Factual Error Correction of Claims from Wikipedia by Question Answering

Øystein Place

Postdoc Fellow Ghazaal Sheikhi

A Master's Thesis
Assignment for INFO390
Department of Information Science and Media Studies
University of Bergen

May 31, 2023

Abstract

In this thesis, we identify a need for improved automated tools for fact-checking due to an increase in disinformation and misinformation. Furthermore, to create trust in the tools created we identify a need for these tools to incorporate some explainability in their determinations. As such we propose in this project "question answering based factual error correction" using a novel approach. Applying question generation on claims in tandem with question answering using the claims and provided evidence as context, we are able to both verify and correct claims. The generated corrections provide us with explainability and in conjunction with a verification model we can verify both the original and corrected claims. Our model is shown to be able to reduce refuted claims verified in the FEVER dev data set by 50% using this approach when compared to resulting verification using no corrections, while also having little effect on supported claims. This thesis also investigates what makes for high-quality questions in the fact-checking and correction domain using feature analysis and selection. We show that there exists no one feature that is the most important in making high-quality questions, but rather a combination of features, including downstream task features.

Contents

List of Figures

Chapter 1

Introduction

1.1 Motivation

Fact-checking in terms of news verification has been a recurring issue for a long time. Especially after the 2016 American election where the phrase "Fake News" hit the mainstream, and bipartisan trust in the media dropped in the following years (*Brenan*, 2021; *Gottfried and Liedke*, 2021). In addition, the 2019 covid-19 epidemic put readers in a difficult position of having to sort between differing medical information given by news outlets and found on social media. Some of the misinformation could be attributed to news sources being too eager to be the first to publish a story and thus not doing their due diligence in terms of internal fact-checking (*Guo et al.*, 2021), or it could be a lack of understanding of the topics themselves. To combat different types of misinformation, there has been an increase of fact-checking outlets around the world that focus on external fact-checking (*Stencel*, 2019). This refers to investigating already published news, and news/information gaining traction on social media. This is however a labor-intensive task, and there is more misinformation than fact-checkers.

To alleviate some of this burden on human fact-checkers, machine learning and natural language processing have gained more relevance within the domain (*Guo et al.*, 2021). By using machines to help validate facts this could speed up the process and help fact-checkers keep up with the growing amount of misinformation. In addition, such machines could also be used for internal fact-checking which could in theory stop some misinformation from ever reaching the public eye. As seen during the January 6th. United States Capitol riot the effect of misinformation in combination with malicious actors can have detrimental real-world effects.

With all this in mind, the motivation for doing this research is that there is a need, now more than ever, to find better ways of performing fact-checking. The potential harms resulting from misinformation will likely not dissipate over time. Especially as nations start to employ more targeted malicious disinformation as a weaponized tool against other nations. This might result in the destabilization of political discourse, or as seen in the US. in the form of attempted election interference, as Russia tried during the 2016 American election according to the U.S. intelligence community (*Myre*, 2021).

1.2 Problem Statement

Within the field of fact-checking, we have identified a need for better automation tools to keep up with the increase of misinformation spreading in news media and on social media. Current such approaches focus on addressing the automated fact-checking problem as a classification problem in Machine Learning(ML). However, for this project, the goal is to extend this task to do factual error correction based on question answering. Question answering being a technique in ML/natural language processing (NLP), and a relatively new approach for this type of problem. Factual error correction has been studied in the context of neural abstractive summarization systems to ensure the factual consistency of the generated summaries. However, it has not been applied in the domain of automated fact-checking.

The main research questions of this project is:

- **R1**: "Can an unsupervised question answering system be used for factual error correction of claims?"
- **R2**: "Is it possible to generate high quality factoid questions from claims?
- **R3**: "Does question-answering lead to better verification results?
- **R4**: "Does question-answering lead to better error correction scores compared to transformer-based distance learning?

