Winter 12-2022

BEVERS: A General, Simple, and Performant Framework for Automatic Fact Verification

Mitchell DeHaven
University of Nebraska-Lincoln, mitchell.dehaven@outlook.com

BEVERS: A GENERAL, SIMPLE, AND PERFORMANT FRAMEWORK FOR AUTOMATIC FACT VERIFICATION

by

Mitchell DeHaven

A THESIS

Presented to the Faculty of

The Graduate College at the University of Nebraska

In Partial Fulfilment of Requirements

For the Degree of Master of Science

Major: Computer Science

Under the Supervision of Stephen Donald Scott

Lincoln, Nebraska

December, 2022

BEVERS: A GENERAL, SIMPLE, AND PERFORMANT FRAMEWORK FOR AUTOMATIC FACT VERIFICATION

Mitchell DeHaven, M.S.

University of Nebraska, 2022

Adviser: Stephen Donald Scott

Fact verification has become an important process, primarily done manually by humans, to verify the authenticity of claims and statements made online. Increasingly, social media companies have utilized human effort to debunk false claims on their platforms, opting to either tag the content as misleading or false, or removing it entirely to combat misinformation on their sites. In tandem, the field of automatic fact verification has become a subject of focus among the natural language processing (NLP) community, spawning new datasets and research. The most popular dataset is the Fact Extraction and VERification (FEVER) dataset.

In this thesis an end-to-end fact verification system is built and trained based on the FEVER dataset. Our system utilizes traditional document retrieval and pretrained transformer models to form predictions. Our system does not deviate significantly from a standard approach, however we thoroughly examine design decisions and data representations to further improve the model. We suspect other results in the literature under optimized their approaches, given that we perform much better with similar system design. Finally, our system is compared against other systems through the FEVER blind test dataset and sets a new state of the art for the task while utilizing relatively simple approaches and smaller models than other systems. Our system attains the highest FEVER score on the task, while scoring second on pure accuracy, and finally our evidence retrieval system achieves the highest recall among reported results of other systems.

DEDICATION

To my parents, Alan and Jana DeHaven.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

In recent years, misinformation online has become a significant problem for companies to contend with. Recently, accusations of large scale misinformation campaigns perpetrated by foreign countries to interfere with elections have become increasingly common, particularly in the US [22, 17]. More recently, with the recent COVID-19 pandemic, a proliferation of misinformation has been spread regarding the virus, masks, and vaccines throughout social media and alternative news sources [42, 45]. These types of misinformation pose serious dangers, as they can intentionally deceive individuals to sway political elections, or be used to undermine health officials goals during health emergencies like we saw with the COVID-19 pandemic. Large tech companies have been heavily scrutinized over the past 5 years for the amount of misinformation hosted and shared on their platforms. Particularly, Facebook has been criticized for it's perceived lack of action to prevent misinformation spread on its platform [4].

In response, many of these companies have begun hiring or outsourcing fact checkers, whose primary jobs are to identify and respond to misinformation being shared on the platform, by checking sources and verifying claims (particularly of the the political or medical variety) [11, 40]. Obviously, doing this kind of fact checking for large social media websites like Facebook and Twitter or large video hosting websites like YouTube is costly due to the large amounts of content that need to be verified. Currently, Facebook employs a variety of fact checkers around the world and has contributed $84 million to fact checking programs since 2016 [12]. This effort, by and large, is done primarily manually by these fact checking organizations, who search the web for relevant information regarding a claim and to determine its validity, according to Snopes [47]. Thus, introducing methods to reduce the amount of manual labor required to verify claims would allow fact checkers to verify more content or could allow these companies to output as much fact verification with fewer employees.

The natural language processing community has recently been giving more attention to this field of automatic fact verifications. Recent datasets, such as PolitiFact [54], Fake News Challenge [1], and FEVER [52] have allowed researchers to explore approaches and improve methods for automatic fact verification. FEVER, released in 2018, is the most popular among these datasets, consisting of 185,445 claims with associated labels and evidence. The evidence for this dataset is formed from a 2017 dump of Wikipedia articles' introduction passages. Thus, to verify the claims in the dataset, one must provide the correct evidence with the correct label to get credit for properly verifying the claim.

The dataset was released approximately around the time that large pretrained transformer [53] models, such as BERT [9], thus initial approaches utilized large recurrent neural network models to form predictions. Of course, once BERT was applied to the problem, as with other areas in NLP, it outperformed previously used methods by a large margin. More recently, approaches to the dataset have utilized graph neural networks (GNNs) to approach the problem, suggesting that utilizing these models with pretrained transformers outperforms just using transformers alone. The best systems in the past 2 years have utilized much larger transformer models, including 900 million and 3 billion parameters models [23, 49], to push the state of the art higher.

In this work, I aim to test two theories. The first theory is that GNNs with transformers provide significant improvement over only using the transformer model for the

FEVER task. The GNN works that show improvements all compare against a couple of early transformer baselines and there could potentially be problems with the baselines. Although the GNN-based systems' performance have been passed recently by purely transformer based models, one would fairly point out that many of these subsequent works have utilized significantly larger transformer models and therefore aren't comparable. Thus, in this work I use the same-sized transformer model their works used. The second theory is that to achieve state-of-the-art performance on the task, significantly larger transformer models are not needed. As mentioned previously, among the top performing systems, the models utilized range from $3\times$ to $10\times$ larger than what I utilize in this thesis and the GNN systems utilized. As is often the case, increasing the size of the pretrained language model generally improves performance, but are such large models necessary?

1.1 Contributions

To test these theories, for this thesis an end-to-end FEVER system is built, including the document retrieval, sentence selection, and final claim verification. We call this system BEVERS (Baseline fact Extraction and VERification System). From building the system, the following contributions are made:

- Introduce a new very strong baseline utilizing standard approaches in NLP that achieves the #1 FEVER score among all systems on the FEVER blind test set.
- Show that with equivalent sized models, GNN based systems utilizing transformer-based systems do not outperform our transformer based system and in fact are worse when comparing FEVER score, contrary to what has previously been reported in the research.
- Show that in comparable evaluation setups, our system outperforms systems that

use transformer models with 3x and 10x as many parameters as the models our system uses.

1.2 Thesis Outline

This section outlines the remainder of the thesis. Chapter 2 discusses background details regarding deep learning in the natural language processing field as it relates to this thesis. Chapter 2 also discusses the details of the FEVER dataset. Chapter 3 reviews research related to this thesis topic, particularly research regarding other systems built for the FEVER dataset. Chaper 4 gives a full system description of the FEVER system built for this thesis, as well as experimental setup. Chapter 5 discusses results and impact of various design decisions. Finally, in Chapter 6 we discuss conclusions and potential future work.

Chapter 2

Background Information

In this chapter additional background information about the various approaches used are discussed in more detail. First, a task description for the FEVER dataset is necessary for understanding why various approaches were used. Next, we give a brief discussion on the information retrieval methods used for gathering relevant evidence for claims within the dataset. Finally, we provide background information for deep learning approaches in natural language processing, including discussions on transformer models, which have not only become a popular deep learning architecture for natural language processing tasks, but for computer vision and audio tasks as well.

2.1 FEVER Dataset

The Fact Extraction and VERification (FEVER) dataset [52] is largest and most popular dataset for automatic fact verification currently in use by the NLP research community today. The task as defined by the dataset is twofold. First, given a sentence expressing a claim, retrieve all the relevant documents for the claim from a corpus. Within the FEVER dataset, the corpus is comprised of the introduction section of all English Wikipedia articles, resulting in a corpus of 5,000,000 documents. From the Wikipedia articles the evidence retrieved is made up of sentences from the Wikipedia page article. Thus, for a given claim, the correct evidence will be made up of a tuple of the correct article and correct sentence number, (e.g. {"Barack Obama", 12}). From a claim and retrieved evidence, the system then must predict whether the claim is a REFUTES, NOT ENOUGH INFO, or SUPPORTS label. The REFUTES label is used when the correct evidence for a claim refutes the claim. The SUPPORT label is used when the correct evidence for a claim supports the claim. The NOT ENOUGH INFO label is used when for a given claim there is no evidence within the corpus that can either refute or support the claim.

For the FEVER dataset, there are several metrics used for measuring performance. The first metric is evidence *F1 Score*, used for measuring the systems retrieval performance. F1 score is a commonly used metric in machine learning for measuring binary classification performance, combining *recall* and *precision*. For binary classification, recall measures the proportion of true positive samples that are correctly classified as positive. The exact formula for calculating recall is given below. Precision is also a common used metric for binary classification and measures the proportion of predicted positive samples that are true positive examples. Its exact formulation is also given below. Recall and precision, however, when taken in isolation, can give misleading results. For example, if a classifier predicts all samples as positive, it will trivially have a 100 % recall, since all true positives were labelled as true. It is worth noting that in a circumstance precision score would likely fall since if all samples are predicted as true, the proportion of those that are true positive samples will be low. Similarly, if a model predicts only a single sample as positive and it happens to be a true positive, the model will have 100% precision, regardless of how many true positive samples it did not correctly label. Again, it is worth noting that recall in this circumstance would fall as well, since many true positive samples are not correctly labelled. Given this relationship between recall and precision, F1 measure avoids these issues by taking the harmonic mean of the two scores, thus simply getting a higher recall while ignoring precision will get a poor F1 score and vis versa. The formulas [32] for computing recall, precision, and F1 score are:

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}}$$

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}}$$

$$\text{F1 score} = \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

There are two important caveats about how recall is computed for FEVER. The first relates to the fact that for some claims in the dataset multiple pieces of evidence are required in order to make a correct prediction. For example, for the claim "Barack Obama was born on Christmas Day", the evidence "Barack Hussein Obama II (born August 4, 1961) is an American politician who served as the 44th president of the United States" and "Christmas is an annual festival commemorating the birth of Jesus Christ, observed primarily on December 25" would be required in order to get credit for retrieval. Thus, only having a strict subset of the required evidence for a claim grants no credit for recall. Additionally, at most 5 pieces of evidence can be returned for scoring. Thus, another common metric used for the task is recall @ 5 (which proportion of samples in which all required evidence can be found within the top 5 retrieved evidence).

The second metric used for measuring performance is label accuracy. This is fairly straightforward, measuring the proportion of predicted label classes are in fact the correct label classes. The final and primary metric for measuring performance is referred to as FEVER score. FEVER score requires for a prediction to be correct that not only must the predicted label be correct, but that all of the relevant evidence for the claim was retrieved as well. Thus, even if the model predicts the correct label but returns only 2 of the 3 required pieces in the top 5 predicted evidence, then FEVER score counts this as a miss.

2.2 Information Retrieval

Information retrieval (IR) refers to the process of retrieving relevant documents for a given query. Among traditional IR approaches for text based retrieval, one of the most popular and oldest techniques (dating back to 1972 [25]) is term frequency–inverse document frequency (also called tf-idf). Tf-idf is a document representation model for representing documents that attempts to consider both frequency of words within documents as well as the infrequency of the words within the entire corpus. The formula for the tf-idf value for a particular term t given a particular document d is given below [32]. The $\text{tf}(t, d)$ term represents the frequency, or raw count, of term t within document d, N represents the total number of documents in the corpus, and finally df_t represents the number of documents the term t is found in.

$$\text{idf}(t) = \ln\left(\frac{N}{\text{df}_t}\right)$$

$$\text{tf-idf}(t, d) = \text{tf}(t, d) \cdot \text{idf}(t)$$

The term frequency (tf) portion is a simple bag-of-words representation, which represents a document by an multi-set of the words that occur and their number of occurrences. A problem, however, is that when computing some form of similarity between two term frequency representations of a document, common words end up dominating the representations since they are frequent within documents and occur frequently across documents. Sometimes a stopword list is used to filter out overly common words (such as ″the″, ″and″, ″a″) that don't provide much semantic value to a document. The use of the inverse document frequency circumvents having to try to determine a stop list word list, by reducing the weight of common terms and increasing the weight of rare terms. This approach is based on the intuition that if two documents share a rare term like ″anthropomorphize″ is much more relevant than two documents a common terms like ″the″, ″where″, or even ″car″. This is particularly true when using bigrams such ″New York″ or ″United States″ as well as unigrams as part of the tf-idf representation. There are a variety of different ways to calculate both the tf and idf term for tf-idf which we will discuss further in the experimental setup section.

2.3 Deep Learning and Natural Language Processing

Natural language processing as a field deals with attempting to process text in order to perform various tasks automatically, such as tagging the part of speech identifier (verb, noun, adjective, ect.) for words in a sentence or predicting the sentiment expressed in a statement. Typically machine learning algorithms are used to complete these tasks and more recently neural networks have become the predominant machine learning model used. Machine learning is a subfield of artificial intelligence that builds statistical models by learning from data. This can be in a supervised fashion, mapping data X to either a set of labels in a classification problem or a real value in regression problems, or in an completely unsupervised fashion finding clusters of similar examples in a dataset. As mentioned, since the resurgence of interest in neural networks over the past decade, natural language processing has increasingly resorted to their use [16]. Neural networks are inspired by the functioning of the human brain, although significantly simplified. Neural networks are made up of layers of neurons and like the human brain, neurons are connected to each other (typically only from the current layer or the next layer). These layers are typically divided into 3 different types of layers: input layer, hidden layer, and output layer. An example of a simple feed forward network is given in Figure 2.1.

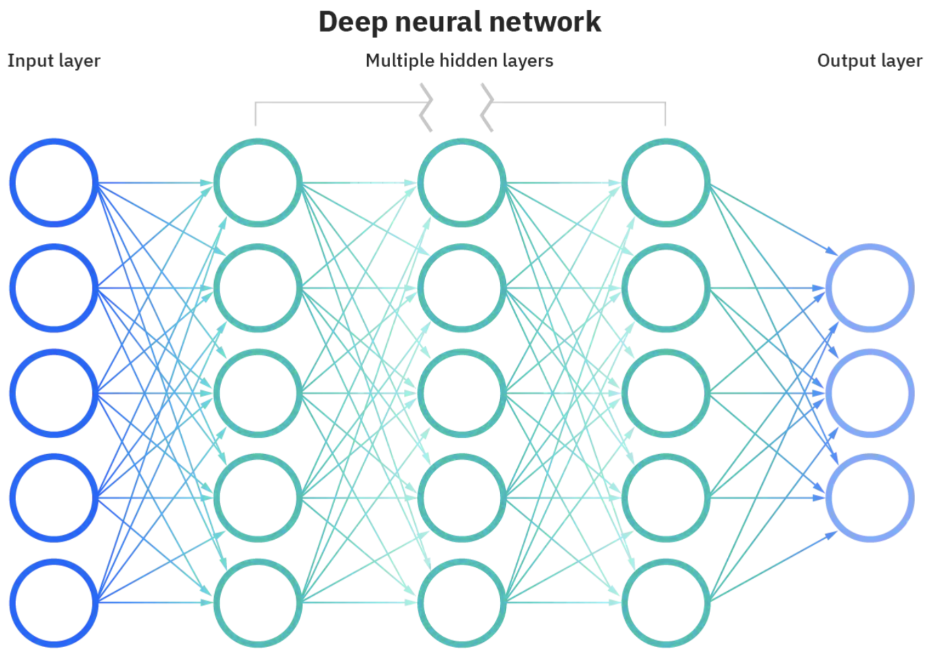


Figure 2.1: A deep neural network network with multiple hidden layers. [10]

Each neuron in each layer (excluding the input layer) is fully connected to each neuron from the previous layer. These connections, represented as edges in Figure 2.1, are weights that scale the influence of the value from a neuron from the previous layer. Thus, a given neuron's value is determined by the weighted sum of the previous layer's neurons and its associated weights to a the given neuron. In addition, an activation function is applied to the resulting value. The in order to model non-linear relationships in the data, a non-linear activation function is generally used, such as a sigmoid activation or the rectified linear unit (ReLU) activation.

Given that understanding natural language is a sequential task (i.e., reading each word in a sequential order), the traditional neural network architecture to tackle NLP problems is a recurrent neural networks (RNNs). An example of a simple RNN is given in Figure 2.2. For an RNN, each time step goes through a transformation to result in representation in the hidden layer. For the first element in the sequence, x_0, this representation, h_0 depends only on the input x_0. For the next time step, the hidden state h_1 depends on both x_1 and h_0. In this way, input from the previous time step, x_0 is also accounted via h_0. This continues for each element of the sequence. For NLP, x_n is generally a representation of the n^{th} word in the input sequence.

