IITK-RSA at SemEval-2020 Task 5: Detecting Counterfactuals

Anirudh Anil Ojha* **Rohin Garg*** **Shashank Gupta*** **Ashutosh Modi**
Indian Institute of Technology Kanpur (IITK)
{aaojha, sronin, gshasha}@iitk.ac.in
ashutoshm@cse.iitk.ac.in

Abstract

This paper describes our efforts in tackling Task 5 (Yang et al., 2020) of SemEval-2020. The task involved detecting a class of textual expressions known as counterfactuals and separating them into their constituent elements. Counterfactual statements describe events that have not or could not have occurred and the possible implications of such events. While counterfactual reasoning is natural for humans, understanding these expressions is difficult for artificial agents due to a variety of linguistic subtleties. Our final submitted approaches were an ensemble of various fine-tuned transformer-based and CNN-based models for the first subtask and a transformer model with dependency tree information for the second subtask. We ranked 4^{th} and 9^{th} in the overall leaderboard. We also explored various other approaches that involved classical methods, other neural architectures and incorporation of different linguistic features.

1 Introduction

Counterfactuals (Starr, 2019) are statements that describe events that did not (or cannot) happen, which implies that their truth cannot be empirically verified. They may be divided into two major categories (Iatridou, 2000). If possible outcomes (had the event in question happened) are described as well, the statement is called a *counterfactual conditional*. Specifically, it contains a logical segment equivalent to "if A, then B" where A did not occur. The "if A" part is called the *antecedent* and the "then B" is called the *consequent*. An example of this is "If we had changed the battery on time, the car wouldn't have broken down." A statement of the second type is called a *counterfactual wish*. They also describe alternate realities, but do not talk about outcomes and do not display a conditional structure. They are called so because they often display 'wish' vocabulary e.g. "I wish there was an easy way to categorise counterfactuals!" Here, the segment referring to a fictional event is termed as the antecedent, and there is no consequent. While there are other ways to categorise these logical forms formally, we believe that this system covers most of the examples seen in our datasets and provides clear intuition about their general structure. Readers interested in a more rigorous treatment may refer to Lewis (2013).

The ability to reason via counterfactuals is considered a higher form of intellect, as one must extrapolate the truth using causal relationships, similar instances and a commonsense understanding of the world. Hence, for artificially intelligent agents, counterfactual reasoning is critical for generalisation and explainability. In terms of more concrete applications, it has been hypothesised to be important for natural language understanding, goal determination (a major challenge in modern reinforcement learning), error analysis and others (Ginsberg, 1986). Recently, Mothilal et al. (2020) have worked on systems that generate counterfactual explanations for why classifiers behave the way they do on given examples.

We tried a variety of approaches on this task, including classical machine learning, CNN-based models, BiLSTM-based models and transformer-based models. We found that the transformer-based neural models (introduced by Vaswani et al. (2017)) consistently outperformed its competitors. We obtained an F1 score

* Authors equally contributed to this work.

Sentence	Gold Label
I wish it had more.	True
Even if the next government awards the NHS an annual funding increase of 1.5 per cent over inflation, which is no simpler than giving everyone good enough medical care whenever they need it, the service would have still had to make annual efficiency savings of 2-3 per cent a year.	True
If Mr McCain is in charge, his record of bipartisan outreach will stand him in good stead; Mr Obama will be able to rely on solid majorities in Congress.	False
If your device is locked and no one can discern the security code, you may not need a patent on it.	False

Table 1: Samples from the training set of the first subtask

Sentence	Antecedent	Consequent
The GOP's malignant amnesia regarding the economy would be hilarious were it not for the wreckage they caused.	were it not for the wreckage they caused	The GOP's malignant amnesia regarding the economy would be hilarious
"I wish there was no limit on the number of groups you could join because there are so many good ones," Larsen said.	I wish there was no limit on the number of groups you could join	
Duda said that if Netanyahu had said what was originally reported, "Israel would not be a good place to meet in spite of the previous arrangements".	Duda said that if Netanyahu had said what was originally reported	Israel would not be a good place to meet in spite of the previous arrangements