1.3 Contribution

On completion of the work in this project, there are mainly three contributions made as a result. The first contribution and main goal is the expansion of knowledge within the domain of explainable fact-checking which is accomplished in the form of the pipeline for factual error correction created as a result of the work. The second contribution comes in the form of a broader investigation and understanding of question generation and question answering as it relates to factual error correction. Understanding how the questions we ask and how they are answered is important when working in this domain. Lastly, a simple classification model for identifying well-formed questions will be created, and tested. The results of this model will also broaden our understanding of what a well-formed question is and how downstream tasks are affected within the domain of factual error correction.

1.4 Thesis outline

In this thesis, I will first present a look at some base concepts relating to the work followed by a literature review of important concepts in automated fact-checking, and give an explanation of current approaches to factual error correction in section 2. Following this I will mention the methodology of the work that is going to be done, in this case, design science, in section 3. In section 4, I will discuss what I will be doing in my work, and how I will be evaluating my expected results. Finally, I will discuss my

results, in section 5. The code produced as a result of my work in this project can be found here:

- Google Drive Folder

Chapter 2

Background

2.1 Base Concepts

The upcoming sections will cover some base concepts relating to technology which is beneficial to have an understanding of for the related research (2.2) and method (4) parts of the paper.

2.1.1 Sequence to Sequence Models

Within the domain of machine learning, there exist many different models that can be employed for Natural Language Processing tasks. Deep Neural Networks work well for many tasks within the domain, but for mapping sequences to each other another option was needed. This is where Sequence to Sequence models (seq2seq) come into play (*Sutskever et al.*, 2014).

Seq2seq models work on tasks such as translation from English to French, question answering, and tasks in general where varying length sequences of text are given as input and a new sequence of text is produced. These types of models are precursors to more complex models which the work in this project employs such as transformer models 2.1.2 and text-to-text transformers models 2.1.3.

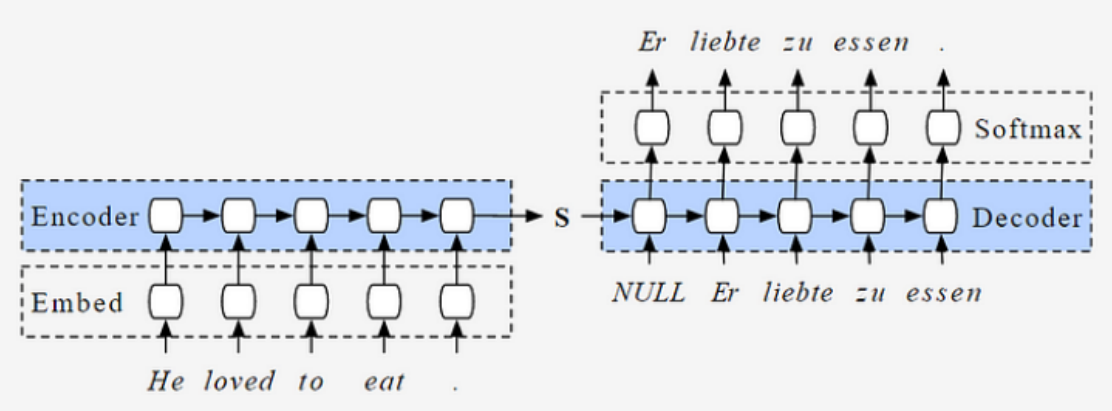

Figure 2.1: Example of seq2seq architecture. Source: (Singh, 2020)

The seq2seq models contain an encoder block and a decoder block as seen in figure 2.1. The encoder block consists of multiple recurrent units such as RNNs, LSTMs or

GRUs, which each accepts one part of the input sequence and passes on the information it collects to the next unit in the sequence. The information of the input elements are summarized and finally gathered into a context vector (*Singh*, 2020). This context vector "S" in figure 2.1 is then given as input into the decoder block of the seq2seq model.

The main use of seq2seq models are to map sequences of different lengths to each other (*Kostadinov*, 2019). However, there are still some drawbacks to using these models. One such drawback is the limited memory of the models. Seq2seq models will try to compress all the information from the input string into one context vector to feed the decoder, and depending on how long the input sentence is, the model will suffer from information loss (*Singh*, 2020). The other main drawback occurs when training the models. Again, depending on the length of the input data the earlier elements will become less and less impactful the more units are present, resulting in a vanishing gradient where the first input elements will have very little impact on the output elements (*Singh*, 2020).