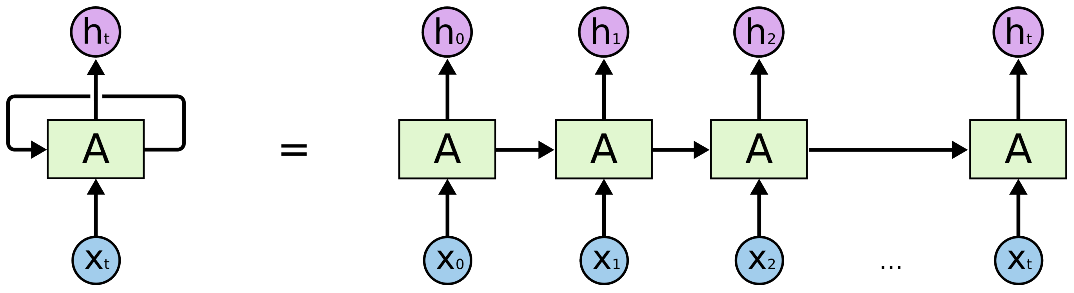


Figure 2.2: A recurrent neural network unrolled over t time steps. [37]

Variants of simple RNNs are much more widely in use, such as the long short term memory (LSTM) model [21] and the gated recurrent unit (GRU) model [7]. These models aimed to resolve optimization issues and problems with the RNN's ability to capture long-term dependencies. Still, two primary issues plague RNNs. First, due to the recurrent nature of the model, large-scale parallelization is not possible. Since h_1 depends on h_0, the two values cannot be computed in parallel, thus h_1 must wait until the value of h_0 has been calculated before proceeding. This restriction means RNNs are much slower to train and use for inference than other types of models. The second issue, mentioned before, is that despite the LSTM model and GRU model improvements, capturing long range dependency was still a problem. Given that the previous time step information had to be passed through as a fixed length vector, the amount of information it can hold is limited. Thus, as more time steps are processed, the information from many time steps ago is overwritten to write information about more recent time steps. This poses problems for tasks like name resolution, where resolving what "she", "him", "it" refers to may require looking back dozens of words in a passage to resolve.

As a solution to the lack of parallelization, the use of convolutional neural networks (CNNs) [27] for NLP tasks started to become more common [26, 62]. CNNs are most commonly used in tasks involving images, since that was the use case imagined when they were designed. CNN layers are composed of a mixture of filter and linear layers. The filter layers generally contain several different number of filters, with a defined receptive field. Each filter is a matrix of weights in the shape of the receptive field. Each filter will "stride" over the image, processing the inputs by multiplying the filter weights by the input. In general, when dealing with images, these receptive fields are square and usually only span a small number of pixels. With NLP tasks however, the receptive field width is generally a small number (since it equates to number of words), but the height corresponds to the dimension of the word representation used. Thus at a given layer, the receptive field which allows the model to learn context is quite provides a very localized context. However, as the CNN layers become deeper, the width of the context becomes wider. With a fixed sequence length, a sufficiently deep CNN can be able to have global context. A simple example of this can be seen in Figure 2.3, where the first layer only has a receptive field of two words, while the next layer is three words. In addition, the model can have efficient bidirectional context, while bi-directional RNNs are similarly inefficient. However, CNNs still suffer from long-range dependency issues, like RNNs.

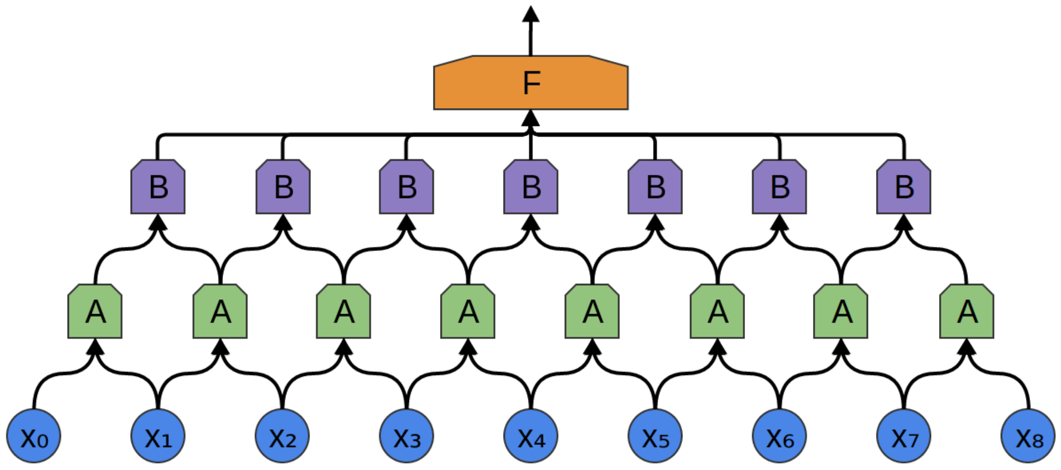


Figure 2.3: A simple convolutional network with increasing receptive field for input. [36]

Attention [3] was proposed as a solution to modeling long-range dependencies within RNNs by allowing any timestep to give more focus to any previous timestep. Originally used for neural machine translation, attention was utilized in encoder decoder architectures. An encoder-decoder architecture is composed of two components, an encoder whose job is to encode the entire input, and a decode whose job is to take the encoded representation of the input and produce output. As the length of the source translation increased, however, translation quality fell. To resolve this, attention allows the model to capture long-range dependencies by allowing it to ″attend″ to any point in time in the encoder. One version of this attention is Bahdanau attention [3], which for each step of the decoding process, derives attention scores for each time step of the encoder. The attention scores are then put through a softmax function and the weighted sum of the attention scores and the embedded representations from the encoder are used as inputs to the decoder. Thus, the model can in a sense choose which words to attend to at a given point can choose from any word processed by the encoder, allowing long-range dependency modelling. The version of attention used within transformer models is shown in Figure 2.4. The Q, K, V in the diagram refer to queries, keys, and values. For the transformer encoder, these are all derived from the same set of vectors, namely the embedded tokens, although with different transformations applied to them.

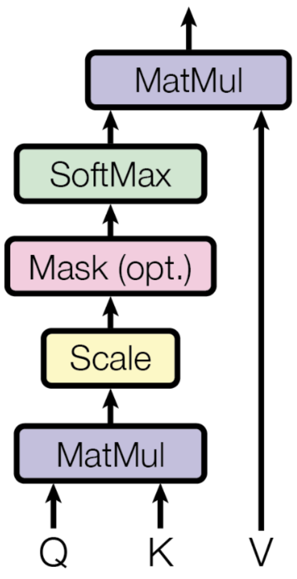


Figure 2.4: Diagram of the steps for computing scaled dot product attention, given a set queries, keys, and values vectors. [53]

Transformer models have gained virtually all the recent architecture attention due to taking advantage of all three of the improvements discussed, namely self-supervised learning of word embedding, better computational efficiency than an RNN, and use of attention. Among transformer models, BERT [9] is among the most popular given that it was the first to show off the performance that could be gained by large pretrained models on a variety of NLP tasks. The initial transformer paper [53] showed that by simply using a feed forward network to process each time step the transformer can model sequential data using positional encodings and attention. Positional encodings are needed since the feed forward networks do not model temporal information like RNNs or CNNs do. With the addition of multi-headed attention shown in Figure 2.5, the model can attend to any point in the input, thus can range short- and long-range dependencies like RNNs with attention, but is also computationally efficient with respect to time (i.e., high computational parallelism). In addition, similar to how word2vec [33], a method for learning word embeddings from large amounts of untrained data, trained word representations in an self-supervised fashion, the entire BERT model is trained in this way, not just the word embeddings. This is done through masked language modelling. The input and first encoding layer of a BERT model are shown in Figure 2.6.

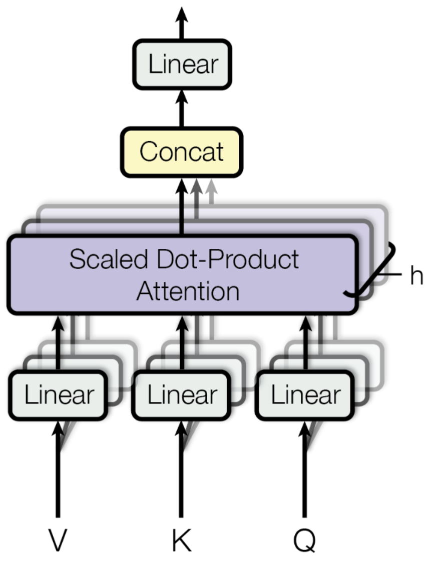


Figure 2.5: Multi-headed attention: The queries, keys, and values are transformed into h vectors of smaller dimension. Attention is applied to the h vectors, concatenated, then put through a linear projection. [53]

The idea of masked language modelling is to take an input word sequence scraped from some source (like Wikipedia, research articles, internet chat rooms) and randomly mask parts of the input. For example, for the input "The dogs drank out of their water bowl", the input to the model may be "The dogs drank out of their <MASK> bowl". The task of the model then, is to maximize $p($ water | "The dogs ... <MASK> bowl"). Doing this for hundreds of millions of examples, the model begins to learn not only the syntactic structure of the language, but also semantics and even some world knowledge. This portion is generally referred to as pretraining. Once a model is pretrained, the model weights can be used as the initial model for another task. This portion is referred to as fine-tuning, in which the pretrained weights are slowly updated for a specific task. This paradigm of using large pretrained transformers, pretrained on vast amounts of text data, and finetuning these models on other tasks has dominated the NLP field for the past few years since it not only gives state of the art performance on large datasets but is particularly effective on low resource datasets as well.

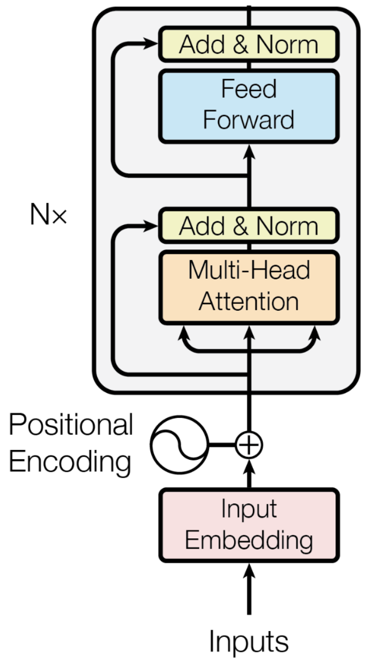


Figure 2.6: Transformer input and encoder layer. For BERT, the model is composed of only encoder layers with no decoder layers, unlike the first transformer model. [53]

2.4 Gradient Boosting

Gradient boosting [15] is a machine learning technique that leverages *weak learners* in an iterative fashion to build an ensemble of weak learners that can achieve high accuracy.

A weak learner in this context refers to a model who's prediction is just slightly better than random chance [14]. Often decision trees are used as the base model that will be ensembled. A decision tree is machine learning model that is represented by a tree. The nodes in the tree are either decision nodes, which determine which path through the tree the prediction will take, or are leaf nodes, which determine the final prediction. They are more commonly found in ensembled models, like a decision tree forest or gradient boosting. An example of a decision tree for predicting types of Iris flowers based on their pedal dimensions is given in Figure 2.7.

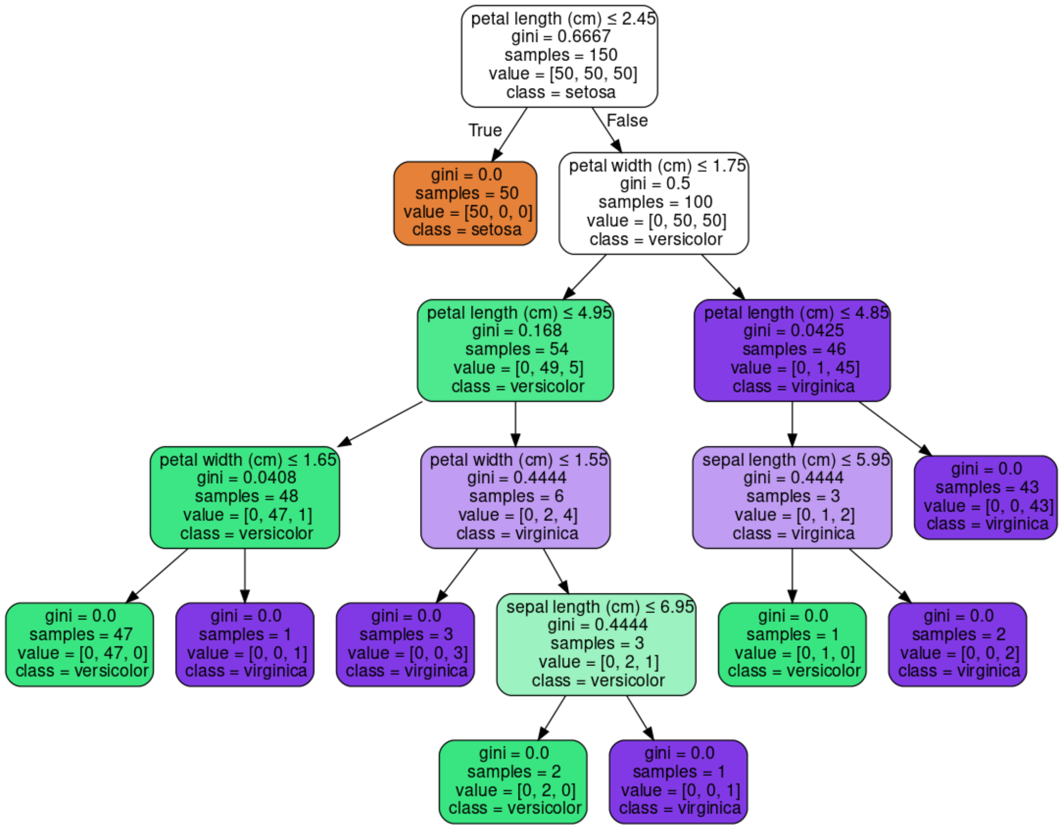


Figure 2.7: A decision tree for predicting 3 different species of Iris plant. [44]

The regression case is a bit simpler than the classification case for such a model, so I will give an example with a regression gradient boosting model. For a regression model, the algorithm from the paper that introduced gradient boosting is given below in Algorithm 1.

Algorithm 1 Least Squares Gradient Boosting

```
$F_0(\mathbf{x}) = \hat{y}$
for $m = 1$ to $M$ do
    $\tilde{y}_i = y_i - F_{m-1}(\mathbf{x}_i) \;\; i = 1, N$
    $\rho_m, \mathbf{a}_m = argmin_{\mathbf{a}, \rho} \sum_{i=1}^{N} [\tilde{y}_i - \rho h(\mathbf{x}, \mathbf{a}]^2$
    $F_m(\mathbf{x}) = F_{m-1}(\mathbf{x}) - \rho_m h(\mathbf{x}, \mathbf{a}_m)$
end for
```

The $F_0(\mathbf{x})$ term represents an initial "best guess" using a single value to best reduce the loss on the entire dataset. For least squared error loss function, this is conveniently equal to the mean value of the targets of the dataset, $\bar{y}$. In the next step the new targets, $\tilde{y}$ are generated for the current iteration using the previous model's predictions. In general, the targets will be the negative derivative of the loss function with respect to F_{m-1}. For least squared error loss (with a scaling factor of 0.5), this is exactly equal to the residual of the target value and the predicted value. The next steps finds ρ_m and a_m, which represents the model optimization steps for the new target values for iteration m. The optimization approach for the original gradient boosting uses steepest descent, which rather than using a constant value α to determine the step of the gradient update, it instead performs a line search over possible values for α to determine the best step value for the gradient update. The a_m simply represents the selected parameters for the model fit on the data, which for a decision tree represents the split values in the decision trees. Finally, a new round of predictions are formed using the model trained for iteration m, $h(\mathbf{x}, a_m)$ which is scaled p_m and added to the previous predictions from the last step $F_{m-1}(\mathbf{x})$. This is done for M iterations, which is the predefined number of models to be trained in the ensemble. Often an additional learning rate value v, commonly set to 0.1, is used as a regularisation step when adding the output values of a new tree, so rather the $F_m(\mathbf{x})$ will be generated by $F_{m-1}(\mathbf{x}) + v \cdot \rho_m h(\mathbf{x}, a_m)$

Chapter 3

Related Works

In this chapter we will discuss various works that implement systems for the FEVER task, or related tasks, primarily focusing on the various different kinds of approaches employed. We will first discuss approaches used in the initial competition, subsequent graph neural network approaches, and finally discuss transformer based approaches.

3.1 Initial Competition Systems

The third best system for the initial competition was published by Hanselowski et al. [18]. Their system like others is a three stage system utilizing a document retrieval system, sentence selection system, and claim classification system. Their document retrieval system detects noun phrases and named entities in claims and utilizes the Wikimedia API [13] to search for Wikipedia articles given a query term. For sentence selection, they utilize enhanced sequential inference model (ESIM) model [5] for predicting which sentences of the document are relevant to a claim. The ESIM model is a bidirection LSTM based model with attention developed for natural language inference tasks. They also use the same model for claim classification, all predictions are made over a 5 evidence set for a single score. Their system achieves a label accuracy of 68.49 and FEVER score 64.74 on the FEVER *dev* set.