Table 2: Samples from the training set of the second subtask

of 89.3% on the test data of the first subtask and 48.3% on the second subtask, ranking our system 4th and 9th respectively. Our implementation is available via Github.[1]

2 Brief Task and Dataset Description

The overall task comprises two subtasks:

- Detecting counterfactual statements
- Parsing such statements into antecedent and consequent (if present)

The first may be modelled as a binary classification problem and the second as a sequence-to-sequence labelling problem. The training set was provided by the task organisers. The examples were segments from news articles in English and were quite varied in length, from around 5 words to around 400 words. There were 13000 examples for the first subtask, with a 88:12 negative-positive split, and 3500 for the second subtask. As the datasets were imbalanced, our evaluation metrics were precision, recall, F1-score and exact match (exclusively for the second subtask). The test set had 7000 sentences for the first subtask and 1950 for the second. Some samples from the training set are given in Table 1 and 2. For more details, please refer to Yang et al. (2020).

3 Previous and Related Work

Research on counterfactuals is a relatively new area in the NLP community, and consequently, limited research literature exists. The closest work is by Son et al. (2017), that aimed to detect counterfactual elements in tweets. They identify a set of rules to filter such examples based on grammatical form

[1] https://github.com/gargrohin/counterfactuals-nlp

(sentences containing "wish", "should have", etc.) and train an SVM to classify filtered sentences on the basis of POS tags and n-gram features. While their method works well on tweets, we found that these rules had limited coverage and were not effective in filtering long sentences like those in our dataset. The filtering also removes several examples, which makes it harder to train further classifiers in an already data-constrained setting.

Iatridou (2000) provides a thorough treatment of counterfactuals in terms of grammar. However, the features and verb forms they identify and use are quite complex, like fake tenses. Such morphology is currently undetectable by even state-of-the-art libraries like spaCy[2], which makes this system hard to implement with computational models without appropriate data. Nevertheless, the paper does allow one to construct certain useful grammatical rules for inference.

Counterfactual implication and causality are closely linked (Morgan and Winship, 2015). Counterfactual conditionals often talk about direct results of the presence (or absence) of a particular factor e.g. "If the gas hadn't run out, we would have been able to cook dinner." Even when they do not directly encode such relationships - one may consider sentences like "Even if you were the last person on the planet, I wouldn't go out with you!" - they often imply the existence of other factors which drive the relationship. Thus, from a linguistic perspective, the vocabulary and structure seen in causal sentences and that seen in counterfactual sentences have a high degree of overlap. Hence, we believe systems to detect and parse causal relationships would perform well on this task as well. With respect to the first subtask, fine-tuning neural models using information-rich word embeddings seems to form the state-of-the-art (Kyriakakis et al., 2019) and forms the backbone of our submitted approach as well. The second subtask bears a resemblance to relation-entity extraction (of which cause-effect may be considered a specific case), which has also been recently dominated by neural models (Li and Tian, 2020; Soares et al., 2019). We found span-based (Joshi et al., 2020) and discourse parsing-based (Son et al., 2018) approaches particularly interesting with regards to our task.

4 Our Approaches

4.1 Subtask 1

4.1.1 Classical machine learning

We tried SVMs and gradient-boosted random forests on linguistic feature-based (POS tags, verb tense and aspect information, etc.) representations of the samples since the size of the dataset was small. The baseline provided by the organisers was an SVM classifier on TFIDF features with stop words removed. We felt that this would result in the removal and down-weighting of terms critical for identifying counterfactuality like 'should have'. Indeed, we found that performance improved remarkably by discarding TF-IDF features and allowing stop words. This implies that the way that closed class words are used in a sample is particularly informative for this task. Even using simple unigram word features provided decent results, which suggests that there are specific words whose presence in itself is useful for detecting counterfactuality. Thus we used these features in our neural model as well (see § 4.1.3 c)). Even though the results improved by the progressive addition of linguistic features, the F1 score eventually stagnated at around 65%.