Still, performance on longer sentences was surprisingly good early on compared to the expectation (*Sutskever et al.*, 2014). In addition, later seq2seq models used an attention mechanism to combat the first drawback. An example of this would be for each recurring unit in the encoder to produce its own hidden state which could also be given to the decoder (*Giacaglia*, 2019). The idea was that the decoder should attend to every word in the input and not just the summarization of them.

2.1.2 Transformer Models

Following the creation of seq2seq models a new type of model was proposed using only the attention part of the more state-of-the-art seq2seq models (*Vaswani et al.*, 2017). The transformer model proclaims to have a simpler architecture than its predecessor and does away with the use of recurrent units and convolutions in favor of these attention mechanisms (see figure: 2.2).

Somewhat similar to how a seq2seq model operates, a transformer model consists of multiple encoder and decoder units placed in sequence. Each encoder unit consists of a multi-head attention layer, followed by a fully connected feed-forward neural network. Furthermore, there are also normalization and residual connection layers in between the sublayers (*Vaswani et al.*, 2017). The decoding units work much the same, however, they employ two attention sub-layers that employ a method of masked attention. Masking prevents the model from depending on later predictions, and the extra attention layer helps the decoder focus on relevant parts of the input.

So why do we use transformers over seq2seq models? Since the multi-head attention layer is responsible for attending to multiple words in the input sentences, this allows more information to flow to the decoder units which in turn have their own way of interpreting these inputs which leads to better performance. Furthermore, transformers often train faster than seq2seq models due to the lower complexity of the model architecture.