The second-best system in the initial competition from Yoneda et al. [60] again utilizes the same three-stage system approach. For their document retrieval system, they retieve documents by checking for document titles within the claim text. To score documents for retrieval, they utilize a logistic regression model, using features such position, capitalization, and token match counts between the first sentence of the article and the claim. For scoring sentences they use another logistic regression model, deriving similar features from the claim and sentence sentences. Finally, for the claim classification model, they also use ESIM for a final prediction over a claim and 5 candidate evidence sentences. Their approaches produces predictions for each claim evidence pair and thus they use an neural network to aggregate the results into a final prediction. Their system achieves a label accuracy of 69.66 and FEVER score 65.41 on the FEVER *dev* set.

The best system in the initial competition from Nie et al. [35] additionally used a three-stage system. They use a neural semantic matching network (NSMN) model throughout their system. Although different than ESIM, it is similarly a bidirectional LSTM model with attention based model. Their document retrieval system utilized keyword matching to do an initial document retrieval model, but for scoring of retrieved documents, the document and claim text is put through a trained NSMN to score each document. A NSMN is trained for sentence selection and another is trained for claim classification. In total, 3 NSMNs are used in their pipeline.

3.2 Transformer Based Approaches

The first approach to utilize BERT into their system was Soleimani et al. [48]. Similar to the other methods, their system utilizes a three-stage system. For document retrieval, they use a more traditional approach of building tf-idf representations for each document and computing a distance metric to retrieve relevant documents. From the retrieved documents, sentences are scored by a fine-tuned BERT model. Finally, for claim classification, a BERT model produces a prediction for each sentence selected by the sentence selection model and a heuristic is used to aggregate these predictions into a final prediction. A substantially different approach taken by Jiang et al. [23] was to take a transformer model built for sequence-to-sequence generation and force the model to be used for sequence classification. T5 [41] is a very large transformer built for sequence generation, however in the work the authors show that when sequence classification tasks (such as sentiment analysis or natural language entailment) are cast as sequence generation (such as predicting the work "positive" for sentiment analysis) T5 can perform very well. Jiang et al. took a similar approach with T5 casting both the sentence selection task and label prediction task as sequence generation tasks. For the sentence selection task, candidate sentences are scored by representing input as "Query: q Document: s Relevant:", where q is the claim and s is a candidate sentence to be used as evidence. Candidate sentences are then ranked by measuring the probability of "true" and "false" tokens in the models outputs. Once they have 5 candidate pieces of evidence from their sentence selection model, they form the label prediction task as "Query: q sentence1: s_1 sentence2: s_2 ... sentence5: s_5 relevant:" and they use the output probabilities of "refutes", "supports", "noinfo" to form a prediction. They both try zero-shot transfer, in which they do not train the model on any FEVER data and get very good performance with no training. When training on the task, using the input representations from above, they achieve one of the strongest performing systems on the dataset. Finally, the previously best performing transformer based model was from [49]. They model the sentence selection stage as a token-level prediction improves the sentence selection performance. Generally this task is treated as a sequence classification task, where for a given sequence a single prediction is formed. In their approach, they generate a binary prediction for each token in a candidate document that determines relevancy. From these scores, they take the average over each token in a sentence to derive a score for a sentence. A benefit from this is that the additional context surrounding sentences can be incorporated, rather than looking at each sentence in isolation. To achieve this, they must utilize a more memory efficient transformer model [61], since the original transformer models tend to only use a maximum of 512 tokens since the complexity of the self-attention is quadratic in terms of input length. For the final label prediction, they utilize a DeBERTaV2-XL model [19] already finetuned on the multi-genre natural language inference (MNLI) dataset [57]. This dataset is a very similar, where given a statement the task is to determine if another statement is a contradiction, neutral, and entailment of the first sentence. Starting from such a checkpoint improves performance, in a similar manner to how starting from a pretrained model gives better performance for an uninitialized model.

3.3 Graph Neural Network-Based Approaches

Graph neural networks (GNNs) have recently been increasing in popularity and several different approaches have been applied to the FEVER dataset. A key distinction between the standard feed forward neural network and graph neural networks is how input and how the transformations on the input are done. A GNN has nodes and edges much like a standard graph. The nodes represent the input and within the GNN there is a feed forward network the runs through each node individually as input. The edges on the graph input define the relationship between the nodes and this additionally affects the input. Nodes are represented as learned vectors and at each layer (GNNs have multiple layers like normal neural networks), the GNN aggregates information about surrounding nodes to update the vector for each node. One can see how the FEVER task can be well formulated as a GNN: first nodes represent pieces of evidence (and embeddings can be computed via a pretrained language model) with edges between these nodes representing the relation between the evidence. In Figure 3.1, an update step for a GNN is shown, with an addition aggregation function for neighboring nodes. The transformation function is generally handled by a trainable neural network. In some GNN formulations, the edges also are learned embeddings which affect the aggregation, although for the GNNs used in FEVER, it appears none of the implementations use this formulation.

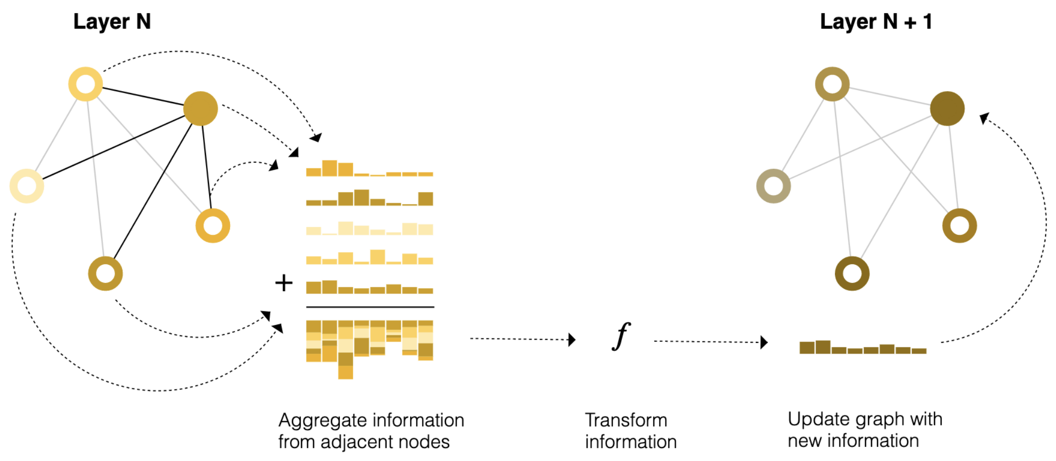

Layer N
Layer N + 1
+
f
Aggregate information from adjacent nodes
Transform information
Update graph with new information

Figure 3.1: A GNN with a simple addition aggregation for node updates. [43]

The GEAR system from Zhou et al. [65] was the first published approach using a GNN to form claim predictions. Like others, they utilize the document retrieval and sentence selection system from the Athene system [18]. For each claim, they create a graph representing the evidence and claims. Nodes in the graph represent the claim and the evidence sentences concatenated with the claim for each piece of evidence retrieved. So, if they retrieve 5 pieces of evidence, the graph will have 6 nodes, one for each evidence sentence concatenated with the claim sentence and 1 for the claim sentence by itself. The initial node embeddings are produced by running the inputs through BERT and the graph is initialized as fully connected (i.e., each node is directly connected to every other node in the graph). With the final output state of the graph, they combine the final node representations by using an aggregation function (eiher mean, max, or an attention based approach) and with the final, single representation, a prediction is formed. Although GEAR did not outperform initial transformer implementations [48], it demonstrated the validity of the approach. The KGAT system from Liu et al. [31] followed up GEAR by making some key improvements. They followed GEAR by utilizing the same document selection approach, however they developed their own sentence selection approach using the BERT model. For the GNN, rather than aggregating node representations to make a final prediction, they form node level predictions (so each piece of evidence has its own prediction) weighted by the retrieval score for the particular evidence. Finally, all the prediction scores are summed over each node for a final prediction score. KGAT is an important system, as it was the first GNN system to demonstrate superior performance to pure transformer base approaches. Additionally, their sentence selection outputs are widely used by other systems.

Chapter 4

Methods

Like many of the other approaches, we split the FEVER task into 3 sub-problems: document retrieval, sentence selection, and claim classification. We include another additional re-retrieval stage from after the sentence selection step. In the following sections, we will discuss the setup and approach to solving these problems.

4.1 Document Retrieval

For document retrieval, we employ two separate systems. The first is a tf-idf document-based retrieval system. The second is a fuzzy string search based on named-entities found in the claim. For tf-idf we utilize a modified version [1] of the Scikit-Learn library [38]. We use a modified version, as the standard formulation of idf is not supported by the released library. Once tf-idf document representations are built, claims and documents are transformed into their tf-idf representations. To compute similarity, a simple dot product function is used. The tf-idf must be stored as sparse matrix representations, as storing the non-sparse version would require too much memory. This storage form also improves dot product computation due to the high degree of sparsity of the matrix.

At a the document level for tf-idf, we explore two different approaches for dealing with

[1] https://github.com/mitchelldehaven/scikit-learn

Wikipedia page titles and article content. The first simply looks at the title and document together for building the tf-idf document representation, in which case we concatenate the strings for both together. This allows a single representation for a document which ensures faster document retrieval. The alternative approach to look at is to split titles and documents into different representation models and do document retrieval for both individually. The main idea behind this is that the optimal parameters for building tf-idf for titles may differ from the optimal parameters for building tf-idf for documents. For example, all Wikipedia page articles are what is referred to as "Sentence case". Sentence case after sentences due to the fact that the first letter of the string is always capitalized with the remainder of the sentence being cased normally. Thus, for a term like "water" the Wikipedia page title for the associated page will be "Water". This is a problem since if the tf-idf is cased, these two words are indexed as different words and thus won't be counted as a shared word occurring in both documents. Obviously, ignoring case can cause problems as well, as the lack of case when searching for "Fox" the broadcasting company could have problems with "fox" the animal. Separating the representation allows for a parameter search in between these approaches for titles and documents. This is, however, at the cost of slower query speed, as now two tf-idf based retrievals must be made.

As mentioned previously in Chapter 2, both the tf and idf terms of tf-idf have a variety of different formulations. For example, rather than taking the raw term count for the tf portion, a common approach is to take $\log(f_{t,d} + 1)$. In addition, there are a variety of other parameters within the Scikit-Learn API that can be tuned. Thus, we define a grid over these parameters and test all parameter combinations, selected the best set of parameters by measuring document recall @ 10. For the concatenated tf-idf version this simply involves returning the top 10 most similar documents via dot product. For the title and document split approach, we return the top 5 most similar from each, to give 10 total.

The *lowercase* hyper-parameter, when True, converts all text to lowercase before building the representations. *ascii* converts all text to ascii. Unicode swaps are distractors used in the dataset, thus this preprocessing approach can fix some of these issues. The *norm* hyper-parameter, when set to L2, will apply L2 normalization to the tf-idf vectors for document representations. This can help modelling text that varies greatly in length. For *sub-linear tf*, True will apply sub-linear term frequency. Sub-linear term frequency applies a logarithm to term frequency and thus as term frequency increases, it scales sub-linearly rather than linearly. The intuition behind this approach is approach is discussed in [32], claiming that it is unlikely that a term occurring 20 times in a document carries 20 times as much value in a information retrieval context than a term occurring just a single time. Finally, *max ngram*, when set to 1, will only use single terms in the vocabulary, so a ”New York” will only be represented as [”New”, ”York”]. When *max ngram* is set to 2, ”New York” will be represented as [”New”, ”York”, ”New York”]. Using bigrams can improve retrieval by modelling co-occurring terms together, rather than splitting them apart.

The second component of our document retrieval is a named-entity recognition-driven fuzzy string search. This approach is derived from [18]. In their system, they use a similar named-entity recognition approach to find query terms from claims. From the query terms they find, they used the WikiMedia API [13] to find the relevant Wikipedia documents. Within our system, we replace the WikiMedia API calls with a version of fuzzy string search, for a couple of reasons. The first is that given the ever changing nature of a live API, the document retrieved may change over time. Second, avoiding the WikiMedia API allows this approach to be applied to other types of datasets in which such an API is not available.

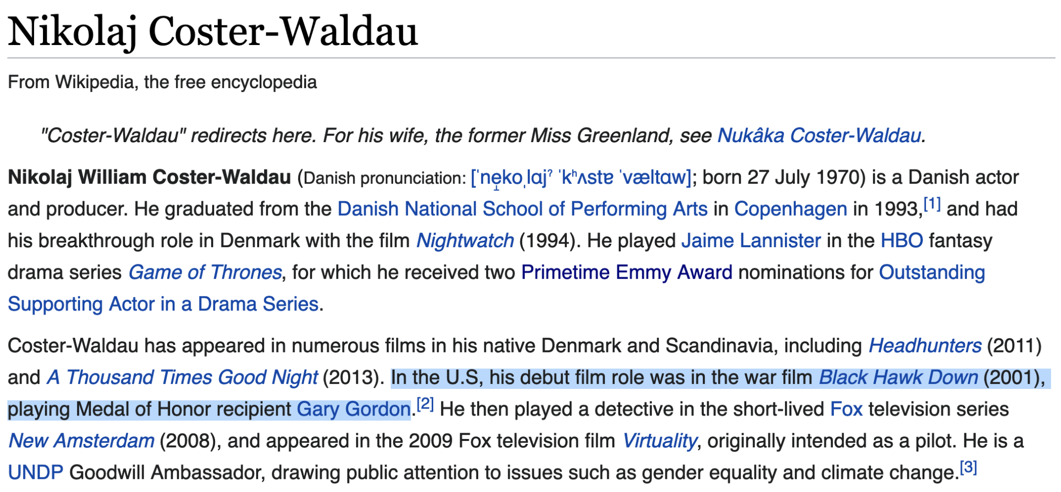
Within our fuzzy string search, the distance metric is Levenshtein distance. The distance compares two strings by measuring the total number of single character edits required to make two strings identical through insertions, deletions, and substitutions. To implement this, we use Sqlite's [20] virtual table, *spellfix1*, which allows queries to be made against stored strings in the database and for the most similar results to be returned. Given that there are over 5 million article titles, the very performant Sqlite implementation of fuzzy string search is essential. The number of documents returned is variable for this step: by default the top 7 are returned for each query, however if there are additional results with an edit distance ≤ 3 total edits, up to 21 queries will be returned. These documents are combined with the documents retrieved from the tf-idf portion to complete the document retrieval step of the pipeline.

4.2 Sentence Selection

From the document retrieval step, there are many documents determined to be relevant for a given claim. For each of these documents, there are a number of sentences that make up the entire document. The task for this step of the pipeline is to score the sentences from each document to get a final top 5 sentences to have an evidence set to make a final prediction for the claim.

For this step, we use a Pytorch to train a RoBERTa large classification model to predict for a given sentence, whether or not it is a relevant sentence for the claim. The results of this model are used to generate a score to rank the sentences in terms of relevancy. The standard approach for this step is to train a binary classifier for this task, predicting RELEVANT or NOT RELEVANT as the tasks targets. For this setup, the model output of $p(\text{ RELEVANT } \mid \text{claim, sentence})$ is used to rank the candidate sentences. An alternative approach we devise is to treat the sentence selection model in a similar fashion to the claim classification model. Specifically, rather than treating the task as a binary classification problem, the model is trained on the 3-class problem and attempts to predict whether a claim, sentence pair is REFUTES, NOT ENOUGH INFO, or SUPPORTS class. For this approach, the value used to rank the sentences is instead $1 - p($ NOT ENOUGH INFO $\mid$ claim, sentence$)$. We devised this approach when we noticed from initial experiments that the retrieval scores for correct evidence for SUPPORTS classes were higher on average than REFUTES evidence when using a binary label set. Our speculation was that due the higher sentence similarity that SUPPORT claims have with their relevant evidence the retrieval scores generally were higher.

For preparing the input for the model, some special steps need to be taken. Given that the evidence for a given claim can come from anywhere in the article, it is often the case that the sentence will not have enough relevant information on its own to form a prediction. This is because the subjects will be referred to via pronouns outside of the introductory passages. For example, in the passage shown in Figure 4.1, suppose the highlighted sentence is the evidence for the claim ″Nikolaj Coster-Waldau has played a metal of honor recipient in a movie″. If taken in isolation, what ″his″ refers to is unclear to the model, as it refers to a subject in another sentence and the sentences are scored individually. A fairly simple approach we employ is to simply concatenate the title of the document onto the sentence. Often the pronouns used in the articles refer to the subject of the article, so this simple approach is an effective way to provide the relevant information to the model when predicting the relevancy of a sentence. An alternative to this would be to run name entity resolution over the entire corpus, so that pronouns were resolved to the correct named entity, however this would be computationally demanding to do over the entire corpus.