We then added a regex-based filtering step, where we divided the samples based on certain typical counterfactual forms (inspired by the work of Iatridou (2000)) and classified them separately. We used four primary forms - sentences containing an 'if' then a modal verb, a modal verb and then an 'if', the word 'wish' and the remaining. In the first two, we wished to capture verb forms like 'would have' and 'could have', which we found to have a strong correlation with counterfactuality. We found the modal-if form the hardest to classify, with an F1 of around 52%. The most notable improvements were on the 'wish' form where the F1 was around 90%, which hints that counterfactual wishes are the easiest to identify. Classifiers for the other two categories obtained an F1 of around 62-65%, similar to what we obtained before filtering.

[2]https://spacy.io

4.1.2 CNN with GloVE and Word2Vec

Inspired by (Kim, 2014), we replaced the words with pretrained word embeddings and applied a CNN-based classification head. For each sentence, the words were tokenised and replaced with their embedding. Thus, a matrix was formed with dimension (embedding length $\times$ maximum sentence length). We experimented with multiple kernel sizes and with static and non-static versions of word-embeddings, as in the paper. We finally settled on 100 kernels of sizes 3, 4 and 5 each. There was a max-pooling layer before the fully-connected layer. We abandoned the non-static embeddings layer because training the embeddings resulted in overfitting. GloVe (Pennington et al., 2014) embeddings performed 3-4% better F1-wise than Word2Vec (Mikolov et al., 2013) embeddings as seen in Table 3.

4.1.3 Transformer-based models

Introduced by Vaswani et al. (2017), transformer-based models have recently reported state-of-the-art (SOTA) results on a variety of NLP tasks, and we found the same to be true for our task as well. Since our dataset was small, we experimented with transfer learning on three state-of-the-art variants - BERT (Devlin et al., 2018), RoBERTa (Liu et al., 2019) and XLNet (Yang et al., 2019). We tried the following modifications (exact values reported in Table 3 and § 5):

a) **Fully-connected head:** This involved combining the word-level embeddings generated by the transformer into a sentence-level embedding and then applying a fully-connected layer for classification. We experimented with various ways of producing the sentence embedding, such as taking the mean, max-pooling (taking the maximum value over each dimension) and selecting particular embeddings. We found that taking last word embedding (the [CLS] token in BERT and RoBERTa) gave us the best results. We applied a tanh activation between the sentence embedding and the linear layer. 'Large' variants performed around 5% better on F1 than their 'base' counterparts. Despite the simplicity of this approach, it was second only to the ensembles (see section § 4.1.5). We also tried to fine-tune these models on subsets of the data after applying a filtering step as in § 4.1.1. However, the F1 score stagnated around 80%, probably because of insufficient data.

b) **CNN head:** Instead of obtaining a sentence embedding, we directly applied a CNN head on the word-embedding matrix from the transformer. This is similar to what we did in the case of vanilla CNN model (§ 4.1.2), and we decided to apply the same final CNN head that we used in that section. While this did provide about 3-4% better F1 than the 'base' versions with the fully-connected heads, it was still prone to overfitting and could not outperform the 'large' versions.

c) **With linguistic features:** Following up on our observations in § 4.1.1, we appended linguistic features like word n-grams and POS n-grams to the transformer embeddings before passing it through a fully connected layer. We experimented with unigrams, bigrams and trigrams of words and POS tags, and eventually settled on the top 1000 trigram features of each. This addition stabilised training and improved performance on 'base' models by 3-4% F1, but could not outperform our fine-tuned 'large' models.

4.1.4 Discourse parsing-based approaches

We took inspiration from Son et al. (2018) and parsed the samples into discourse arguments with PDTB-style methods. We were motivated by two key observations - there was a high degree of overlap between relevant discourse arguments and sections encoding counterfactuality (antecedent and consequent), and in longer examples, only a small part was counterfactual. Thus, we hoped that this parsing could help the system better filter out antecedent and consequent from larger examples and focus on understanding their specific structure.