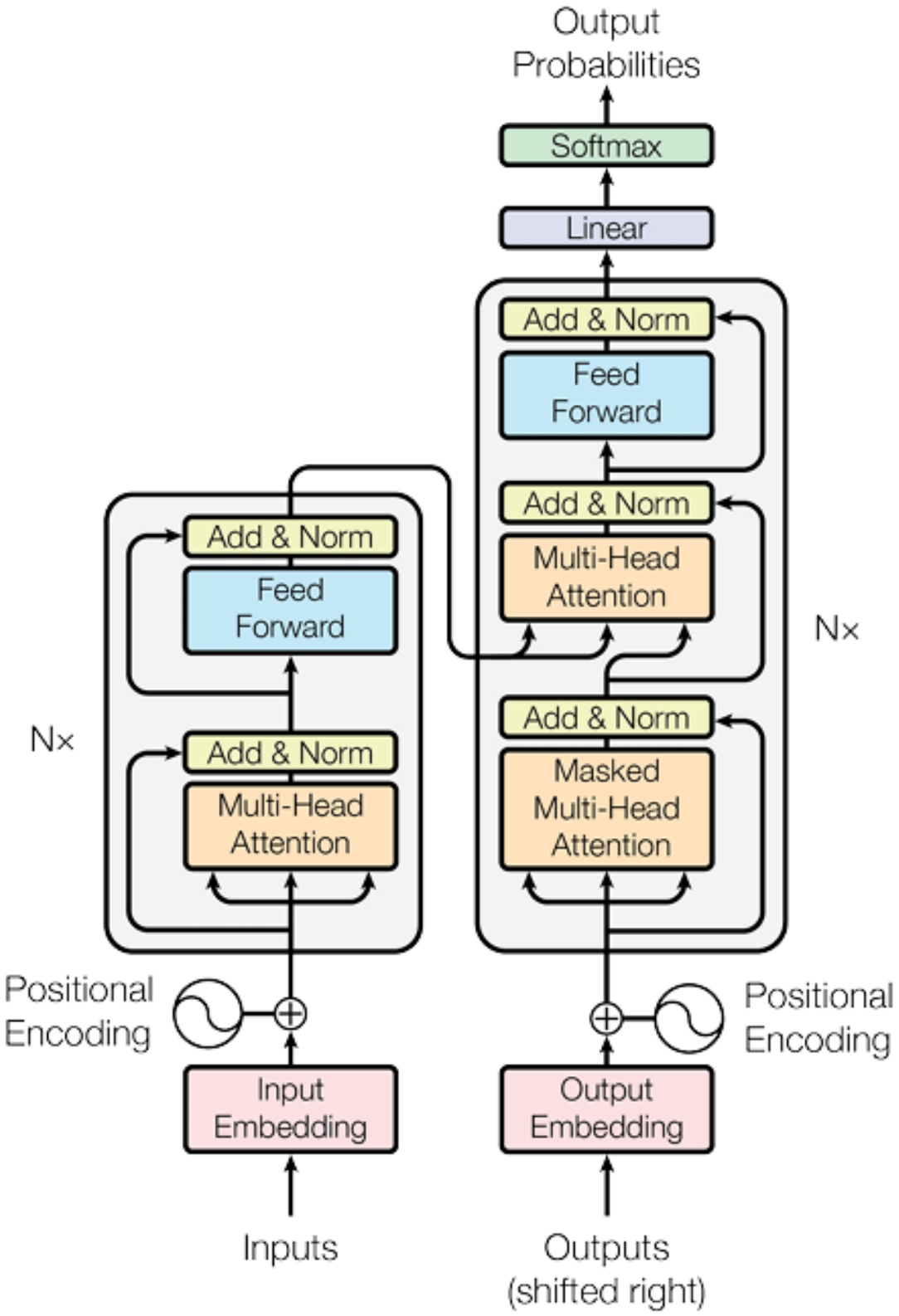

Figure 2.2: Example of Transformer model architecture. Source: (Vaswani et al., 2017)

2.1.3 Transfer learning and Text-to-Text Transformers

Now having somewhat of an understanding of what transformer models are, we can discuss text-to-text transformers and transfer learning. The idea of pre-training trans-

former models on data-rich tasks has become commonplace in NLP in the hope that this pre-training step gives a given model some understanding of language that can help performance on downstream tasks such as translation, question answering, etc (*Raffel et al.*, 2020). This idea of pre-training for the purposes of carrying on some knowledge about language is called "transfer learning" and is crucial to many modern state-of-the-art NLP-models.

One model that employs this concept is the T5-transformer model. The T5 model was pre-trained using MLM masking on the C4 (Colossal Clean Crawled Corpus) dataset (*Mishra*, 2020). MLM masking is, in essence, the process of masking words in a given dataset, and having the model learn how to predict these words to achieve some basic knowledge of language (*Issa*, 2023). This results in a model that is able to perform many tasks in addition to being flexible enough to be fine-tuned for specific language tasks. For the purposes of Question Generation in this project, the T5 transformer model fine-tuned on SQuADv1 data set was used 4.1.3.

Some other models that also employ pre-training are the BERT (*Devlin et al.*, 2019) and Roberta (*Liu et al.*, 2019) -models. Both models are used mainly to encode word embeddings for a given sentence or single-word input. For the purposes of this project, a fine-tuned version of Roberta on the SQuAD2.0 data set was used for the task of question answering 4.1.4, and another fine-tuned version of Roberta on the conll2003 data set was used for NER 4.1.7.

2.1.4 Named Entity Recognition

As mentioned, transformers can be used for the task of Named Entity Recognition (NER). NER is the NLP task of extracting named entities from a text or sentence. Named entities are in general key entities of a text such as a person, location, organization, and/or time etc. ((*Marshall*, 2019)). NER works by first detecting these key entities in a text, and then extracting them and categorizing them into the correct categories. Note, some entities consist of more than one word, and despite this NER-models are able to identify these with great accuracy. Within the sentence "Elvis Presley traveled to Machu Picchu", both "Elvis Presley" and "Machu Picchu" would be categorized as the named entities, person, and location respectively.