Nikolaj Coster-Waldau

From Wikipedia, the free encyclopedia

"Coster-Waldau" redirects here. For his wife, the former Miss Greenland, see Nukâka Coster-Waldau.

Nikolaj William Coster-Waldau (Danish pronunciation: [ˈne̝koˌlɑjˀ ˈkʰʌstɐ ˈvæltɑw]; born 27 July 1970) is a Danish actor and producer. He graduated from the Danish National School of Performing Arts in Copenhagen in 1993,[1] and had his breakthrough role in Denmark with the film *Nightwatch* (1994). He played Jaime Lannister in the HBO fantasy drama series *Game of Thrones*, for which he received two Primetime Emmy Award nominations for Outstanding Supporting Actor in a Drama Series.

Coster-Waldau has appeared in numerous films in his native Denmark and Scandinavia, including *Headhunters* (2011) and *A Thousand Times Good Night* (2013). In the U.S, his debut film role was in the war film *Black Hawk Down* (2001), playing Medal of Honor recipient Gary Gordon.[2] He then played a detective in the short-lived Fox television series *New Amsterdam* (2008), and appeared in the 2009 Fox television film *Virtuality*, originally intended as a pilot. He is a UNDP Goodwill Ambassador, drawing public attention to issues such as gender equality and climate change.[3]

Figure 4.1: The highlighted text is an example of problematic pronouns when scoring sentences independently. Who "his" refers to is ambiguous without the surrounding context.

For the data provided within the FEVER dataset only two of the classes, REFUTES and SUPPORTS, provide the relevant evidence needed to make a correct prediction. For NOT ENOUGH INFO only the label is supplied, given that the NOT ENOUGH INFO indicates a lack of evidence rather than any given sentence in the corpus. Thus, to build a sentence selection model in either the binary or the ternary label case training examples must be generated. To do this, we employ negative sampling. Negative sampling as a technique can be used in a variety of circumstances, but in our case when training a classifier to determine which evidence sentence is relevant for a given claim, the model should be able to properly distinguish irrelevant sentences from relevant ones. The problem then is that the dataset only provides training samples for relevant sentences for a given claim. Thus, these irrelevant sentences must be generated via negative sampling.

In general, negative sampling draws randomly from a set of candidate data points to generate random samples. A different approach, referred to as hard negative sampling, tries to find negative samples that are more difficult to predict than a random sample. At the beginning of training, the random negatives may provide useful examples to train from. However, purely random sampled negatives can generate examples that as the model trains because trivially simple and thus provide little information to the model as training continues. Our approach to negative sampling is much closer to hard negative sampling, as we do not randomly sample from the entire set of sentences in the corpus. Instead, given that the document retrieval step returns many sentences that are not the true evidence for a given claim, we randomly sample sentences retrieved by the document selection process to generate negative samples for training. In this way, the sentences are more likely to be related to the given claim than randomly sampled sentences and thus more difficult, but are still not relevant to the verification of the claim. Given that the labelling process of finding all relevant sentences for a given claim is not always exhaustive the negative samples drawn may not be true negatives, although they should be infrequent enough to not be problematic to training. The main variable to this approach is determining how many negative samples to draw. We test a range of values to determine which one yields the best recall @ 5.

4.2.1 Evidence-Based Re-retrieval

An important part of the pipeline that we will discuss here is what we refer to as "evidence based re-retrieval". The name is from the fact that after doing an initial round of retrieval, we use the evidence we retrieved to do another round of retrieval. This is necessary as there are many claims where the correct evidence is not easily accessible from the claim. An example of this is given below.

c: Stomp the Yard stars an actress born in July.
e1: The film stars Columbus Short, Meagan Good, Darrin Henson, Brian White, Laz Alonso, and Valarie Pettiford, with Harry Lennix, and, in their film debuts, R&B singers Ne-Yo and Chris Brown.

e2: Valarie Pettiford (born July 8, 1960) is an American stage and television actress, dancer, and jazz singer.

In this circumstance, there is a very low chance that e2 is retrieved because the e2 represents the entire introduction for that particular article (the corpus is only compose of Wikipedia introductions). Note there is no mention of the movie "Stomp the Yard" in e2 (and thus the entire introduction), so the only words that will be in common between c and e2 will be "July" and "actress". There will be many articles in which these words occur, they may occur multiple times, and the other words may occur as well. This is to say simply that tf-idf will likely not place Valarie Pettiford article in top selections. In addition, the named entity based fuzzy string search will also not retrieve it, as "Stomp the Yard" and "July" are the only named entities in the claim text. Thus, even though the sentence selection model may be able to select e2 as relevant, the model can't provide a score if the document it is from is never retrieved. However, by looking at e1, a sentence that will likely be selected as evidence, we can see that Valarie Pettiford is listed. Provided by the FEVER dataset is a list of hyperlinks for each evidence sentence. Thus, from e1 there is a connection to e2 we can use. Others have used the hyperlinks provided by the dataset as a means to improve their extraction by including all documents that are linked to retrieved documents, however to our knowledge Stammbach [49] was the first to do this form of re-retrieval.

Given all this, we perform re-retrieval in the following procedure. First, with the results of sentence selection, we have a top 5 set of evidence sentences. For each of these sentences, we gather the hyperlinks and add each document to a set of re-retrieved documents. Next, for each document in re-retrieved set, we add each sentence to a set of re-retrieved sentences. Then, for each sentence in the re-retrieved sentence set, we generate a score via the sentence selection model. At this point, we have a top 5 evidence set from the initial retrieval and an additional set of many more candidate evidence sentences, so we must merge these into a final top 5 set. The approach we took differs from [49] in that we re-weight the sentence selection scores of the re-retrieved sentences by the sentence score of the initial evidence it was linked to. Thus, for a given piece of evidence $e_{i,j}$ which is evidence e_j derived from evidence e_i, the sentence selection score is $ss(e_{i,j}) = ss(e_i) * ss(e_j)$. This is based on two assertions: first is that a piece of evidence derived from another piece of evidence should not have a higher score than the evidence it was derived from and the second is that the score from a derived piece of evidence should be proportional to the evidence it was derived from. With this new score for re-retrieved evidence, the final top 5 set is based on the scores from the initial top 5 and the re-retreived, rescored sentences. In the entire pipeline, this extra step has a larger than anticipated impact on performance, as will be seen in the Section 6.2.

4.3 Claim Classification

For our claim classification model, this involves a two step processes, depending on the configuration. Previous works have taken two different approaches to the problem of claim classification using a transformer model. The first looks at each sentence and claim pair to form a prediction individually. By doing this, the results of the model with 5 pieces of evidence is thus 5 predictions. Given that the task is to produce a single output for a given claim, one must aggregate these predictions into a single prediction. We treat the aggregation as a modelling problem, training a new model particularly for aggregation. A problem with the approach of generating a prediction for piece of evidence, however, is that 10% of the claims in the dataset require multiple hops across sentences in order to verify them. For example, for a claim such as "A team captain of the Green Bay Packers was born in October" may require the sentence "Aaron Rodgers is the offensive team captain for the Green Bay Packers" in addition to "Aaron Rodgers was born in October" in order to correctly classify the claim. By looking at each candidate evidence sentence individually, these types of claims are not properly modelled. The second approach attempts to rectify this issue by concatenating all the candidate evidence sentences together into a single passage, then predicting from the single passage and the claim. In this case, multi-hop claims are modelled and since there is only one passage there is no need for an aggregation model. Our new approach is a simple extension of this, which is to take each candidate sentence and claim individually as well as the concatenation of them all together as a single string as input to the model. In this case, there are 6 sentences/passages input to the model and thus this setup also requires an aggregator model.

4.3.1 Claim Classification Transformer

For the transformer model, we again use RoBERTa large, the exact model used by previous graph neural network models we plan to compare against. Given that we use 3 different approaches to the claim classification setup, the training input varies based on those approaches.

When predicting individual pieces of evidences with an aggregator model, the training of the model includes individual claim and evidence sentence pairs, with their associated label. The training samples are generated by taking each SUPPORTS and REFUTES claim and concatenating the associated evidence sentences to make a single sentence for each claim evidence pair, with its associated label. Two examples are given below

For NOT ENOUGH INFO training data, for each claim that has this label, we take the top 5 predictions generated from the sentence selection model to provide NOT ENOUGH INFO training samples. Now, as a result of the imbalance between SUPPORTS and REFUTES claims and the fact that we take 5 examples for NOT ENOUGH INFO, the training dataset generated is highly imbalanced. Both the *dev* and *test* sets are balanced, e.g., both have 6666 SUPPORT, 6666 REFUTES, and 6666 NOT ENOUGH INFO samples in each set. Thus, since the training dataset is imbalanced while the testing sets are balanced, either over sampling minority classes or re-weighting the loss function needs to be done to obtain optimal performance on the testing sets. As discussed in the previous section, during sentence selection we opted for over-sampling, however the justification for doing so in that circumstance does not apply here, thus we simply re-weight the loss function to make poor predictions on minority classes more costly. A modified version of HuggingFace's popular Transformer library [58] was used to do this [2].

For the second approach to claim classification in which all the evidence is concatenated into a single input, rather than individually, the gold evidence from the training dataset is not directly used. Instead, similar to how the NOT ENOUGH INFO was treated, all the training samples use the top 5 selected evidence from the sentence selection to create the input. The input is thus the claim with up to 5 sentences concatenated info a single input representation. The justification for doing this is that at inference time, the model will have to be able to filter out irrelevant evidence that made it to top 5. If at training time the model only was given relevant information for SUPPORTS and REFUTES claims, the model would not have had to encounter this issue during training and this kind of mismatch between training and inference can cause issues with model performance. It should be acknowledged, however, that since the sentence retrieval component is not perfect, i.e., recall @ 5 is not 1.0, that some the training examples will not contain all the correct evidence to make a prediction. In our approach, if none of the golden evidence is found in the claim, then the training example is skipped. However, we do not require that all the golden evidence be found in the top 5 evidence to admit a sample into the training set.

[2]https://github.com/mitchelldehaven/transformers

For the final approach, the training setup is identical to the the concatenated approach that is discussed above. The only meaningful difference between the two is that at evaluation time, the model generated a prediction for each claim and predicted evidence pair individually, as well as the concatenation of each of the evidence with the claim. Thus, a total of 6 predictions (one for each evidence and one for the concatenated representation) are generated. An aggregation model must be used for this approach, similar to the individual approach.

4.3.2 Aggregation Model

For the aggregation model there are wide variety of machine learning models that one could choose, as the task is simply taking a 5x4 matrix (or 6x4 matrix) and forming a final prediction. Due to the dominance of gradient boosting algorithms when dealing with tabular data [46], we use a gradient boosting forest for building an aggregation model.

For the input to the aggregator, there are 4 fields. The first 3 are from the claim classification model softmax scores for each of the 3 labels for the claim, evidence pair. The final field is from the sentence selection model retrieval score for the evidence. This is important because the aggregator will not see the source sentences, thus has no notion of how relevant some of the evidence without these scores.

Chapter 5

Experimental Setup

5.1 Document Retrieval

To determine the best set of hyperparameters, we use a grid search over over the hyperparameters mentioned in Section 4.1. When using grid search, all possible hyperparamter combinations are used to build the tf-idf representations and each is tested to determine the best set of hyperparameters. We looked at 5 different hyperparameters, each with two possible values, resulting in a total of 32 different tf-idf configurations. In Table 5.1, we show all the various hyperparameter configurations with an associated experiment ID for tracking configurations. For each configuration, we test the configuration with the title and document concatenated into a single string and also test keeping the title and document portions separated. Thus, in total, we have a total of 64 different tf-idf model representations built on the corpus. For testing the performance of each configuration, we use the *dev* set to measure document recall @ 10. When using the concatenated tf-idf representation, we take the first 10 documents total. When using the title and document separated representations, we take the first 5 from the title representations and the first 5 from the document representations to get a total of 10 retrieved documents.

Table 5.1: Various hyperparameter combinations for TF-IDF

Experiment ID	Lowercase	ASCII	Norm	Sublinear TF	Max Ngram
Expt 1	True	ascii	l2	True	1
Expt 2	True	ascii	l2	False	1
Expt 3	True	None	l2	True	1
Expt 4	True	None	l2	False	1
Expt 5	True	ascii	None	True	1
Expt 6	True	ascii	None	False	1
Expt 7	True	None	None	True	1
Expt 8	True	None	None	False	1
Expt 9	True	ascii	l2	True	2
Expt 10	True	ascii	l2	False	2
Expt 11	True	None	l2	True	2
Expt 12	True	None	l2	False	2
Expt 13	True	ascii	None	True	2
Expt 14	True	ascii	None	False	2
Expt 15	True	None	None	True	2
Expt 16	True	None	None	False	2
Expt 17	False	ascii	l2	True	1
Expt 18	False	ascii	l2	False	1
Expt 19	False	None	l2	True	1
Expt 20	False	None	l2	False	1
Expt 21	False	ascii	None	True	1
Expt 22	False	ascii	None	False	1
Expt 23	False	None	None	True	1
Expt 24	False	None	None	False	1
Expt 25	False	ascii	l2	True	2
Expt 26	False	ascii	l2	False	2
Expt 27	False	None	l2	True	2
Expt 28	False	None	l2	False	2
Expt 29	False	ascii	None	True	2
Expt 30	False	ascii	None	False	2
Expt 31	False	None	None	True	2
Expt 32	False	None	None	False	2

5.2 Sentence Selection

We look at the impact of the number of negative samples used and modelling sentence selection as a binary or ternary classification problem. In addition, we perform a search of hyperparameters, including learning rate and label smoothing. Learning rate is fairly important with all pretrained language models. A known problem that can occur when too high of a learning rate is used is called catastrophic forgetting. This occurs when the model ”forgets” what was learning during masked pretraining. The ”forgetting” occurs when updates are made to the model weights during training and when the learning rate is too high, the weights change too rapidly and the model ”forgets” what was learned during pretraining. Setting the learning rate too low has its own consequences, primarily resulting in the model training much slower than necessary. We test various values of label smoothing to test if it is effective on this dataset as well. Label smoothing, as the name implies, smooths the hard target labels of training instances. Usually for classification, the labels are represented as 1-hot vectors. For uniform label smoothing [51] the vector is ”smoothed” by reducing the label index in the vector from 1 and distributing the value of the reduction across the remaining classes uniformly. The reduction factor is what can be tuned here. An example of this processes is shown below.

In addition, regarding the learning rate, we follow standard practice with fine-tuning pretrained language models, we use a two-stage learning schedule. The first stage is a ”warm-up” cycle, in which the learning rate is slowly increased from a very small value up to a specified learning rate at its peak. In the second stage, the learning rate is annealed down to a very low learning rate again. The warm-up period is 10% of the training cycle.

For training the sentence selection model configurations, we use the entire *train* set to train the model. During training, 4 times per epoch, we measure the model's performance against the *dev* set and save the best performing model according to the model on the *dev* set. In this way, *dev* is used for model selection, but is not directly trained on. We train the sentence selection model for 2 epochs.

We optimize the hyperparameters and modelling decisions in two stages. First we look at negative samples and binary/ternary labels together. For the number of negative samples drawn, scan look at 5, 10, and 20 negative samples drawn per training sample. Taking the best model from that stage, we then tune hyperparameters, i.e., learning rate and label smoothing. We test the learning rates {3e-6, 7e-6, 1e-5, 2.5-e5, 5e-5} and the following label smoothing values {0.0, 0.1, 0.2}. Obviously this opens the possibility that there could be a more optimal set of decisions, since we aren't trying all possible combinations, but we accept that to reduce the number of fine-tuning runs that must be done. Each fine-tuning run takes approximately 10 hours on a NVIDIA RTX 3090 GPU, so running an extensive grid search over these decisions is quite expensive.

For training the models, we use Pytorch-lightning [2], the Transformers library [58], and Ray Tune [29]. Pytorch-lightning is a wrapper over the standard Pytorch library from Facebook used for building deep neural networks, which takes care of some of the more engineering based parts of training (tracking metrics, saving checkpoints, using GPUs). Transformers is a popular library for utilizing pretrained transformer models. Their API simplifies loading the relevant model weights and tokenizers for models. Finally, Ray Tune is a hyperparameter optimization library. It simplifies running the various hyperparameter configurations and automatically selects the best configuration.

5.3 Claim Classification

For the experiment setup for claim classification, it is essentially the same as the with sentence selection. The hyperparameter tuning looks at the same learning rate and label smoothing values as before with. Since we do not draw additional negative random samples, the training dataset is smaller than the sentence selection training dataset, so we train to 5 epochs instead. As before, we evaluate 4 times per epoch against the *dev* set and keep the best model checkpoint according to *dev* set performance.

