Zhang, Adam Nik, Mingyu Li, and Jie Fu. 2022. 1Cademy@ Causal News Corpus 2022: Enhance Causal Span Detection via Beam-Search-based Position Selector. *arXiv preprint arXiv:2210.17157* (2022).

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

Jesse Dunietz, Lori Levin, and Jaime G Carbonell. 2017. The BECauSE corpus 2.0: Annotating causality and overlapping relations. In *Proceedings of the 11th Linguistic Annotation Workshop*. 95–104.

Paramita Mirza and Sara Tonelli. 2016. Catena: Causal and temporal relation extraction from natural language texts. In *The 26th international conference on computational linguistics*. ACL, 64–75.

Ines Rehbein and Josef Ruppenhofer. 2020. A new resource for German causal language. In *Proceedings of the Twelfth Language Resources and Evaluation Conference*. 5968–5977.

A APPENDIX

A.1 Work distribution

We believe that both authors made equal contributions which maybe cannot be seen from the code or report since both have been through many modifications.

Table 5. Work done by Muhamed

Tasks performed
text cleaning
label encoding
constructed baseline models
implemented evaluation of models (classification summary and confusion matrix)
implemented decision tree training
implemented random forest training
implemented DistilBERT training

Table 6. Work done by Cristoph

Tasks performed
data loading
tokenizations (vectorization and BertTokenizer)
data spliting
implemented LSTM training
implemented BERT training
hyperparameter tuning