We first used spaCy dependency parser on the samples, and then used this information along with the presence of known discourse connectives like 'even though' and 'because' to obtain discourse arguments. After parsing, we tried two architectures. In the first, we obtained word-level GloVe embeddings as in § 4.1.2 and applied a hierarchical BiLSTM network. The first BiLSTM network was applied at the word level, over each discourse argument, thus giving us a discourse-level embedding in the form of the final hidden state. These discourse-level embeddings were taken sequentially and fed into another BiLSTM. The final hidden state thus generated was fed into a fully-connected layer.

In the second approach, we used a RoBERTa model to obtain the discourse-level embedding, similar to how we obtained the sentence-level embedding in § 4.1.3 a). Each discourse argument was limited to 32 words, with a maximum of 8 discourse arguments. These discourse embeddings were recurrently fed into a BiLSTM as before. We initialised the discourse-level BiLSTM with the sentence-level embedding of the samples obtained as in § 4.1.3 a). Hence, the hidden state dimension of the BiLSTM was the same as that of the transformer embedding. While we were able to train this with 'base' transformers, the 'large' variants took too long.

Despite our intuition, GloVe embeddings and the hierarchical BiLSTM performed slightly worse than the Glove + CNN model in § 4.1.2. In the case of transformer embeddings with a BiLSTM, we did see an improvement over the results in § 4.1.3 a) on 'base' transformers, but not more than what we had already obtained before in § 4.1.3 b) and § 4.1.3 c), suggesting that this also suffer from overfitting due to a large number of parameters. The results are recorded in Table 3.

4.1.5 Ensembles

We took ensembles of the different models we had trained so far. This was based on our observation that different networks failed on different examples, and thus a simple voting algorithm would smooth out inconsistencies. We also observed that RoBERTa models usually had higher recall while XLNet models had higher precision, and thus could complement each other. We could only experiment with hard voting ensembles due to a lack of time - it would be interesting to explore how other methods would perform. This yielded the best results overall. We finally submitted two ensembles. The first ensemble model is a combination of (individually fine-tuned) an XLNet large, an XLNet base and a RoBERTa large. The second ensemble model is a combination of a RoBERTa large, an XLNet large, an XLNet base, a BERT base with CNN head, a BERT large and a CNN model with GloVe embeddings. The second model gave the best results on the final test set.

4.2 Subtask 2

4.2.1 Sequence Prediction

We tried two approaches while treating this as a sequence prediction task:

a) **Token classification using transformers:** Motivated from the performance of transformers in Subtask 1, we experimented with transformer-based models on this subtask as well. We approached the problem like a sequence prediction task similar to named-entity recognition. We tokenised the string into words and gave appropriate BIO tags to the antecedent and consequent separately for training. We fine-tuned separate models for detecting antecedent and consequent, as we noticed that the two sections often overlapped (sometimes completely). As RoBERTa gave us the F1-best performance in the first subtask, we experimented with BERT and RoBERTa.

b) **End to End Sequence Labelling:** We also modelled the task as an end-to-end sequence labelling problem, where each word would be labelled as an antecedent or a consequent. We used the algorithm described by (Ma and Hovy, 2016) in which word representations and character-level representations are used as inputs to a Bi-directional LSTM model. The two hidden states for each word are concatenated to form the final output. Finally, a Conditional Random Field (CRF) (Lafferty et al., 2001) was used to define a probabilistic model for sequence labelling.

4.2.2 Using dependency tree

An exploratory analysis of the dataset showed that around 60% of the sentences in the dataset had the word 'if'. Out of these, 90% were in antecedents, thus showing a correlation between the usage of the word 'if' and part of the sentence being an antecedent. Given this observation, we tried to use this marker to deal with the antecedents separately. We separated these sentences and analysed the dependency tree using a dependency visualiser to find a pattern to the antecedents which started with the word 'if'. In most cases where it was used, the word 'if' was a subordinating conjunction that depended on a noun/verb from an independent clause in the dependency tree. The last word of the antecedent in nearly all the cases was the node of that branch of the tree rooted in that word, i.e. the last word up to which that branch of the

tree extended. The antecedents extracted from the dependency tree were closest to ground truth, which is why we used them in our final model for the relevant sentences.