After determining the best hyperparameter configuration, we train the 2 models (”mixed” and ”concatenate” use the same underlying model, the ”singleton” approach uses its own model) using the RoBERTa-large weights provided by Facebook. For a more fair comparison against Stammbach’s model, which uses DeBERTa V2 XL finetuned on MNLI [57], we also train models using RoBERTa large finetuned on MNLI. Although Stammbach’s model is substantially larger, by using RoBERTa large MNLI both models at least have be finetuned on the MNLI dataset, which is a very similar task to FEVER.

5.4 Aggregation Model

For our aggregation model, the training is slightly different. For both the sentence selection and claim classification models, the *train* set was used for training and the *dev* set was used only for model selection. For the aggregation model, we instead opt to use the *dev* set to train the model. The reason for this is that the aggregation model is being trained on the model outputs of the sentence selection model and claim classification model. Since both those models were trained on the *train* set, the scores they produce for those instances will be biased and over confident in the correct prediction, because the model has been trained on that data. This poses a problem when moving to the *test* set, since the scores produced by those models will not be biased on *test* set, thus creating a mismatch. By using the *dev* set, we avoid these issues.

By using the dev set, however, there is no more labelled data to test against, as it is either being used to train the aggregator or was already used to train previous models. To solve this problem, we use k-fold cross validation to generate a subset of *dev* to test various hyperparameters against. K-fold cross validation is commonly used machine learning technique to measure hyperparameters without needing additional testing data. The process involves splitting the training data into k equally sized partitions. Each partition will serve as a test set, while the rest serve as the training set. In our case, we use 5-fold cross validation, and thus 5 folds are created. For a given set of parameters, 5 models are trained, each time with a different set of 4 folds used for training and 1 fold used for testing. Whatever evaluation metric is being used is measure for each test set and the mean value of the metric for each test set is used to determine the quality of a given set of hyper parameters. Thus, when choosing a set of hyperparamters, the set with the best mean metric across the 5 test sets is chosen. In our case, the metric is simply label accuracy.

For training our gradient boosting algorithm, we use the XGBoost library [6], a popular gradient boosting library. The library allows using Scikit Learn's grid search functionality to fine the optimal hyper-parameters, so we also run XGBoost through a grid search. The hyperparameter values tested can be seen below in Table 5.2. The max depth for a gradient boosting classifier defines the maximum depth that each decision tree that forms the ensemble can be. The number of estimators defines the number of decision trees that will be used in the ensemble. Learning rate simply controls the rate of updates, as per usual in gradient based models with a learning rate hyperparameter.

Table 5.2: XGBoost classifier hyperparameter search values.

Hyperparameter	Value Range
max depth	$2, 4, 6, 8$
number of estimators	$20, 40, 60, 80, 100$
learning rate	$0.1, 0.3$

Chapter 6

Experimental Results

Following the format of the previous section, we discuss our experimental results separately by each part of the pipeline. First, we will discuss results of the grid search over TF-IDF parameters with fuzzy string search on document retrieval. Second, we will discuss the results for data preparation and model formulation of sentence similarity, with the additional hyper parameter search. Finally, we will discuss the impact of input representation for aggregation for claim classification, again with an additional hyper parameter search and contrasting the use of an aggregation model.

6.1 Document Retrieval

To begin, we will discuss the results of the hyper-parameter search used for finding the best setup for document retrieval using tf-idf. As discussed in the experimental setup for document retrieval, we ran a grid-search over a variety of different parameters and determine the best by measuring recall 10. Given that there are a total of 32 different combinations across hyper-parameters for the concatenated tf-idf setup and 1,024 (32 * 32) for the title and document separated approach, we will only include the top 5 and worst 5 parameters for both the combined tf-idf representations and title document separated representations, with full results being reserved to Appendix A. We include the worst performing for each to highlight that choosing reasonable hyper-parameters has a significant impact on performance, even for something as simple as tf-idf. The results for this grid search of the combined representation are shown in Table 6.1, indicating the document recall @ 10 on the development set.

Table 6.1: Concatenated TF-IDF dev set recall.

Experiment ID	Dev Recall @ 10
26	86.94 %
10	86.87 %
28	86.47 %
12	86.28 %
25	86.24 %
...	...
16	53.86 %
24	32.15 %
22	32.01 %
8	29.15 %
6	29.06 %

From the results, it is quite clear that hyper-parameter choice is quite important, with the difference between the worse 5 configurations and best 5 being quite significant. It is worth noting that for the top 5 results, all 5 have both max ngram set to 2 and norm set to ”l2”. Of the worst 5, all 5 have norm set to ”None”, thus no normalization is done to the vector representations of the documents, and additionally 4 of the 5 worst performing configurations had max ngram set to 1. Thus, modelling bigrams and normalizing the resulting tf-idf representations has a significant impact on performance.

In Table 6.2 (full results can be found in A.2), we have a similar table for the title and document separated approach. As mentioned previously, in this approach titles and documents are treated separately and top 5 retrieval is done on both. We combine top 5 from the title representations and document representations to have a final top 10. It is worth noting that this does not always produce 10 unique documents, since often the retrieval from the title representations and document representations will return some of the same articles. So this approach returns up to 10 documents, but often less than 10.

Table 6.2: Title and document separated TF-IDF dev set recall

Title Experiment ID	Document Experiment ID	Dev Recall @ 10
25	15	88.46 %
25	13	88.44 %
26	15	88.43 %
26	13	88.40 %
25	31	88.37 %
...	...	...
6	8	44.48 %
6	6	44.46 %
8	3	44.37 %
22	8	44.11 %
22	6	44.10 %

Similarly with the combined representation there is quite a difference between the best performing configuration and the worst, although difference is for the title and document separated approach. By separating the title and documents for retrieval, recall @ 10 for the best configuration improves over the best configuration on the combined representation by 1.52% absolute improvement. The best configuration from both setups significantly outperform the baseline system developed by the creators of the dataset, which achieved a recall of 65.86% @ 10. This does come at the cost of having to run an additional retrieval step (since 1 must be run for the document representation and title representation), although this is not a significant time cost. As can be seen in the two tables, although the title and document separated representation will return less documents overall than the concatenated representation, the best performing configuration outperforms concatenated representations by 1.52 % recall @ 10 on the *dev* set.

6.1.1 Adding Fuzzy String Search

In addition to our tf-idf base retrieval, we apply fuzzy string search over detected named-entities in the claim text. Due to how we apply fuzzy string search the total number of documents retrieved per claim is variable. Thus, we cannot provide an exact recall @ N metric, however we do compare our results against the full retrieval systems reported by others. For the development set our retrieval system returns less than or equal to 100 documents for 99.53% of all claims, thus we include the baseline provided by the dataset authors for recall @ 100. Unfortunately, many newer systems do not report their document retrieval system's performance and thus our comparisons are primarily drawn from earlier systems. This is likely due to the fact that many systems utilize [18] for their document retrieval. The oracle accuracy is the most commonly used metric used for those who report document retrieval metrics, which we will report on as well (in addition to recall). The oracle accuracy in this context indicates what accuracy is possible if each subsequent component in the pipeline is 100% accurate. Thus, for measuring oracle accuracy if all of the required evidence for a claim is in retrieved documents, then it is counted as correct. If the claim's label is NOT ENOUGH INFO, then it is counted as correct as well.

Our system achieves a 93.78% document recall with an oracle accuracy of 95.86%. Our oracle accuracy is compared with other systems on the development set in Table 6.3. It is difficult to make direct comparisons, since different system return a different number of documents. For example, for pure document recall [23] reports 96.87% recall @ 1000, which out performs our system by 3.09%, but they return 1000 documents to reach this recall level while our system returns on average 31 documents per claim. Our system also likely returns more document on average than others, besides LisT5, since our system is the combination of two different retrieval approaches.

Table 6.3: Oracle FEVER (OFEVER) Comparison of our retrieval system

System	Document OFEVER
FEVER baseline [52] (@ 100)	91.06%
Nie et al. (UNC) [35]	92.82 %
Zhou et al. (GEAR) [65]	93.33 %
Soleimani et al. (BERT) [48]	93.20 %
Hanselowski et al. (Athene) [18]	93.55 %
Subramanian et al. (HESM) [50]	93.70 %
Jiang et al. (LisT5) [23]	97.91%
Ours	95.86 %

6.2 Sentence Selection

As mentioned previously, we split the model tuning into two parts to reduce the overall compute needed for a full grid search. The first set of results in Table 6.4 shows the model performance for recall @ 5 regarding retrieved sentences with various negative sample values and ternary vs binary labels for sentence selection. For the learning rate we use 3e-6 as initial experimentation indicated this was a good learning rate. For label smoothing, we used a value of 0 which means no label smoothing was applied.

Table 6.4: Effects of negative sampling and label set changes on sentence selection

Negative Samples	Label Set	Dev Sentence Recall @ 5
5	binary	91.82 %
5	ternary	91.87 %
10	binary	91.73 %
10	ternary	**92.03** %
20	binary	91.91 %
20	ternary	91.95 %

From the results, the best performing model is ternary labels with 10 negative samples per positive example. However, the difference between the various setups is quite marginal, indicating that for recall @ 5 the difference between 5, 10, 20 negative samples per positive example and binary vs ternary labels is minimal.

For the next set of results, we use the generated dataset with 10 negative samples per positive example using ternary labels, since this is the best performing model from the previous step. Table 6.5 presents the results from searching over various learning rates and label smoothing values. The learning rate specified in the original RoBERTa paper is 1e-5 and thus we select values around this specified value to determine if other learning rates work better for this task. For label smoothing, typically values of 0.1 or 0.2 are common [34] and thus we used 0 (no label smoothing), 0.1, and 0.2 for searching. From the table below, we exclude results with learning rate 5e-5, as the resulting models ending up degenerating into very poor models due to the high learning rate.

Table 6.5: Effects of learning rate and label smoothing on sentence selection

Learning Rate	Label Smoothing	Dev Sentence Recall @ 5
3e-6	0.0	**92.03** %
3e-6	0.1	91.91 %
3e-6	0.2	91.90 %
7e-6	0.0	91.78 %
7e-6	0.1	91.83 %
7e-6	0.2	91.94 %
1e-5	0.0	91.76 %
1e-5	0.1	91.67 %
1e-5	0.2	91.87 %
2.5e-5	0.0	91.81 %
2.5e-5	0.1	91.68 %
2.5e-5	0.2	91.71 %

As can be seen, varying these parameters does not significantly affect performance, indicating that RoBERTa is somewhat robust to these settings when dealing with a large amount of data. However, the best performing model is with a substantially lower learning rate than usual and with no label smoothing applied.

Finally, for the last set of results we use our best model from the tuning process, which uses 10 negative samples per positive example, ternary labels, the lowest learning rate of 3e-6, and no label smoothing. Using this model we compare against other results for sentence selection in the literature. In the table we compare against both the recall @ 5 for sentence selection for both the development set and the test set. Unfortunately, many authors do not report test recall.

Table 6.6: Comparison of various sentence selection approaches

Author (System)	Dev Set Recall @ 5	Test Set Recall @ 5
Yoneda et al. (UCL) [60]	84.54%	-
Nie et al. (UNC) [35]	86.79%	-
Hanselowski et al. (Athene) [18]	87.10%	-
Zhou et al. (Gear) [65]	86.72%	-
Liu et al. (KGAT) [31]	94.37%	87.47%
Zhong et al. (DREAM) [64]	87.64%	-
Subramanian et al. (HESM) [50]	90.50%	-
Soleimani et al. (BERT) [48]	88.38%	-
Jiang et al. (LisT5) [23]	90.54%	-
Stammbach (BigBird) [49]	90.79%	-
+ re-retrieval	93.62%	-
Ours (RoBERTa Large)	92.03 %	-
+ re-retrieval	**94.41** %	**93.42** %

As seen in Table 6.6, our retrieval system without re-retrieval for sentence selection is only outperformed by LisT5 [23] and Stammbach [49] with re-retrieval. For KGAT [31], the authors claim that if the initial document retrieval does not return the correct sentences, they add them to the set. Clearly this helps their performance on the *dev* set, however their performance drops significantly for sentence retrieval when predicting on the *test* set. Comparing against Domink's system, without re-retrieval, our system outperforms theirs by 1.24% recall. With re-retrieval, our system outperforms all other systems, the closest being another system utilizing re-retrieval. As seen by the boost given by re-retrieval for ours and Stammbach's system, it is clear that it improves sentence selection performance significantly.

6.3 Claim Classification

For the claim classification model we have 3 different versions excluding any hyper-parameter considerations: the singleton model where each piece of evidence is considered separately, the concatenated model where all the evidence is considered together in a single input concatenated together, and a mixed model where each piece of evidence is considered individually as well as a concatenated version of all the evidence together. For testing hyper-parameters for fine-tuning, we use the concatenated model as this allows tuning hyper-parameters without using an aggregation model and we assume the selected hyper-parameters are the best for all versions of the model. In Table 6.7 are the results of the hyper-parameter tuning. We tune the same hyper-parameters as the sentence selection model with the same possible values. As before with sentence selection, we exclude 5e-5 learning rate results from the table, as the models always degenerated in to poor performers.

Table 6.7: Effects of learning rate and label smoothing on claim classification

Learning Rate	Label Smoothing	Dev Accuracy
3e-6	0.0	80.65 %
3e-6	0.1	80.69 %
3e-6	0.2	**80.82** %
7e-6	0.0	80.39 %
7e-6	0.1	80.30 %
7e-6	0.2	80.45 %
1e-5	0.0	80.10 %
1e-5	0.1	80.39 %
1e-5	0.2	80.40 %
2.5e-5	0.0	80.20 %
2.5e-5	0.1	80.09 %
2.5e-5	0.2	80.15 %

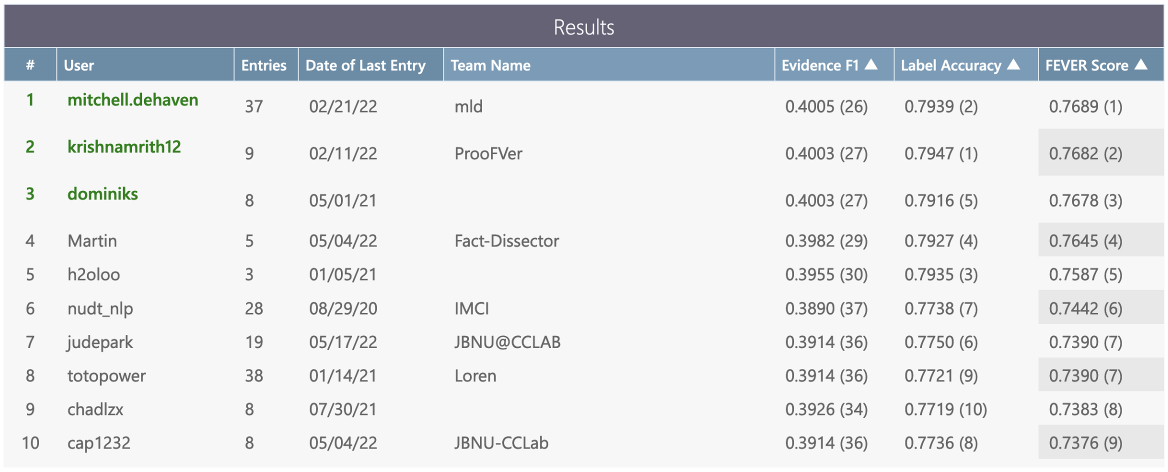
The best performing configuration uses 3e-6 for the learning rate and label smoothing of 0.2. Interestingly, some of the models with 2.5e-5 learning rate ended up as degenerate models at the end of finetuning. This is only a 2.5x increase over the suggested learning rate from the RoBERTa paper with the same batch size but still has a reasonable chance of failure. However, not only does a significantly lower learning rate such as 3e-6 not result in degenerate models but for both sentence selection and claim classification, it creates the best models. It appears that the suggested learning rate 1e-5 is more of an upper bound on the max learning rate, as trending to higher values can induce issues. Using the optimal hyper-parameters, we train a model using the RoBERTa Large MNLI checkpoint. We use an aggregation model for the singleton and mixed claim classification models. In Table 6.8 we include our three final models for comparison with other systems. For each system we generate scores with both initial retrieval and re-retrieval, to show the impact evidence base re-retrieval has in terms of performance. We report "-" where no metrics are provided and we also report "-" in instances where used the dev set for training the aggregation model, since this would give a biased view on our systems performance on that set.

