Abstract

In this paper, we explore strategies to detect and evaluate counterfactual sentences. Since causal insight is an inherent characteristic of a counterfactual, is it possible to use this information in order to locate antecedent and consequent fragments in counterfactual statements? We thus propose to compare and evaluate models to correctly identify and chunk counterfactual sentences. In our experiments, we attempt to answer the following questions: First, can a learned model discern counterfactual statements reasonably well? Second, is it possible to clearly identify antecedent and consequent parts of counterfactual sentences?

1 Introduction

A counterfactual can be defined as something that is contrary to the truth or that did not actually occur. It refers to an event that did not or cannot happen, as well as the possible consequences if it had happened. In the sentence "If dogs had no ears, they could not hear", the statement "if dogs had no ears" is an example of a counterfactual because dogs do have ears. Task 5 of SemEval-2020 (Yang et al., 2020) focuses on identifying these specific sentence types amongst sentences delivering close semantic similarities. This implies understanding and disambiguating the causal link between two sentence fragments.

We approached this task as an opportunity to test the effectiveness of grammatical disambiguation against baseline systems. This paper describes a parallel approach in deriving meaning from text to leverage the influence of context and relevance of structure in recognizing counterfactual statements. Specifically, we will study how many expressions of such statements a mapping of grammatical types can cover before falling short to high-performing models, most notably BERT (Devlin et al., 2018), which performed well on both tasks with an F1 score of 85.00% on Task 1 and 83.90% on Task 2.

2 Related Work

Although the task of detecting counterfactuals is relatively new, (Son et al., 2017) proposes using modal logic to form rule-based determination methods from social media posts. These methods are supplemented to a statistical classifier (Linear SVM) that is retrained to tackle more challenging counterfactual forms.

Previous work on causal identification by (Levin and Hovav, 1994) studied the contribution of verbs in the determination of causal relations. Specifically, they found similarities in meaning from different structures, some involving transitive verbs and others involving intransitive verbs. This is to say that a causal meaning can lie in the verb or extend to an object (another word or word type). By analyzing closely different formulations, they concluded that the same reasoning can be carved out from different grammatical structures.

As for causality relation extraction, different deep learning systems built on the success of neural-based models have been proposed. The linguistically informed CNN model (Dasgupta et al., 2018) leverages the use of word embeddings and other linguistic features to detect causal patterns and outperforms rule-based classification. (Liang et al., 2019) introduces a multi-level causal detector that makes use of

This work is licensed under a Creative Commons Attribution 4.0 International Licence. Licence details: http://creativecommons.org/licenses/by/4.0/
multi-head self-attention to capture semantic features at word level and infer causality at segment level. This engineered system has rivaled state-of-the-art models in terms of performance and thorough understanding of complex semantic information such as discourse relations and transitivity rules. Finally, (Li et al., 2019) presents a self-attentive BiLSTM-CRF model that makes use of transfer learning to overcome the problem of data insufficiency and extract causal relations in natural language text. This solution transfers a trained embedding from a large corpus and uses the causality tagging scheme to identify dependencies between cause and effect. Experimental results prove the effectiveness of this model, but its major limitation is the insufficiency of high-quality annotated data to learn from.

3 Dataset

The datasets used are those provided by the shared task organisers. The data is described in (Yang et al., 2020). As per official task instructions, no additional data was used.

4 Task 1: Classification Problem

Task 1 is a classification problem which aims at recognising text sections as either counterfactual or not. Counterfactual sections are labeled 1 and non-counterfactual sections are labeled 0.

The proposed baseline by the task organisers is a SVM classifier. We evaluate two different approaches and compare them to the baseline: a classical approach using popular machine learning classifiers with additional linguistic features, a combination that has shown to perform well on information retrieval tasks relying on text understanding (Dai and Callan, 2019), and a deep learning approach using a BERT linear classifier. The objective of this competitive approach is to determine whether a supplement of linguistic features can be sufficient to correctly recognize counterfactual structures as opposed to running heavier models integrating broader contextual knowledge.