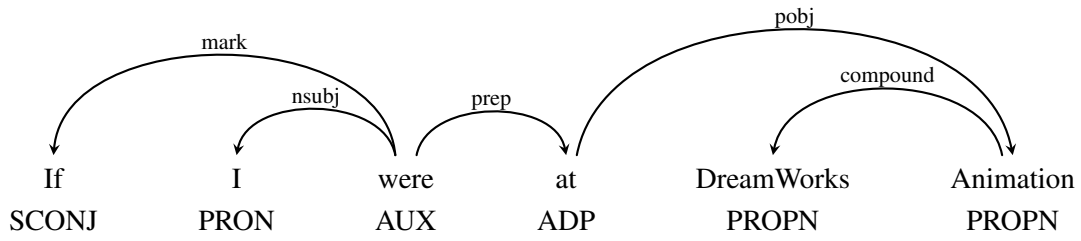

Figure 1: Antecedent extends from 'if' to the other end of its root branch

5 Implementation Details and Results

All the scores reported in the paper are on the validation set created by us by randomly splitting the dataset in a ratio of 75%-25%. We used the same train-validation split across models to make sure the results were comparable. Please note we only report results of the key experiments - more details can be found at the GitHub repository. The results are reported in Table 3 and Table 4.

For classical methods, we used the scikit-learn toolbox[3] and spaCy to generate linguistic features. A linear SVM with the regularisation parameter set as 1.0 yielded the best results. The linear SVM has been known to perform the best in sentence classification tasks (Joachims, 1998).

The implementation for the CNN based models in § 4.1.2 was kept similar to the original implementation by (Kim, 2014). The convolutional kernel of sizes {3,4,5} was used with the activation function ReLU after each convolution and a max-pool layer after the final convolution. A fully-connected layer with output being the number of classes (2 in our case) completed the model. A dropout (Srivastava et al., 2014) layer was added before the final layer. An AdamW (Loshchilov and Hutter, 2017) optimiser is used with an initial learning rate of 1×10^{-3}. As for the transformer models with a CNN head, the word-embedding matrix simply replaced the GloVe embeddings that were used in a stand-alone CNN model, with the rest of the model remaining the same.

The transformer-based models described in § 4.1.3 were implemented with the help of the Transformers library in PyTorch by HuggingFace[4]. We used an AdamW optimiser for all of our models with an initial learning rate of 1×10^{-5} over all the parameters and an epsilon of 1×10^{-8}. We trained the models for a total of 20 epochs and saved the model after each epoch if it provided the best F1 score on the validation set so far. We found that performance peaked around 16 epochs. We decayed the learning rate linearly to 0 over the training period. The choice of these involved quite a bit of fine-tuning as we found that results were sensitive to these hyper-parameters. The algorithm would not converge outside a specific range, and even within this range, the F1 could vary by 4-5 points upon small changes.

As we mentioned earlier, the 'large' versions of the transformers outperformed their 'base' versions by 1-2% in the case of BERT and 4-5% in the case of the others. As shown, adding the CNN layer did actually improve performance in the relatively less-pretrained BERT, but slightly worsened performance on the sophisticated RoBERTa (large). Similarly, our experiments with linguistic features (LF) improved performance on base-sized transformers, but could not beat the performance of a fine-tuned large RoBERTa. The results for the discourse parsing-based (DP) method are also shown here - as mentioned earlier, they perform marginally (0.3%-0.5% F1) better than the models of § 4.1.3 b) and c).