As can be seen all versions of our model outperform all previously published approaches, including all of the GNN based models. The two published works that perform similarly are two transformer based approaches, however both utilize much large models. Stammbach's model uses DeBERTv2 XL MNLI [19], which uses approximately 900 million parameters while Jiang uses T5 [41], which has 3 billion parameters. Additionally, the "mixed" approach to prediction, where both predictions on single pieces of evidence as well as all the evidence concatenated outperforms the other approaches. We suspect this is due to the aggregator being able to determine whether to give more focus to single predictions or the concatenated prediction. When dealing with a claim that requires a single piece of evidence, concatenating a bunch of irrelevant information introduces a lot of noise and thus makes it more difficult. However, the single prediction approach

Table 6.8: Comparison of initial, GNN based, and recent transformer systems.

Author (System)	Dev LA	Dev FEVER	Test LA	Test FEVER
Yoneda et al. (UCL) [60]	69.66 %	65.41 %	67.62 %	62.52 %
Nie et al. (UNC) [35]	69.72%	66.49 %	68.21 %	64.21 %
Hanselowski et al. (Athene) [18]	68.49%	64.74 %	65.46 %	61.58 %
Zhou et al. (GEAR) [65]	74.84 %	70.69 %	71.60 %	67.10 %
Liu et al. (KGAT) [31]	78.29 %	76.11 %	74.07 %	70.38 %
Zhong et al. (DREAM) [64]	-	-	76.85 %	70.60 %
Subramanian et al. (HESM) [50]	75.77 %	73.44 %	74.64 %	71.48 %
Soleimani et al. (BERT) [48]	74.59 %	72.42 %	71.86 %	69.66 %
Jiang et al. (LisT5) [23]	**81.26** %	77.75 %	79.35 %	75.87 %
Stammbach (DeBERTv2 XL MNLI) [49]	-	-	79.20 %	76.80 %
Ours (RoBERTa Large MNLI) singleton	-	-	78.00 %	75.21 %
+ re-retrieval	-	-	78.01 %	76.09 %
Ours (RoBERTa Large MNLI) concatenated	80.82 %	77.27 %	78.74 %	74.97 %
+ re-retrieval	81.23 %	79.02 %	79.14 %	76.69 %
Ours (RoBERTa Large MNLI) mixed	-	-	78.99 %	75.46 %
+ re-retrieval	-	-	**79.39** %	**76.89** %

cannot model multi-hop predictions, thus the mixed approach is able to take advantage of the benefits of both approaches.


Results

#	User	Entries	Date of Last Entry	Team Name	Evidence F1 ▲	Label Accuracy ▲	FEVER Score ▲
1	mitchell.dehaven	37	02/21/22	mld	0.4005 (26)	0.7939 (2)	0.7689 (1)
2	krishnamrith12	9	02/11/22	ProofVer	0.4003 (27)	0.7947 (1)	0.7682 (2)
3	dominiks	8	05/01/21		0.4003 (27)	0.7916 (5)	0.7678 (3)
4	Martin	5	05/04/22	Fact-Dissector	0.3982 (29)	0.7927 (4)	0.7645 (4)
5	h2oloo	3	01/05/21		0.3955 (30)	0.7935 (3)	0.7587 (5)
6	nudt_nlp	28	08/29/20	IMCI	0.3890 (37)	0.7738 (7)	0.7442 (6)
7	judepark	19	05/17/22	JBNU@CCLAB	0.3914 (36)	0.7750 (6)	0.7390 (7)
8	totopower	38	01/14/21	Loren	0.3914 (36)	0.7721 (9)	0.7390 (7)
9	chadlzx	8	07/30/21		0.3926 (34)	0.7719 (10)	0.7383 (8)
10	cap1232	8	05/04/22	JBNU-CCLab	0.3914 (36)	0.7736 (8)	0.7376 (9)

Figure 6.1: Leaderboard for private FEVER test set. Our system is first overall for FEVER score, the primary metric being measure, and second overall for label accuracy.

To test just the predictive performance of our approach in a more direct comparison with a GNN based approach, we use the retrieval system of KGAT rather than our retrieval system. Since our retrieval system performs better than KGAT's, the improvements seen could simply be due to an improved retrieval system. We use the best sentence selection model to get a sentence score for the retrieved sentences from the KGAT retrieval system and generate predictions with the best claim classification model. Note we do not change what was retrieved by KGAT by using our sentence selection model, but only to add retrieval scores, since our aggregation model utilizes these scores in the final prediction. The only changes we make from our FEVER system is that we re-trained the aggregation model, since it is fairly low cost to train. In Table 6.9 are the results of this setup and as can be seen, even when restricting out model to use the exact same retrieval, our model outperforms KGAT by a fair margin.

Table 6.9: A comparison of our predictive model vs. KGAT utilizing the same retrieval system.

Author (Model)	Test LA	Test FEVER
KGAT (RoBERTa Large) [31]	74.07 %	70.38 %
KGAT (CorefRoBERTa) [59]	75.96 %	72.30 %
Ours (KGAT retrieval, RoBERTa Large)	76.60 %	73.21 %
Ours (KGAT retrieval, RoBERTa Large MNLI)	77.95 %	74.08 %

6.4 Beyond FEVER: Expanding to the Scifact Verification Dataset

As we have demonstrated, this fairly simple pipeline performs particularly well on the FEVER dataset, outperforming all the GNN based approaches that the claimed to outperform transformer based approaches. To test if this simple framework works outside of this dataset, we apply this same approach to another fact verification dataset, namely the Scifact [55] dataset. The Scifact dataset is very similar to the FEVER dataset. It is composed of claims which can either be supported, refuted by evidence and can also be unverifiable, like the FEVER dataset. The key differences are that Scifact deals only with scientific claims and the dataset is much smaller. It only contains 1,409 data points across the entire dataset (including the blind test set) and the entire corpus size contains abstracts from 5,183 scientific articles, making it $< 1\%$ the size of the FEVER dataset.

Aside from these differences, however, the task remains the same, which is for a given claim retrieve all the relevant information from the corpus for that claim and classify if the claim is supported, refuted, or if there is not enough information to do either. Given the small size of the dataset and that is is domain specific to scientific claims, we follow the approach taken with the MultiVers [56] system, which is the current state of the art on the task. Rather than starting with the RoBERTa large model, which is trained on a wide variety of data, we instead start with a RoBERTa model trained exclusively on biomedical data during pretraining [28]. This model is then fine-tuned on the FEVER dataset as we did previously. The resulting model is then fine-tuned again, on a variant of the PubMedQA [24] dataset which has been modified into a NLI inference task. Finally, the resulting base model is then used to train a sentence selection model and a claim classification model from data provided by Scifact. Again, we did not devise this approach, but merely following the approach used by MultiVers for more fair comparison between our approaches. Other than this modified training approach for the sentence selection model and claim classification model, the TF-IDF retrieval and final aggregation model remain process remain unchanged.

Like the FEVER dataset, the Scifact dataset measures several different metrics to measure a system's performance. The four metrics is tracks are: Abstract Label Only (A + LO), Sentence Selection (SS), Abstract + Rationalized (A + R), and Sentence Selection + Label (SS + L). The Abstract Label Only metric simple calculates if the predicted label matches the ground truth (ignoring "not enough info" examples). The Abstract + Rationalized metric is very similar to the FEVER metric, namely the correct label is predicts and at least one piece of evidence for the claim is also retrieved as part of the sentence selection. Sentence Selection measures the ability of the model to retrieve the relevant sentence's from the abstracts for a given claim. It is important to note is that the recall here is measured over all the pieces of evidence that can individually verify a claim. In the FEVER metric, if 5 pieces of evidence can independently verify a claim, returning just 1 is enough to get a recall of 1. For this metric, however, returning only piece of evidence, even if it alone can independently verify the claim, will only have a 0.2 recall score for this particular sample. So the key difference is that Abstract + Rationalized and FEVER score only care that at least 1 piece of evidence that can independently verify the claim and the Sentence Selection metric cares that every piece of evidence listed for a claim is returned. Finally, the Sentence Selection + Label metric is essentially the Sentence Selection metric, with the added requirement that the label is correct.

For the Scifact dataset, it appears that the Sentence Selection + Label metric is the primary task. Additionally, for all the tasks, F1 score is generally the preferred measurement for each metric. Below in Table 6.10, we compare against the state of the art on the task, MultiVers, as well as VerT5erini [39] (a T5 based approach similar to [23] on the FEVER task), ARSJoint [63] and the provided baseline approach for the task, VeriSci [55]. As can be seen, our approach fails to be competitive in both the Sentence Selection and Sentence Selection + Label task when compared to MultiVers and VerT5erini. We suspect that this is due to our relatively simple approach to sentence selection, in which we evaluate the relevancy of each sentence independently of each other. Both MultiVers and VerT5erini use the entire abstract as input to determine sentence relevancy. While our approach worked well for the FEVER dataset, scientific abstracts may differ in nature, where previous statements support conclusions in later sentences, which would pose problems for our approach. Despite this, our model achieves the highest F1 on pure label accuracy on the task, outperforming MultiVers. Despite this, it does not perform better on the Abstract + Rationalized metric F1, which is the closest analog to the FEVER score for the FEVER dataset. This is also due to our model's retrieval system, since we implement a threshold to improve F1 on the sentence selection task, some of the relevant sentences for a claim of not returned. Often, the correct sentences are retrieved, but due to the limited context, the scores are low enough to be filtered out. It is worth noting, our approach forms predictions from all 5 retrieved sentences, even if some of the low scoring retrieved sentences are later filtered out.

Table 6.10: Comparison against other top Scifact systems

System	SS + L F1	SS F1	Abstract + R F1	Abstract + LO F1
Wadden et al. (MultiVers) [56]	**67.2**	74.3	**71.1**	72.5
Zhang et al. (ARSJoint) [63]	63.1	**75.8**	65.7	68.1
Pradeep et al. (VerT5erini) [39]	58.8	64.8	62.7	64.9
VeriSci (baseline) [55]	39.5	46.1	46.5	47.4
Ours	58.1	61.6	67.8	**73.2**

Chapter 7

Conclusions

In this work, we built an end-to-end system for the FEVER dataset which attains the 1st best performance on the FEVER score metrics for the dataset and 2nd best score for label accuracy. We have shown that contrary to previous works, graph neural networks do not outperform transformers based models on the task when using similarly-sized models. When comparing our work to other transformer based models, the closest works in terms of [49], [23] use significantly larger models (3x and 10x, respectively) to achieve similar performance to our model. This performance is due to giving special attention to each portion of the process. Whereas many systems reuse the document selection from Athene [18], we built our own and achieve much better document retrieval. Similarly, many systems also use KGAT's [31] selected sentences and focus only on trying to improve label accuracy. By building these components, our final prediction model is not limited by any of the shortcomings of those approaches. Finally, we show that the simple mixed approach, in which the aggregator model has access to both the singleton predictions as well as the concatenated prediction outperforms the other approaches which have been used for transformer based predictions.

7.1 Future Work

In future work we plan to explore improving the sentence selection model's performance. As shown in both FEVER and Scifact, this simple framework yields surprisingly good predictive performance. However, on the Scifact dataset in particular, it was clear that our sentence selection model performed worse on the task when compared to other systems. This work would likely focus on ways to include the appropriate context when needed for selecting sentences, as it because clear that making predictions with only a single sentence of context hurt our performance on Scifact. An area of interest would be avoiding full document processing, which requires much more computation than our approach, and instead attempting to predict which groups of sentences should be processed in conjunction, as usually the entire document was unnecessary for determining which sentences were the most relevant.

Appendix A

Additional Experimental Results

Table A.1: Full combined tf-idf results.

Expt Id	Recall @10	Expt Id	Recall @10	Expt Id	Recall @10	Expt Id	Recall @10
1	75.62 %	2	79.06 %	3	75.00 %	4	78.52 %
5	76.91 %	6	29.06 %	7	76.82 %	8	29.15 %
9	85.93 %	10	86.87 %	11	85.43 %	12	86.28 %
13	85.76 %	14	54.19 %	15	85.60 %	16	53.86 %
17	78.72 %	18	81.04 %	19	78.15 %	20	80.60 %
21	79.37 %	22	32.01 %	23	79.21 %	24	32.15 %
25	86.24 %	26	86.94 %	27	85.77 %	28	86.47 %
29	85.92 %	30	57.40 %	31	85.78 %	32	57.27 %

Table A.2: Full title and document tf-idf results. The "Expt Ids" comlumn represents the title experiment and document experiment respectively.

Expt Ids	Recall @10	Expt Ids	Recall @10	Expt Ids	Recall @10	Expt Ids	Recall @10
1,1	79.01 %	1,2	80.37 %	1,3	79.01 %	1,4	80.30 %
1,5	84.88 %	1,6	78.59 %	1,7	84.82 %	1,8	78.59 %

1,9	80.49 %	1,10	82.22 %	1,11	80.56 %	1,12	82.28 %
1,13	87.32 %	1,14	80.74 %	1,15	87.35 %	1,16	80.78 %
1,17	79.30 %	1,18	80.69 %	1,19	79.25 %	1,20	80.63 %
1,21	85.49 %	1,22	78.86 %	1,23	85.43 %	1,24	78.86 %
1,25	80.87 %	1,26	82.52 %	1,27	80.81 %	1,28	82.40 %
1,29	88.00 %	1,30	81.32 %	1,31	88.01 %	1,32	81.35 %
2,1	78.81 %	2,2	80.25 %	2,3	78.81 %	2,4	80.18 %
2,5	84.78 %	2,6	78.35 %	2,7	84.71 %	2,8	78.35 %
2,9	80.30 %	2,10	82.08 %	2,11	80.36 %	2,12	82.14 %
2,13	87.21 %	2,14	80.57 %	2,15	87.23 %	2,16	80.60 %
2,17	79.07 %	2,18	80.52 %	2,19	79.01 %	2,20	80.46 %
2,21	85.39 %	2,22	78.62 %	2,23	85.32 %	2,24	78.62 %
2,25	80.69 %	2,26	82.35 %	2,27	80.60 %	2,28	82.23 %
2,29	87.88 %	2,30	81.13 %	2,31	87.88 %	2,32	81.14 %
3,1	78.27 %	3,2	79.68 %	3,3	78.18 %	3,4	79.54 %
3,5	84.62 %	3,6	77.90 %	3,7	84.44 %	3,8	77.90 %
3,9	79.89 %	3,10	81.68 %	3,11	79.80 %	3,12	81.60 %
3,13	87.08 %	3,14	80.12 %	3,15	87.02 %	3,16	80.12 %
3,17	78.56 %	3,18	80.00 %	3,19	78.42 %	3,20	79.88 %
3,21	85.19 %	3,22	78.17 %	3,23	85.07 %	3,24	78.18 %
3,25	80.28 %	3,26	82.00 %	3,27	80.06 %	3,28	81.75 %
3,29	87.77 %	3,30	80.78 %	3,31	87.68 %	3,32	80.78 %
4,1	78.07 %	4,2	79.56 %	4,3	77.99 %	4,4	79.42 %
4,5	84.50 %	4,6	77.67 %	4,7	84.32 %	4,8	77.68 %

4,9	79.69 %	4,10	81.55 %	4,11	79.60 %	4,12	81.47 %
4,13	86.97 %	4,14	79.96 %	4,15	86.92 %	4,16	79.95 %
4,17	78.34 %	4,18	79.82 %	4,19	78.20 %	4,20	79.70 %
4,21	85.07 %	4,22	77.95 %	4,23	84.95 %	4,24	77.96 %
4,25	80.09 %	4,26	81.85 %	4,27	79.87 %	4,28	81.60 %
4,29	87.67 %	4,30	80.60 %	4,31	87.59 %	4,32	80.60 %
5,1	50.83 %	5,2	56.64 %	5,3	50.71 %	5,4	56.47 %
5,5	70.84 %	5,6	51.52 %	5,7	70.66 %	5,8	51.52 %
5,9	55.47 %	5,10	62.00 %	5,11	55.41 %	5,12	61.96 %
5,13	76.04 %	5,14	58.15 %	5,15	76.01 %	5,16	58.14 %
5,17	51.19 %	5,18	56.55 %	5,19	51.04 %	5,20	56.42 %
5,21	71.87 %	5,22	52.07 %	5,23	71.75 %	5,24	52.09 %
5,25	55.94 %	5,26	61.92 %	5,27	55.68 %	5,28	61.69 %
5,29	76.90 %	5,30	59.30 %	5,31	76.85 %	5,32	59.29 %
6,1	44.67 %	6,2	51.07 %	6,3	44.49 %	6,4	50.87 %
6,5	67.08 %	6,6	44.46 %	6,7	66.82 %	6,8	44.48 %
6,9	50.20 %	6,10	57.50 %	6,11	50.11 %	6,12	57.36 %
6,13	73.26 %	6,14	52.24 %	6,15	73.20 %	6,16	52.21 %
6,17	45.10 %	6,18	50.90 %	6,19	44.93 %	6,20	50.76 %
6,21	68.22 %	6,22	45.07 %	6,23	68.06 %	6,24	45.11 %
6,25	50.77 %	6,26	57.54 %	6,27	50.50 %	6,28	57.26 %
6,29	74.25 %	6,30	53.53 %	6,31	74.20 %	6,32	53.53 %
7,1	50.85 %	7,2	56.55 %	7,3	50.64 %	7,4	56.30 %
7,5	70.95 %	7,6	51.63 %	7,7	70.76 %	7,8	51.64 %