4.1 Proposed Method

We approach the task as a disambiguation problem. A manual linguistic analysis of the training dataset shows that verbs are key elements for the detection of counterfactuals. (Son et al., 2017) identify 7 structures related to counterfactuals, all depending on a verb feature.

In order to test these structures, we create a grammar allowing us to generate combinations of tokens based on their grammatical category (i.e., verb, pronoun) as well as their linguistic features (i.e., tense).

We test the grammar on the training set and obtain an output of 2833 sentences recognised as counterfactuals (on a total of 13000). However, only 593 of them are labeled “1” in the training set. A manual analysis of the output reveals that many statements are recognised as counterfactuals because of ambiguities.

We create a few additional grammars to disambiguate some of the cases. They mostly concern verb tenses. For example, we tell the grammar that if present participles are located behind the structure have been, it becomes a gerund. We also categorize could, would and should as modals so they wouldn’t be identified as verbs in the preterit tense. We also proceed to disambiguate the grammatical category of words such as this, that, these and those. Finally, we create a grammar to disambiguate the category of wish and wishes.

However, some cases resist disambiguation, especially the sentences containing could/would/should have structures, which are very common in English and in many cases do not imply a counterfactual. Moreover, the context of these structures can vary which makes them even more complex to disambiguate. Counterfactuals based on the combination of two specific structures, such as conjunctives, also prove problematic. The two structures can be joined by elements of various nature, i.e., verb phrase (If I could go back and do it over, I would be more negative) or noun phrase (If I was the Fed, I’d be happy with this rather than unhappy). These elements are hard to identify because of the large combinatorics of their possible aggregation. However, by ignoring these elements, we increase the chances of ambiguities.

Eventually, the ambiguity between adjectives and past participles can raise many mistakes in the identification of counterfactuals when using grammars. Adjectives constructed with the suffix -ed are also recognized as verbs in preterit tense or past participles by the grammar (i.e., he wasn’t prepared).
We choose to try different machine learning methods to evaluate if existing latent variables can possibly lie in the syntax and semantic of counterfactual statements.

For the classical approach, we supplement classifiers with word vectorisation and grammatical features. The vectorisation methods used are Bag Of Words (BOW), TF-IDF, Word2Vec and BERT vectors. For TF-IDF and Word2Vec, we experiment with and without stop word removal. Grammatical features follow the recommendation of (Son et al., 2017) relating to counterfactual forms. The classifiers are SVM, Multinomial Naive Bayes (NB), Decision Tree (DT), Random Forest (RF) and Multi-Layer Perceptron (MLP). In addition to grammatical features extraction, we perform a series of operations like removing numbers, punctuation and stop words, replacing negative contraction verbs with their complete forms (i.e., won’t), splitting compound nouns (i.e., state-of-the-art) and transforming text to lowercase.

We obtain a concise features list for the classical classifiers. In order to remove stop words, we use the NLTK\(^1\) stop words set and increase it with contraction patterns like ’re or ’m. We apply this custom stop word list to the Word2Vec vectorisation. Stemming and lemmatisation based on POS tags are used for the BOW and TF-IDF embeddings. We also replace white spaces with a single space. Finally, for BOW and TF-IDF, we remove words with frequency less than 5. This effectively decreases the dimensions of BOW and TF-IDF vectors. We also add a cross validation method with a stratified fold of 3 to the above-mentioned hybrid models. For the BERT embeddings applied in this phase, we refrain from applying the cleaning operations to maintain more semantic freedom.

For the deep learning approach, we use the same BERT model that achieved the best result in a similar text classification task in the NLP4IF-2019 Shared Task (San Martino et al., 2019). We use the sentence tokenisation version of the uncased model, with 12 Transformer layers and 110 million parameters. This model has proven to pay special attention to adjectives and verbs, two grammatical structures that can play a role in identifying counterfactual statements (Levin and Hovav, 1994).

### 4.2 Results

The results are based on the official training set provided by the organisers. The dataset contains 13000 lines split in the following: 40% training sample, 30% validation sample and 30% test sample.