For subtask 2, the transformer-based model in § 4.2.2 was implemented with the help of the Simple Transformers library[5] in PyTorch. We tried BERT and RoBERTa as the models and used an AdamW

[3]https://scikit-learn.org/stable/index.html
[4]https://github.com/huggingface/transformers
[5]https://github.com/ThilinaRajapakse/simpletransformers

Model	Precision	Recall	F1	Model	Precision	Recall	F1
Baseline (given)	73.46	7.86	14.20	SVM+Trigram	64.64	55.89	59.95
SVM+Tf-Idf	77.45	34.49	47.73	SVM+Trigram+POS	73.60	58.37	65.10
CNN (word2vec)	64.41	75.78	69.63	CNN (GloVe)	69.18	75.50	72.20
RoBERTa (large)	88.98	88.98	**88.98**	XLNet (large)	**92.96**	83.75	88.12
BERT (large)	89.25	82.40	85.67	BERT (b, CNN)	89.05	85.12	87.04
RoBERTa (l, CNN)	89.82	82.64	86.08	RoBERTa (b, LF)	89.02	85.40	87.20
RoBERTa (l, LF)	86.01	**91.46**	88.65	SVM on 'wish'	**96.55**	84.84	**90.32**
DP (Transformer)	90.08	85.12	87.54	DP (Word2Vec)	69.83	71.36	70.59
Vote (first)*	85.96	92.83	89.27	Vote (second)*	90.04	**93.24**	**91.18**

Table 3: Major results for subtask 1. A $*$ indicates those submitted for the competition.

Model	Precision	Recall	F1	Exact match
RoBERTa (large)	45.80	48.10	44.90	**0.0349**
RoBERTa + DepParse*	**48.30**	**51.80**	**47.10**	0.0318

Table 4: Major test results for subtask 2. A $*$ indicates those submitted for the competition.

optimiser with an initial learning rate of 4×10^{-5} over all the parameters and an epsilon of 1×10^{-8}, and were trained for 5 epochs. We used spaCy for generating and visualising the dependency tree in § 4.2.2.

6 Qualitative Analysis of Results

Here we present a brief analysis of the classification results for subtask 1. We try to identify some of the major reasons why our models perform the way they do, mostly by analysing the training data, and the false predictions made by our best models.

Identifying whether a sentence is a counterfactual or not an easy task even for humans - even we were unable to agree on whether a certain sentence was a counterfactual amongst ourselves in many cases. However, one pattern is quite evident - many counterfactuals are of the form "if... else...". Hence, the model learns to give a high weight to the word embeddings corresponding to these words. Unfortunately, this means that a sentence having a structure similar to this one is often falsely classified as a counterfactual. For example, the following statements were falsely classified as counterfactuals: *"Using simple math, you'd think that if you had worked 33 years and chose to work one more year, then you'd boost your benefits by about 1/33, or 3%.", "Even if the Prime Minister's deal had been passed on Tuesday, there is a huge raft of legislation the Government would still need to pass"*.

But the major issue in all the models was a low recall, i.e. a high number of false negatives. A low recall value was the major reason why we kept the voting threshold low (less than 50%) for a statement to be classified as a counterfactual in our ensemble models,. We believe that this was due to the models not being able to capture certain types of counterfactuals that do not follow a specific sentence structure. For example, consider the sentence *' "Can you imagine if I said the things she said?" Mr. Trump told the crowd.'* This is a counterfactual, but our model was not able to focus on the word *imagine* that makes it one. This shows that we either need a lot more data for different types of counterfactuals or a more objective set of rules that a model can be forced to learn. Sometimes a counterfactual is present as a part of a quote in a much more complex sentence such as *' "So even if he can spend infinite money, it doesn't follow that he can use that money to condemn property," Somin said, adding that the administration would then be limited to building the predicted wall on land the federal government already own.'* and *"It's become fashionable to tell a disability story in a hopeful arc, where the heroine may have moments of discouragement or fear, but comes out into full life at the end - into mainstream schools, love and romance, full participation in the social world, and these stories have become so pervasive that if they were to spread to aliens they'd find them familiar."*. It is possible that the complexity and length of a sentence and presence of many clauses that are clearly not counterfactual overshadow the one clause that was a

counterfactual and brought down the final score.