7,9	55.66 %	7,10	62.08 %	7,11	55.48 %	7,12	61.93 %
7,13	76.19 %	7,14	58.34 %	7,15	76.10 %	7,16	58.30 %
7,17	51.22 %	7,18	56.45 %	7,19	51.01 %	7,20	56.29 %
7,21	71.95 %	7,22	52.15 %	7,23	71.83 %	7,24	52.18 %
7,25	56.14 %	7,26	62.02 %	7,27	55.76 %	7,28	61.69 %
7,29	77.03 %	7,30	59.46 %	7,31	76.92 %	7,32	59.43 %
8,1	44.63 %	8,2	51.08 %	8,3	44.37 %	8,4	50.83 %
8,5	67.15 %	8,6	44.54 %	8,7	66.88 %	8,8	44.56 %
8,9	50.21 %	8,10	57.47 %	8,11	50.01 %	8,12	57.24 %
8,13	73.42 %	8,14	52.40 %	8,15	73.33 %	8,16	52.38 %
8,17	45.05 %	8,18	50.89 %	8,19	44.81 %	8,20	50.71 %
8,21	68.29 %	8,22	45.19 %	8,23	68.11 %	8,24	45.23 %
8,25	50.80 %	8,26	57.56 %	8,27	50.43 %	8,28	57.21 %
8,29	74.35 %	8,30	53.66 %	8,31	74.25 %	8,32	53.66 %
9,1	80.20 %	9,2	82.13 %	9,3	80.21 %	9,4	82.03 %
9,5	86.35 %	9,6	79.99 %	9,7	86.27 %	9,8	80.00 %
9,9	81.20 %	9,10	83.08 %	9,11	81.29 %	9,12	83.14 %
9,13	87.74 %	9,14	81.85 %	9,15	87.76 %	9,16	81.87 %
9,17	80.30 %	9,18	82.03 %	9,19	80.27 %	9,20	81.95 %
9,21	86.71 %	9,22	80.24 %	9,23	86.63 %	9,24	80.24 %
9,25	81.52 %	9,26	83.09 %	9,27	81.47 %	9,28	83.00 %
9,29	88.31 %	9,30	82.33 %	9,31	88.32 %	9,32	82.35 %
10,1	80.07 %	10,2	82.04 %	10,3	80.06 %	10,4	81.93 %
10,5	86.33 %	10,6	79.86 %	10,7	86.24 %	10,8	79.87 %

10,9	81.11 %	10,10	82.98 %	10,11	81.17 %	10,12	83.04 %
10,13	87.71 %	10,14	81.76 %	10,15	87.72 %	10,16	81.77 %
10,17	80.18 %	10,18	81.93 %	10,19	80.15 %	10,20	81.85 %
10,21	86.68 %	10,22	80.12 %	10,23	86.59 %	10,24	80.12 %
10,25	81.44 %	10,26	83.01 %	10,27	81.38 %	10,28	82.91 %
10,29	88.27 %	10,30	82.24 %	10,31	88.27 %	10,32	82.25 %
11,1	79.70 %	11,2	81.62 %	11,3	79.59 %	11,4	81.45 %
11,5	86.13 %	11,6	79.47 %	11,7	85.94 %	11,8	79.49 %
11,9	80.77 %	11,10	82.68 %	11,11	80.68 %	11,12	82.58 %
11,13	87.53 %	11,14	81.37 %	11,15	87.47 %	11,16	81.35 %
11,17	79.79 %	11,18	81.49 %	11,19	79.65 %	11,20	81.35 %
11,21	86.45 %	11,22	79.74 %	11,23	86.30 %	11,24	79.76 %
11,25	81.07 %	11,26	82.70 %	11,27	80.85 %	11,28	82.45 %
11,29	88.13 %	11,30	81.92 %	11,31	88.05 %	11,32	81.92 %
12,1	79.58 %	12,2	81.52 %	12,3	79.46 %	12,4	81.35 %
12,5	86.10 %	12,6	79.37 %	12,7	85.91 %	12,8	79.38 %
12,9	80.72 %	12,10	82.64 %	12,11	80.62 %	12,12	82.53 %
12,13	87.52 %	12,14	81.30 %	12,15	87.46 %	12,16	81.28 %
12,17	79.68 %	12,18	81.41 %	12,19	79.55 %	12,20	81.27 %
12,21	86.42 %	12,22	79.64 %	12,23	86.27 %	12,24	79.66 %
12,25	81.02 %	12,26	82.66 %	12,27	80.80 %	12,28	82.41 %
12,29	88.10 %	12,30	81.85 %	12,31	88.02 %	12,32	81.85 %
13,1	56.58 %	13,2	62.16 %	13,3	56.40 %	13,4	61.90 %
13,5	75.00 %	13,6	56.84 %	13,7	74.78 %	13,8	56.84 %

13,9	59.37 %	13,10	65.16 %	13,11	59.28 %	13,12	65.10 %
13,13	78.35 %	13,14	62.50 %	13,15	78.30 %	13,16	62.45 %
13,17	56.80 %	13,18	61.75 %	13,19	56.64 %	13,20	61.59 %
13,21	75.58 %	13,22	57.50 %	13,23	75.42 %	13,24	57.52 %
13,25	59.74 %	13,26	64.87 %	13,27	59.51 %	13,28	64.61 %
13,29	78.94 %	13,30	63.52 %	13,31	78.92 %	13,32	63.49 %
14,1	55.54 %	14,2	61.21 %	14,3	55.37 %	14,4	60.94 %
14,5	74.43 %	14,6	55.66 %	14,7	74.21 %	14,8	55.66 %
14,9	58.42 %	14,10	64.33 %	14,11	58.33 %	14,12	64.28 %
14,13	78.01 %	14,14	61.48 %	14,15	77.96 %	14,16	61.44 %
14,17	55.75 %	14,18	60.83 %	14,19	55.58 %	14,20	60.67 %
14,21	75.07 %	14,22	56.35 %	14,23	74.90 %	14,24	56.36 %
14,25	58.81 %	14,26	64.07 %	14,27	58.60 %	14,28	63.83 %
14,29	78.62 %	14,30	62.57 %	14,31	78.61 %	14,32	62.53 %
15,1	56.43 %	15,2	61.94 %	15,3	56.22 %	15,4	61.66 %
15,5	74.86 %	15,6	56.63 %	15,7	74.63 %	15,8	56.66 %
15,9	59.37 %	15,10	65.13 %	15,11	59.17 %	15,12	64.99 %
15,13	78.33 %	15,14	62.23 %	15,15	78.26 %	15,16	62.18 %
15,17	56.60 %	15,18	61.47 %	15,19	56.42 %	15,20	61.28 %
15,21	75.44 %	15,22	57.28 %	15,23	75.29 %	15,24	57.31 %
15,25	59.73 %	15,26	64.81 %	15,27	59.42 %	15,28	64.48 %
15,29	78.91 %	15,30	63.27 %	15,31	78.86 %	15,32	63.24 %
16,1	55.43 %	16,2	61.17 %	16,3	55.22 %	16,4	60.88 %
16,5	74.41 %	16,6	55.62 %	16,7	74.18 %	16,8	55.65 %

16,9	58.47 %	16,10	64.39 %	16,11	58.25 %	16,12	64.24 %
16,13	77.99 %	16,14	61.39 %	16,15	77.93 %	16,16	61.35 %
16,17	55.71 %	16,18	60.73 %	16,19	55.50 %	16,20	60.52 %
16,21	75.03 %	16,22	56.30 %	16,23	74.87 %	16,24	56.34 %
16,25	58.86 %	16,26	64.09 %	16,27	58.54 %	16,28	63.76 %
16,29	78.62 %	16,30	62.46 %	16,31	78.57 %	16,32	62.44 %
17,1	77.90 %	17,2	79.30 %	17,3	77.90 %	17,4	79.26 %
17,5	84.23 %	17,6	77.13 %	17,7	84.20 %	17,8	77.14 %
17,9	79.49 %	17,10	81.27 %	17,11	79.55 %	17,12	81.33 %
17,13	86.78 %	17,14	79.55 %	17,15	86.81 %	17,16	79.60 %
17,17	77.99 %	17,18	79.49 %	17,19	77.93 %	17,20	79.44 %
17,21	84.25 %	17,22	77.37 %	17,23	84.22 %	17,24	77.37 %
17,25	79.77 %	17,26	81.27 %	17,27	79.69 %	17,28	81.20 %
17,29	86.93 %	17,30	79.97 %	17,31	86.96 %	17,32	80.00 %
18,1	77.74 %	18,2	79.18 %	18,3	77.74 %	18,4	79.13 %
18,5	84.17 %	18,6	76.96 %	18,7	84.13 %	18,8	76.97 %
18,9	79.37 %	18,10	81.17 %	18,11	79.42 %	18,12	81.23 %
18,13	86.71 %	18,14	79.42 %	18,15	86.73 %	18,16	79.46 %
18,17	77.81 %	18,18	79.34 %	18,19	77.75 %	18,20	79.28 %
18,21	84.19 %	18,22	77.19 %	18,23	84.15 %	18,24	77.19 %
18,25	79.64 %	18,26	81.16 %	18,27	79.55 %	18,28	81.08 %
18,29	86.87 %	18,30	79.83 %	18,31	86.88 %	18,32	79.85 %
19,1	77.32 %	19,2	78.77 %	19,3	77.24 %	19,4	78.68 %
19,5	84.11 %	19,6	76.60 %	19,7	83.95 %	19,8	76.61 %

19,9	79.05 %	19,10	80.94 %	19,11	78.96 %	19,12	80.85 %
19,13	86.66 %	19,14	79.08 %	19,15	86.60 %	19,16	79.09 %
19,17	77.46 %	19,18	78.98 %	19,19	77.32 %	19,20	78.87 %
19,21	84.08 %	19,22	76.82 %	19,23	83.97 %	19,24	76.83 %
19,25	79.34 %	19,26	80.97 %	19,27	79.13 %	19,28	80.76 %
19,29	86.82 %	19,30	79.55 %	19,31	86.75 %	19,32	79.55 %
20,1	77.24 %	20,2	78.72 %	20,3	77.15 %	20,4	78.62 %
20,5	84.08 %	20,6	76.48 %	20,7	83.92 %	20,8	76.49 %
20,9	78.99 %	20,10	80.90 %	20,11	78.90 %	20,12	80.81 %
20,13	86.63 %	20,14	78.99 %	20,15	86.57 %	20,16	79.00 %
20,17	77.36 %	20,18	78.91 %	20,19	77.22 %	20,20	78.80 %
20,21	84.04 %	20,22	76.70 %	20,23	83.93 %	20,24	76.71 %
20,25	79.28 %	20,26	80.93 %	20,27	79.06 %	20,28	80.72 %
20,29	86.79 %	20,30	79.44 %	20,31	86.72 %	20,32	79.45 %
21,1	47.76 %	21,2	53.65 %	21,3	47.58 %	21,4	53.47 %
21,5	68.77 %	21,6	47.54 %	21,7	68.53 %	21,8	47.55 %
21,9	52.70 %	21,10	59.33 %	21,11	52.66 %	21,12	59.25 %
21,13	74.43 %	21,14	54.88 %	21,15	74.38 %	21,16	54.85 %
21,17	48.08 %	21,18	53.40 %	21,19	47.91 %	21,20	53.25 %
21,21	69.48 %	21,22	48.13 %	21,23	69.29 %	21,24	48.17 %
21,25	53.20 %	21,26	59.17 %	21,27	52.99 %	21,28	58.90 %
21,29	75.00 %	21,30	55.90 %	21,31	74.89 %	21,32	55.87 %
22,1	44.94 %	22,2	51.03 %	22,3	44.71 %	22,4	50.82 %
22,5	66.94 %	22,6	44.10 %	22,7	66.66 %	22,8	44.11 %

22,9	50.38 %	22,10	57.26 %	22,11	50.31 %	22,12	57.13 %
22,13	73.15 %	22,14	51.90 %	22,15	73.05 %	22,16	51.85 %
22,17	45.21 %	22,18	50.60 %	22,19	45.00 %	22,20	50.44 %
22,21	67.71 %	22,22	44.71 %	22,23	67.51 %	22,24	44.74 %
22,25	50.81 %	22,26	57.05 %	22,27	50.56 %	22,28	56.75 %
22,29	73.71 %	22,30	53.02 %	22,31	73.60 %	22,32	52.98 %
23,1	48.08 %	23,2	53.86 %	23,3	47.86 %	23,4	53.63 %
23,5	69.03 %	23,6	47.95 %	23,7	68.80 %	23,8	47.97 %
23,9	53.08 %	23,10	59.68 %	23,11	52.92 %	23,12	59.49 %
23,13	74.77 %	23,14	55.21 %	23,15	74.68 %	23,16	55.17 %
23,17	48.40 %	23,18	53.64 %	23,19	48.20 %	23,20	53.47 %
23,21	69.74 %	23,22	48.56 %	23,23	69.57 %	23,24	48.60 %
23,25	53.54 %	23,26	59.51 %	23,27	53.20 %	23,28	59.11 %
23,29	75.29 %	23,30	56.17 %	23,31	75.15 %	23,32	56.16 %
24,1	45.28 %	24,2	51.41 %	24,3	45.03 %	24,4	51.18 %
24,5	67.23 %	24,6	44.58 %	24,7	66.94 %	24,8	44.61 %
24,9	50.68 %	24,10	57.61 %	24,11	50.51 %	24,12	57.40 %
24,13	73.43 %	24,14	52.45 %	24,15	73.33 %	24,16	52.41 %
24,17	45.54 %	24,18	51.07 %	24,19	45.29 %	24,20	50.90 %
24,21	67.99 %	24,22	45.24 %	24,23	67.80 %	24,24	45.27 %
24,25	51.12 %	24,26	57.46 %	24,27	50.77 %	24,28	57.08 %
24,29	73.98 %	24,30	53.55 %	24,31	73.86 %	24,32	53.53 %
25,1	81.50 %	25,2	83.04 %	25,3	81.50 %	25,4	82.97 %
25,5	87.18 %	25,6	81.14 %	25,7	87.11 %	25,8	81.16 %

25,9	82.39 %	25,10	83.95 %	25,11	82.47 %	25,12	84.00 %
25,13	88.43 %	25,14	82.97 %	25,15	88.46 %	25,16	82.99 %
25,17	81.47 %	25,18	82.95 %	25,19	81.44 %	25,20	82.89 %
25,21	87.10 %	25,22	81.35 %	25,23	87.02 %	25,24	81.35 %
25,25	82.53 %	25,26	83.75 %	25,27	82.46 %	25,28	83.66 %
25,29	88.36 %	25,30	83.22 %	25,31	88.37 %	25,32	83.24 %
26,1	81.43 %	26,2	83.02 %	26,3	81.43 %	26,4	82.94 %
26,5	87.15 %	26,6	81.07 %	26,7	87.08 %	26,8	81.08 %
26,9	82.33 %	26,10	83.90 %	26,11	82.41 %	26,12	83.96 %
26,13	88.40 %	26,14	82.90 %	26,15	88.43 %	26,16	82.92 %
26,17	81.39 %	26,18	82.91 %	26,19	81.37 %	26,20	82.85 %
26,21	87.07 %	26,22	81.29 %	26,23	86.99 %	26,24	81.29 %
26,25	82.47 %	26,26	83.69 %	26,27	82.40 %	26,28	83.62 %
26,29	88.32 %	26,30	83.17 %	26,31	88.34 %	26,32	83.19 %
27,1	80.99 %	27,2	82.52 %	27,3	80.88 %	27,4	82.40 %
27,5	86.97 %	27,6	80.66 %	27,7	86.78 %	27,8	80.67 %
27,9	81.99 %	27,10	83.63 %	27,11	81.89 %	27,12	83.51 %
27,13	88.24 %	27,14	82.51 %	27,15	88.18 %	27,16	82.49 %
27,17	80.96 %	27,18	82.46 %	27,19	80.82 %	27,20	82.34 %
27,21	86.85 %	27,22	80.89 %	27,23	86.70 %	27,24	80.91 %
27,25	82.12 %	27,26	83.40 %	27,27	81.88 %	27,28	83.18 %
27,29	88.16 %	27,30	82.83 %	27,31	88.10 %	27,32	82.83 %
28,1	80.92 %	28,2	82.48 %	28,3	80.81 %	28,4	82.36 %
28,5	86.90 %	28,6	80.57 %	28,7	86.72 %	28,8	80.60 %