For the classical approach, the best 6 results have been retained out of all the possible combinations of classifiers and word embeddings, and measured against the baseline provided for the task and the deep learning model. The +/- scores are the averaged measures from the 3-fold cross-validation results.

| Classifier     | Processing          | Precision | Recall | F1 score |
|----------------|---------------------|-----------|--------|----------|
| Baseline SVM   | TF-IDF              | 72.72 %   | 8.73 % | 15.59 %  |
| MLP            | BERT Sentence Version | 80 +/- 1 % | 80 +/- 1 % | 80 +/- 1 % |
| SVM            | BERT Sentence Version | 79 +/- 0 % | 81 +/- 1 % | 80 +/- 0 % |
| SVM            | TF-IDF with Stop Words | 86 +/- 0 % | 69 +/- 1 % | 74 +/- 1 % |
| DT             | TF-IDF with Stop Words | 69 +/- 1 % | 70 +/- 1 % | 70 +/- 1 % |
| NB             | BERT Sentence Version | 64 +/- 1 % | 75 +/- 2 % | 66 +/- 1 % |
| RF             | TF-IDF with Stop Words | 91 +/- 1 % | 61 +/- 1 % | 65 +/- 1 % |
| BERT Linear    | BERT Uncased        | 87 +/- 1 % | 83 +/- 1 % | 86 +/- 1 % |

Table 1: Task 1 - Benchmark Tests

The blind test set consists of 7000 unlabeled lines of text. Our best model, the uncased Linear BERT, achieves an F1 score of 85.00%, a Precision score of 84.20% and a Recall score of 85.90% on this set. This experiment showcases the difficulty of disambiguation when syntactic and semantic features are very close in both alternatives being classified.

### 5 Task 2: Sequence Extraction

Task 2 is a sequence extraction problem. The text sections are similar to the ones labeled “1” in the Task 1 dataset. The purpose of this task is to extract, in a text section identified as counterfactual, the sub-strings

---

\(^1\)https://github.com/nltk/nltk
identifying the antecedent and the consequent elements (Yang et al., 2020). We use the following split sampling for the training data (3551 individuals): 1740 sentences for the training sample, 746 sentences for the validation sample and 1065 sentences for the test sample.

5.1 Proposed Method

Our approach for this task is also comparative and evaluates two Deep Learning models. It consists in testing whether we can supply enough linguistic knowledge to determine all causal forms in counterfactual statements to challenge the breadth and depth of our BERT-based model from Task 1.

Since this task is a sequence classification problem, our first system takes advantage of the BERT model used in task 1, which is here deployed as a Sequence Extractor. To that end, we use a wrapper to initialise and fine-tune the base uncased BERT model and add a Multi-Layer Perceptron classifier layer on top as demonstrated in (Dai and Callan, 2019). For parameter tuning, we use a Random Search to facilitate multi-parameter testing. The optimal parameter selection is represented below:

Number of MLP Layers: 3
Number of MLP Hidden Neurons: 500
Max Sequence Length: 173
Number of Epochs: 3
Learning Rate: 5e−5
Batch Size: 16
Gradient Accumulation Steps: 2

The second system is inspired by discriminative models and Conditional Random Fields (CRF) in particular. We model the task as a Named Entity tagging problem, each token being assigned a specific set of features and evaluated against the prediction of its target feature. The target features are defined as C when the token belongs to a sub-string Consequent and A when the token belongs to a sub-string Antecedent. Tokens belonging to neither are marked I.

We enhance the discriminative properties of the CRF by working on some additional layers and modeling a Deep Learning CRF. We experiment with linguistic embeddings, features and regularisation methods as enhancements for the final BI-LSTM CRF Neural Model. The parameters for this system are detailed below in sections 5.1.1 to 5.1.4.