For subtask 2, the major challenge was that the antecedent and consequent which contributed to counterfactuality were often only parts of the longer sentence and hence it was difficult to conclude where the contributing bits ended. This problem was easier to solve in some cases where there was a pattern to the sentences as in the case of sentences containing 'if'. There we saw that the dependency tree could be relied upon as there was a clear structure to the start of the antecedent, which depended on the main noun/verb of the clause.

7 Conclusion and Future Work

The identification and analysis of counterfactual statements are essential for any computational natural language task, be it knowledge extraction or question answering. We attempt to identify such statements from a diverse corpus and produce competitive results using several SOTA language models. We modify some of the models and also use linguistic features to make our models task-specific, and with an ensemble of these models, we were able to rank 4th on the leaderboard in the first subtask. There is still scope for further improvements, such as hardcoding more specific rules in the model related to various kinds of counterfactuals.

We also attempt to analyse the counterfactual statements by identifying the antecedent and the consequent. We model the problem as a sequence prediction problem and try out multiple transformer-based models as well as probabilistic ones. We ranked 9th on the leaderboard for this subtask. There are some other methods that one can experiment with for this task, such as separating the sentence into clauses and focusing on individual clauses, span-based models, analysing patterns involving other parts-of-speech, and graph-based networks that can leverage the power of the dependency tree correlation that we identified. We leave these methods for future work.

References

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*.

Matthew L Ginsberg. 1986. Counterfactuals. *Artificial intelligence*, 30(1):35–79.

Sabine Iatridou. 2000. The grammatical ingredients of counterfactuality. *Linguistic inquiry*, 31(2):231–270.

Thorsten Joachims. 1998. Text categorization with support vector machines: Learning with many relevant features. In *European conference on machine learning*, pages 137–142. Springer.

Mandar Joshi, Danqi Chen, Yinhan Liu, Daniel S Weld, Luke Zettlemoyer, and Omer Levy. 2020. Spanbert: Improving pre-training by representing and predicting spans. *Transactions of the Association for Computational Linguistics*, 8:64–77.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751, Doha, Qatar, October. Association for Computational Linguistics.

Manolis Kyriakakis, Ion Androutsopoulos, Artur Saudabayev, et al. 2019. Transfer learning for causal sentence detection. *arXiv preprint arXiv:1906.07544*.

John Lafferty, Andrew McCallum, and Fernando CN Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data.

David Lewis. 2013. *Counterfactuals*. John Wiley & Sons.

Cheng Li and Ye Tian. 2020. Downstream model design of pre-trained language model for relation extraction task.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Ilya Loshchilov and Frank Hutter. 2017. Decoupled weight decay regularization.

Xuezhe Ma and Eduard Hovy. 2016. End-to-end sequence labeling via bi-directional lstm-cnns-crf. *arXiv preprint arXiv:1603.01354*.

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.

Stephen L Morgan and Christopher Winship. 2015. *Counterfactuals and causal inference*. Cambridge University Press.

Ramaravind K. Mothilal, Amit Sharma, and Chenhao Tan. 2020. Explaining machine learning classifiers through diverse counterfactual examples. In *ACM Conference on Fairness, Accountability, and Transparency*, January.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Livio Baldini Soares, Nicholas FitzGerald, Jeffrey Ling, and Tom Kwiatkowski. 2019. Matching the blanks: Distributional similarity for relation learning. *arXiv preprint arXiv:1906.03158*.

Youngseo Son, Anneke Buffone, Joe Raso, Allegra Larche, Anthony Janocko, Kevin Zembroski, H Andrew Schwartz, and Lyle Ungar. 2017. Recognizing counterfactual thinking in social media texts. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 654–658.

Youngseo Son, Nipun Bayas, and H. Andrew Schwartz. 2018. Causal explanation analysis on social media.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958.

William Starr. 2019. Counterfactuals. In Edward N. Zalta, editor, *The Stanford Encyclopedia of Philosophy*. Metaphysics Research Lab, Stanford University, fall 2019 edition.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. In *Advances in neural information processing systems*, pages 5754–5764.

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