28,9	81.92 %	28,10	83.56 %	28,11	81.82 %	28,12	83.45 %
28,13	88.20 %	28,14	82.46 %	28,15	88.14 %	28,16	82.43 %
28,17	80.88 %	28,18	82.39 %	28,19	80.75 %	28,20	82.27 %
28,21	86.79 %	28,22	80.81 %	28,23	86.64 %	28,24	80.84 %
28,25	82.05 %	28,26	83.34 %	28,27	81.81 %	28,28	83.12 %
28,29	88.13 %	28,30	82.78 %	28,31	88.06 %	28,32	82.78 %
29,1	58.02 %	29,2	63.42 %	29,3	57.87 %	29,4	63.20 %
29,5	75.91 %	29,6	58.23 %	29,7	75.75 %	29,8	58.25 %
29,9	60.79 %	29,10	66.13 %	29,11	60.75 %	29,12	66.12 %
29,13	79.19 %	29,14	63.79 %	29,15	79.16 %	29,16	63.76 %
29,17	58.15 %	29,18	62.99 %	29,19	58.03 %	29,20	62.88 %
29,21	76.23 %	29,22	58.85 %	29,23	76.09 %	29,24	58.89 %
29,25	61.03 %	29,26	65.74 %	29,27	60.85 %	29,28	65.55 %
29,29	79.40 %	29,30	64.69 %	29,31	79.36 %	29,32	64.68 %
30,1	56.94 %	30,2	62.58 %	30,3	56.79 %	30,4	62.35 %
30,5	75.45 %	30,6	57.11 %	30,7	75.24 %	30,8	57.13 %
30,9	59.83 %	30,10	65.38 %	30,11	59.79 %	30,12	65.35 %
30,13	78.66 %	30,14	62.83 %	30,15	78.61 %	30,16	62.79 %
30,17	57.14 %	30,18	62.11 %	30,19	57.04 %	30,20	61.99 %
30,21	75.72 %	30,22	57.73 %	30,23	75.58 %	30,24	57.76 %
30,25	60.09 %	30,26	65.08 %	30,27	59.89 %	30,28	64.86 %
30,29	78.91 %	30,30	63.80 %	30,31	78.85 %	30,32	63.76 %
31,1	58.03 %	31,2	63.40 %	31,3	57.85 %	31,4	63.15 %
31,5	75.96 %	31,6	58.34 %	31,7	75.76 %	31,8	58.36 %

31,9	60.77 %	31,10	66.14 %	31,11	60.61 %	31,12	66.02 %
31,13	79.17 %	31,14	63.80 %	31,15	79.11 %	31,16	63.78 %
31,17	58.15 %	31,18	62.99 %	31,19	57.99 %	31,20	62.86 %
31,21	76.18 %	31,22	58.90 %	31,23	76.04 %	31,24	58.93 %
31,25	61.01 %	31,26	65.82 %	31,27	60.73 %	31,28	65.53 %
31,29	79.37 %	31,30	64.65 %	31,31	79.32 %	31,32	64.64 %
32,1	57.19 %	32,2	62.74 %	32,3	57.01 %	32,4	62.49 %
32,5	75.40 %	32,6	57.38 %	32,7	75.20 %	32,8	57.40 %
32,9	60.01 %	32,10	65.51 %	32,11	59.85 %	32,12	65.38 %
32,13	78.69 %	32,14	63.01 %	32,15	78.62 %	32,16	62.98 %
32,17	57.32 %	32,18	62.25 %	32,19	57.16 %	32,20	62.11 %
32,21	75.66 %	32,22	57.99 %	32,23	75.51 %	32,24	58.02 %
32,25	60.28 %	32,26	65.18 %	32,27	60.00 %	32,28	64.91 %
32,29	78.92 %	32,30	63.89 %	32,31	78.87 %	32,32	63.89 %

Bibliography

[1] Fake news challenge. http://www.fakenewschallenge.org/. Accessed 29-October-2022. 1

[2] Pytorch lightning. https://www.pytorchlightning.ai". 5.2

[3] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. 2.3

[4] Verb Bergengruen and Billy Perrigo. Facebook acted too late to tackle misinformation on 2020 election, report finds. https://time.com/5949210/facebook-misinformation-2020-election-report/. Accessed 29-October-2022. 1

[5] Qian Chen, Xiaodan Zhu, Zhen-Hua Ling, Si Wei, Hui Jiang, and Diana Inkpen. Enhanced LSTM for natural language inference. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1657–1668, Vancouver, Canada, July 2017. Association for Computational Linguistics. 3.1

[6] Tianqi Chen and Carlos Guestrin. XGBoost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, pages 785–794, New York, NY, USA, 2016. ACM. 5.4

[7] Kyunghyun Cho, Bart van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. On the properties of neural machine translation: Encoder–decoder approaches. In *Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*, pages 103–111, Doha, Qatar, October 2014. Association for Computational Linguistics. 2.3

[8] Mitchell DeHaven. Bevers: A general, simple, and performant framework for automatic fact verification. Master's thesis, University of Nebraska-Lincoln, Lincoln, NE, November 2022.

[9] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. 1, 2.3

[10] IBM Cloud Education. What are deep neural networks? `https://www.ibm.com/cloud/learn/neural-networks`. (document), 2.1

[11] Facebook. Facebook launches accelerator challenge for global fact-checkers to expand reach of reliable information, 2021. `https://www.facebook.com/formedia/blog/third-party-fact-checking-how-it-works` Accessed 29-October-2022. 1

[12] Facebook. Facebook launches accelerator challenge for global fact-checkers to expand reach of reliable information, 2021. `https://www.facebook.com/formedia/blog/accelerator-fact-checkers` Accessed 29-October-2022. 1

[13] Wikimedia Foundation. Wikimedia api portal. 3.1, 4.1

[14] Yoav Freund and Robert E. Schapire. A desicion-theoretic generalization of on-line learning and an application to boosting. In Paul Vitányi, editor, *Computational Learning Theory*, pages 23–37, Berlin, Heidelberg, 1995. Springer Berlin Heidelberg. 2.4

[15] Jerome H. Friedman. Greedy function approximation: A gradient boosting machine. *The Annals of Statistics*, 29(5):1189 – 1232, 2001. 2.4

[16] Yoav Goldberg. A primer on neural network models for natural language processing. 2.3

[17] Nir Grinberg, Kenneth Joseph, Lisa Friedland, Briony Swire-Thompson, and David Lazer. Fake news on twitter during the 2016 u.s. presidential election. *Science*, 363(6425):374–378, 2019. 1

[18] Andreas Hanselowski, Hao Zhang, Zile Li, Daniil Sorokin, Benjamin Schiller, Claudia Schulz, and Iryna Gurevych. UKP-athene: Multi-sentence textual entailment for claim verification. In *Proceedings of the First Workshop on Fact Extraction and VERification (FEVER)*, pages 103–108, Brussels, Belgium, November 2018. Association for Computational Linguistics. 3.1, 3.3, 4.1, 6.1.1, 6.3, 6.6, 6.8, 7

[19] Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. Deberta: Decoding-enhanced bert with disentangled attention. In *International Conference on Learning Representations*, 2021. 3.2, 6.3

[20] Richard D Hipp. SQLite. 4.1

[21] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997. 2.3

[22] Philip N. Howard, Bharath Ganesh, Dimitra Liotsiou, John Kelly, and Camille François. *The IRA, Social Media and Political Polarization in the United States, 2012-2018*. Project on Computational Propaganda, 2018. 1

[23] Kelvin Jiang, Ronak Pradeep, and Jimmy Lin. Exploring listwise evidence reasoning with t5 for fact verification. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 402–410, Online, August 2021. Association for Computational Linguistics. 1, 3.2, 6.1.1, 6.3, 6.6, 6.2, 6.8, 6.4, 7

[24] Qiao Jin, Bhuwan Dhingra, Zhengping Liu, William Cohen, and Xinghua Lu. PubMedQA: A dataset for biomedical research question answering. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2567–2577, Hong Kong, China, November 2019. Association for Computational Linguistics. 6.4

[25] Karen Spärck Jones. A statistical interpretation of term specificity and its application in retrieval. *Journal of Documentation*, 28:11–22, 1972. 2.2

[26] Yoon Kim. 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 2014. Association for Computational Linguistics. 2.3

[27] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. 2.3

[28] Patrick Lewis, Myle Ott, Jingfei Du, and Veselin Stoyanov. Pretrained language models for biomedical and clinical tasks: Understanding and extending the state-of-

the-art. In *Proceedings of the 3rd Clinical Natural Language Processing Workshop*, pages 146–157, Online, November 2020. Association for Computational Linguistics. 6.4

[29] Richard Liaw, Eric Liang, Robert Nishihara, Philipp Moritz, Joseph E Gonzalez, and Ion Stoica. Tune: A research platform for distributed model selection and training. *arXiv preprint arXiv:1807.05118*, 2018. 5.2

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

[31] Zhenghao Liu, Chenyan Xiong, Maosong Sun, and Zhiyuan Liu. Fine-grained fact verification with kernel graph attention network. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7342–7351, Online, July 2020. Association for Computational Linguistics. 3.3, 6.6, 6.2, 6.8, 6.9, 7

[32] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to Information Retrieval*. Cambridge University Press, Cambridge, UK, 2008. 2.1, 2.2, 4.1

[33] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In C.J. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 26. Curran Associates, Inc., 2013. 2.3

[34] Rafael Müller, Simon Kornblith, and Geoffrey Hinton. *When Does Label Smoothing Help?* Curran Associates Inc., Red Hook, NY, USA, 2019. 6.2

[35] Yixin Nie, Haonan Chen, and Mohit Bansal. Combining fact extraction and verification with neural semantic matching networks. In *Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence and Thirty-First Innovative Applications of Artificial*

Intelligence Conference and Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, AAAI'19/IAAI'19/EAAI'19. AAAI Press, 2019. 3.1, 6.3, 6.6, 6.8

[36] Christopher Olah. Conv nets: A modular perspective. `https://colah.github.io/posts/2014-07-Conv-Nets-Modular/`. (document), 2.3

[37] Christopher Olah. Understanding lstm networks. `https://colah.github.io/posts/2015-08-Understanding-LSTMs/`. (document), 2.2

[38] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011. 4.1

[39] Ronak Pradeep, Xueguang Ma, Rodrigo Nogueira, and Jimmy Lin. Scientific claim verification with VerT5erini. In *Proceedings of the 12th International Workshop on Health Text Mining and Information Analysis*, pages 94–103, online, April 2021. Association for Computational Linguistics. 6.4, 6.10

[40] Associated Press. Ap expands access to factual information with new twitter collaboration, 2021. `https://www.ap.org/press-releases/2021/ap-expands-access-to-factual-information-with-new-twitter-collaboration` Accessed 29-October-2022. 1

[41] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21(1), jan 2020. 3.2, 6.3

[42] Jon Roozenbeek, Claudia R Schneider, Sarah Dryhurst, John Kerr, Alexandra LJ Freeman, Gabriel Recchia, Anne Marthe Van Der Bles, and Sander Van Der Linden.

Susceptibility to misinformation about covid-19 around the world. *Royal Society open science*, 7(10):201199, 2020. 1

[43] Benjamin Sanchez-Lengeling, Emily Reif, Adam Pearce, and Alexander B. Wiltschko. A gentle introduction to graph neural networks. *Distill*, 2021. https://distill.pub/2021/gnn-intro. (document), 3.1

[44] SCic. MS Windows NT kernel description. hhttps://scikit-learn.org/stable/auto_examples/tree/plot_iris_dtc.html. Accessed: 2010-09-30. (document), 2.7

[45] Gautam Kishore Shahi, Anne Dirkson, and Tim A Majchrzak. An exploratory study of covid-19 misinformation on twitter. *Online social networks and media*, 22:100104, 2021. 1

[46] Ravid Shwartz-Ziv and Amitai Armon. Tabular data: Deep learning is not all you need. *Information Fusion*, 81:84–90, 2022. 4.3.2

[47] Snoopes. Transparency. https://www.snopes.com/transparency/. Accessed 29-October-2022. 1

[48] Amir Soleimani, Christof Monz, and Marcel Worring. Bert for evidence retrieval and claim verification. In *Advances in Information Retrieval: 42nd European Conference on IR Research, ECIR 2020, Lisbon, Portugal, April 14–17, 2020, Proceedings, Part II*, page 359–366, Berlin, Heidelberg, 2020. Springer-Verlag. 3.2, 3.3, 6.3, 6.6, 6.8

[49] Dominik Stammbach. Evidence selection as a token-level prediction task. In *Proceedings of the Fourth Workshop on Fact Extraction and VERification (FEVER)*, pages 14–20, Dominican Republic, November 2021. Association for Computational Linguistics. 1, 3.2, 4.2.1, 6.6, 6.2, 6.8, 7

[50] Shyam Subramanian and Kyumin Lee. Hierarchical Evidence Set Modeling for automated fact extraction and verification. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7798–7809, Online, November 2020. Association for Computational Linguistics. 6.3, 6.6, 6.8

[51] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2818–2826, 2016. 5.2

[52] James Thorne, Andreas Vlachos, Christos Christodoulopoulos, and Arpit Mittal. FEVER: a large-scale dataset for fact extraction and VERification. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 809–819, New Orleans, Louisiana, June 2018. Association for Computational Linguistics. 1, 2.1, 6.3

[53] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. (document), 1, 2.4, 2.3, 2.5, 2.6

[54] Nguyen Vo and Kyumin Lee. Where are the facts? searching for fact-checked information to alleviate the spread of fake news. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7717–7731, Online, November 2020. Association for Computational Linguistics. 1

[55] David Wadden, Shanchuan Lin, Kyle Lo, Lucy Lu Wang, Madeleine van Zuylen, Arman Cohan, and Hannaneh Hajishirzi. Fact or fiction: Verifying scientific claims. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*

(EMNLP), pages 7534–7550, Online, November 2020. Association for Computational Linguistics. 6.4, 6.10

[56] David Wadden, Kyle Lo, Lucy Wang, Arman Cohan, Iz Beltagy, and Hannaneh Hajishirzi. MultiVerS: Improving scientific claim verification with weak supervision and full-document context. In *Findings of the Association for Computational Linguistics: NAACL 2022*, pages 61–76, Seattle, United States, July 2022. Association for Computational Linguistics. 6.4, 6.10

[57] Adina Williams, Nikita Nangia, and Samuel Bowman. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1112–1122, New Orleans, Louisiana, June 2018. Association for Computational Linguistics. 3.2, 5.3

[58] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. Huggingface's transformers: State-of-the-art natural language processing, 2019. 4.3.1, 5.2

[59] Deming Ye, Yankai Lin, Jiaju Du, Zhenghao Liu, Peng Li, Maosong Sun, and Zhiyuan Liu. Coreferential Reasoning Learning for Language Representation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7170–7186, Online, November 2020. Association for Computational Linguistics. 6.9

[60] Takuma Yoneda, Jeff Mitchell, Johannes Welbl, Pontus Stenetorp, and Sebastian Riedel. UCL machine reading group: Four factor framework for fact finding (HexaF). In *Proceedings of the First Workshop on Fact Extraction and VERification (FEVER)*, pages 97–102, Brussels, Belgium, November 2018. Association for Computational Linguistics. 3.1, 6.6, 6.8

[61] Manzil Zaheer, Guru Guruganesh, Kumar Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontanon, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, et al. Big bird: Transformers for longer sequences. *Advances in Neural Information Processing Systems*, 33, 2020. 3.2

[62] Ye Zhang and Byron Wallace. A sensitivity analysis of (and practitioners' guide to) convolutional neural networks for sentence classification. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 253–263, Taipei, Taiwan, November 2017. Asian Federation of Natural Language Processing. 2.3

[63] Zhiwei Zhang, Jiyi Li, Fumiyo Fukumoto, and Yanming Ye. Abstract, rationale, stance: A joint model for scientific claim verification. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 3580–3586, Online and Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics. 6.4, 6.10

[64] Wanjun Zhong, Jingjing Xu, Duyu Tang, Zenan Xu, Nan Duan, Ming Zhou, Jiahai Wang, and Jian Yin. Reasoning over semantic-level graph for fact checking. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6170–6180, Online, July 2020. Association for Computational Linguistics. 6.6, 6.8

[65] Jie Zhou, Xu Han, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li, and Maosong Sun. GEAR: Graph-based evidence aggregating and reasoning for fact verification. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 892–901, Florence, Italy, July 2019. Association for Computational Linguistics. 3.3, 6.3, 6.6, 6.8