5.1.1 Embeddings

We evaluate several embedding methods. The first is Pre-trained Word Embeddings. This widely used technique helps tackle the problem of generalising unseen words, since word embeddings are good at capturing general syntactic as well as semantic properties of words (Reimers, 2017). For our experiments, we focus on two different approaches: the GloVe embeddings trained on Common Crawl (about 840 billion tokens) and the FastText approach trained on Common Crawl (600 billion tokens) which also extracts subword information. We also experiment with Character Level Embeddings (C2IDX). We select the recommended approach of (Ma and Hovy, 2016) using Convolutional Neural Networks (CNN) based off the evaluation of this method and the BI-LSTM approach for character representation from (Reimers, 2017). The CNN approach only takes into account trigrams and is position-independent (i.e., the network will not be able to distinguish between trigrams occurring at the beginning, middle, or end of the word), which itself can be crucial information in our task. The BI-LSTM, on the other hand, takes all word characters into account and is position-aware. The results of (Reimers, 2017) prove that while intuitively the BI-LSTM should be the better approach, both perform equally well and the CNN technique is computationally more efficient, which makes it our favored choice for this task. Finally, we also consider Stacked Embeddings (i.e., a combination of existing embedding techniques designed to act in succession in a single pipeline to generate a refined embedding for our input). We use the Flair library (version 0.4.5) to achieve this. Our embeddings pipeline is composed of: a GloVe model, a Flair-forward model, a Flair-backward model, and a BERT embedding model. By targeting these different models, we cover different aspects of semantic representation for our input: the GloVe module

---

2https://github.com/charles9n/bert-sklearn
3https://github.com/flairNLP/flair
targets word representation, the Flair modules are for character contextualisation, and the BERT layer for sentence-level information extraction.

5.1.2 Features
Features are declared in the CRF layer. For POS tags, we use the Stanford\(^4\) Part-Of-Speech Tagger to determine the role of each word in the discourse. For Chunking, sentences are chunked into Consequent and Antecedent segments. Each segment is tokenised into words and each word is tagged with its segment label (i.e., A for Antecedent and C for Consequent). In order to generate BERT features, we use the BERT-as-service\(^5\) library (version 1.10.0) as a sentence-encoder to map variable-length sentences to fixed-length feature embeddings. Lastly, we also experiment with Syntactic Grammars (SG).

Since our task requires labeling tokens, we use a special tagging scheme to identify segment boundaries and add structure to the custom labels introduced during the chunking process. As the results from (Reimers, 2017) show, the BIO tagging scheme performs consistently well for this type of task and is adopted for our features.

5.1.3 Regularisation
Given the danger of over-parameterisation that neural networks present, we introduce some regularisation techniques for a more generalised performance of our model. We perform a K-Fold Cross Validation with K = 3. The data is not shuffled before splitting into batches. We also add Dropout. Results from (Reimers, 2017) show that variational dropout performs best when it comes to BI-LSTM networks. Furthermore, it can be shown in (Cheng et al., 2017) that relatively smaller dropout tends to yield better results for LSTM networks. For our experiments, we implement a variational dropout on all layers with the fraction \(p\) of dropped values from the set \(\{0.1, 0.3, 0.5\}\). The value of \(p = 0.1\) performs best after empirical testing and is retained for our final round of benchmark tests. We couple our system with an Elasticnet method (i.e., a linear regression model with combined L1 and L2 priors). The tested combinations of regularisation methods are the following:

- Cross Validation (A)
- Cross Validation + Dropout (B)
- Cross Validation + Dropout + Elasticnet (C)

The reference labels A, B, C are used in Table 3.

5.1.4 Hyperparameters
We also evaluate the effects of different hyperparameters on the model performance.

**Character Embedding Dimension:** We select the CNN approach recommended for deep LSTM networks (Reimers, 2017) and test configurations on the embedding size. Larger embedding dimensions cause redundancy while capturing character-level information and perform worse than smaller sizes.

**Word Embedding Dimension:** For word-level embeddings, we also test different configurations and their impact on the model. Since we also make use of pre-trained embedding models, we observe that setting the word dimension to a size close to the usual pre-trained word vectors (100 - 300) yields the best performing models.

**BI-LSTM Layer Dimension:** We evaluate 1, 2 and 3 stacked BI-LSTM layers.

**Recurrent Units:** The number of recurrent units is selected from a range of sizes with \(32 \leq u \leq 100\). The forward and reverse running LSTM networks have the same number of recurrent units. Multiple BI-LSTM layers also have the same number of recurrent units.

**Optimizer:** We experiment with two commonly selected optimizers, namely stochastic gradient descent (SGD) and Adamax. SGD is highly sensitive to the learning rate, meaning choosing a too high rate can cause the system to diverge in terms of the objective function, whereas a too low rate results in a slow learning process. Adamax is selected to bypass the shortcomings of SGD.

---

\(^4\)https://nlp.stanford.edu/software/ tagger.shtml
\(^5\)https://github.com/hanxiao/bert-as-service
**Learning Rate:** We tune the learning rate by hand and observe in many instances that it fails to converge to a minimum. We select a range of viable rates to test for best performance.

**Weight Decay (L2 Penalty):** We couple weight decay with our optimizer to add regularisation and evaluate the best value for our model.

| Parameter                        | Tested Configurations | Best Configuration |
|----------------------------------|-----------------------|--------------------|
| Character Embedding Dimension    | 25, 50, 75, 100       | 25                 |
| Word Embedding Dimension         | 50, 200, 300, 500     | 300                |
| BI-LSTM Layer Dimension          | 1, 2, 3               | 3                  |
| Recurrent Units                  | 32, 50, 64, 75, 100   | 100                |
| Optimizer                        | Adamax, SGD           | Adamax             |
| Learning Rate                    | 0.0015, 0.002, 0.015, 0.02 | 0.0015             |
| Weight Decay (L2 Penalty)        | 15e−10, 15e−12, 15e−14 | 15e−12             |

Table 2: Task 2 - Hyperparameter Evaluation

The best configuration for each parameter is considered the optimal one for our use case and our models are tuned to those values for the benchmark tests.

### 5.2 Results

Since we are working with a complex model with high tuning capabilities, the most relevant results are retained and presented in the following table instead of exhaustively listing all combinations of experiments. Here again the +/- measures are averaged from the 3-fold cross validation scores. Below are the most competitive results gathered from our models on the training dataset. The BERT-MLP model results are included as the last row for comparison.

| Model         | Embedding                     | Features      | Regularisation | Precision   | Recall   | F1 score |
|---------------|-------------------------------|---------------|----------------|-------------|---------|----------|
| BI-LSTM CRF   | FastText + C2IDX              | POS + CHUNKS  | B              | 77 +/- 1 %  | 76 +/- 2 % | 77 +/- 2 % |
| BI-LSTM CRF   | Stacked + C2IDX               | POS + CHUNKS + SG | C       | 76 +/- 1 %  | 75 +/- 1 % | 77 +/- 2 % |
| BI-LSTM CRF   | GloVe + C2IDX                 | POS + CHUNKS  | C              | 78 +/- 1 %  | 78 +/- 1 % | 79 +/- 1 % |
| BI-LSTM CRF   | GloVe + C2IDX                 | POS + CHUNKS + SG | A       | 76 +/- 1 %  | 75 +/- 1 % | 77 +/- 1 % |
| BI-LSTM CRF   | C2IDX                         | POS + CHUNKS + BERT + SG | B | 76 +/- 1 %  | 76 +/- 1 % | 77 +/- 1 % |
| BI-LSTM CRF   | Stacked + C2IDX               | POS + CHUNKS  | C              | 76 +/- 1 %  | 76 +/- 1 % | 76 +/- 1 % |
| BERT-MLP      | BERT                          | CHUNKS + BERT | A              | 84 +/- 1 %  | 85 +/- 1 % | 84 +/- 1 % |

Table 3: Task 2 - Benchmark Tests

From the table, we can observe that the best performing BI-LSTM CRF solution is the GloVe word-embedding, character-level embedding model augmented with chunking and POS tags. However, a big problem with this system is its reliance on context-free word embeddings to perform well, which can be a weakness in the domain of textual understanding. Our BERT-MLP model, on the other hand, is capable of high-performance textual recognition using contextual data. The BERT-MLP results show a significant jump in performance after incorporating context learning of in-domain neighboring words. On the blind test dataset, our best model achieves a Precision score of **82.30%**, a Recall score of **88.80%**, an F1 score of **83.90%** and an Exact Match of **25.90%**.

### 6 Conclusion

In this paper, we presented our experiments for identifying counterfactual statements. Modeling features derived from a linguistic analysis such as specific grammar structures for counterfactual statements and coupling them with established machine learning or deep learning models did not perform as well as context-learning models, as our hybrid BERT-MLP solution outperforms even complex combinations of deep learners and displays a better level of understanding and handling challenging counterfactual forms. Future work could explore the impact of graph knowledge in accommodating systems and rendering them more perceptive of implicit and ambiguous textual meanings.
References

Nabiha Ashgar. 2016. Automatic Extraction of Causal Relations from Natural Language Texts: A Comprehensive Survey. Arxiv 2016.

Cheng, Gaofeng and Peddinti, Vijayaditya and Povey, Daniel and Manohar, Vimal and Khudanpur, Sanjeev and Yan, Yonghong. 2017. An Exploration of Dropout with LSTMs. Interspeech 2017.

Giovanni Da San Martino, Alberto Barron-Cedeno, Preslav Nakov. 2019. Findings of the NLP4IF-2019 Shared Task on Fine-Grained Propaganda Detection. Arxiv 2019.

Zhuyun Dai and Jamie Callan. 2019. Deeper Text Understanding for IR with Contextual Neural Language Modeling. Arxiv 2019.

Dasgupta, Tirthankar and Saha, Rupsa and Dey, Lipika and Naskar, Abir. 2018. Automatic Extraction of Causal Relations from Text using Linguistically Informed Deep Neural Networks. Proceedings of the 19th Annual SIGdial Meeting on Discourse and Dialogue, pages 306316.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. CoRR, abs/1810.04805.

Jesse Dunietz. 2018. Annotating and Automatically Tagging Constructions of Causal Language. Carnegie Mellon University.

Jesse Dunietz, Lori Levin, Jaime Carbonell. 2017. The BECauSE Corpus 2.0: Annotating Causality and Overlapping Relations. Proceedings of the 11th Linguistic Annotation Workshop, ACL Anthology 2017.

Beth Levin and Malka R. Hovav. 1994. A Preliminary Analysis of Causative Verbs in English. Lingua 92, 35-77.

Li, Zhaoning and Li, Qi and Zou, Xiaotian and Ren, Jiangtao. 2019. Causality Extraction based on Self-Attentive BiLSTM-CRF with Transferred Embeddings. Arxiv 2019.

Liang, Shining and Zuo, Wanli and Shi, Zhenkun and Wang, Sen. 2019. A Multi-level Neural Network for Implicit Causality Detection in Web Texts. Arxiv 2019.

Xuezhe Ma and Eduard H. Hovy. 2016. End-to-end Sequence Labeling via Bi-directional LSTM-CNNs-CRF. Arxiv 2016.

Erika Nazaruka. 2019. Identification of Causal Dependencies by using Natural Language Processing: A Survey. ENASE 2019.

Nils Reimers and Iryna Gurevych. 2017. Optimal Hyperparameters for Deep LSTM-Networks for Sequence Labeling Tasks. Arxiv 2017.

Son, Youngseo and Buffone, Anneke and Raso, Joe and Larche, Allegra and Janocko, Anthony and Zembroski, Kevin and Schwartz, H. and Ungar, Lyle. 2017. Recognizing Counterfactual Thinking in Social Media Texts. Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers), pages 654658.

Yang, Xiaoyu and Obadinma, Stephen and Zhao, Huasha and Zhang, Qiong and Matwin, Stan and Zhu, Xiaodan. 2020. SemEval-2020 Task 5: Counterfactual Recognition. Proceedings of the 14th International Workshop on Semantic Evaluation (SemEval-2020).