2.1.5 Question Generation

Transformers can also be used for Question Generation tasks. Question Generation within machine learning is the task of creating relevant questions for a given input text. When doing machine learning question generation there are generally two approaches. One using answer-dependent question generation and the other is answer-agnostic question generation (*Suraj*, 2022). For answer dependent question generation a context-text is needed. The context text can be something like "42 is the answer to life, the universe and everything." In addition, an answer-like span such as "42" is also needed for the model to know which questions to generate. From this, the model could generate the question "What is the answer to life, the universe and everything?" (*Suraj*, 2022).

When using answer-dependent question generation you also need a way to extract

these answer-like spans from the context in order to generate questions. One could of course do this manually by using human annotators, but more likely is the use of some answer extraction model. One can do this by using named entity recognition, or noun-phrase extraction to extract possible entities for which to use as answers or train another ML-model to find answer-like spans within the context.

This extra step of answer extraction adds some complexity to a QG-model, however, there is an alternative. End-to-end question generation or answer-agnostic question generation is one such alternative approach (*Lopez et al.*, 2020). Answer-agnostic models for question generation only use the context as input to generate its questions. This results in a generally less computationally complex model with similar performance.

2.1.6 Question Answering

With models that can generate questions, one would also expect a model that could answer questions. Question-answering models exist for this reason and come in generally two types; extractive and generative -models (*Chiusano*, 2022; *HuggingFace*, 2023). Extractive question-answering models work by providing a question in addition to some context in order to answer. Said model will extract the span from the context that is most likely to be the answer to the question and as such complete its task. This means that the answer given will match one-to-one with some span within the context.

Generative models work much the same way as extractive models but instead of extracting answers from the context resulting in one-to-one similarities, the answers are generated based on the context. This results in answers that are more free from the form of the context as it leverages text generation models and their text generation capabilities. There also exist generative models which do not require context at all for question answering. These models rely on the innate capabilities of storage and retrieval of knowledge within many text generation models (*Roberts et al.*, 2020). We differentiate between context-dependent models with the terms open and closed QA-models.

Additionally, there also is a question of domain-specific models. Some models perform better on general QA-tasks, while other models are domain specific. Some possible domains a domain-specific QA-model could work better on include legal documents or medical documents.

2.1.7 Similarity Metrics

Similarity metrics come in many forms, but the ones to focus on for this project are "Levenshtein Distance" and "Cosine Similarity". Levenshtein distance is a measure used to ascertain the difference between two strings. The number representing the distance is acquired by determining how many edits of the type; insertion of a character, deletion of a character, or replacement of a character are present between the starting string and the target string (*Nam*, 2019). In the case of "kitten" and "smitten" the distance is 3 as shown in figure 2.3.

Cosine similarity is a measure of similarity used when comparing two documents of often differing lengths. This is accomplished by measuring the cosine of the angle

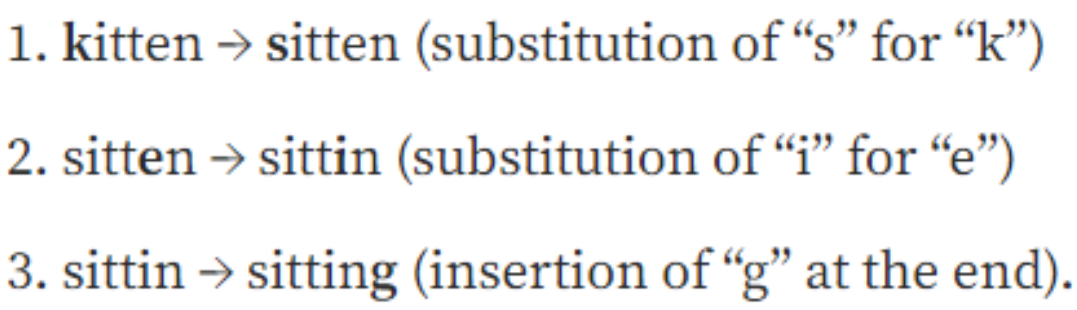

Figure 2.3: Example of Levenshtein Distance process. Source: (Nam, 2019)

between two vectors in a multi-dimensional space (*Prabhakaran*, 2018). In terms of the machine-learning domain, this can take differing forms. One can use cosine to measure similarity between vectors containing the word count of a document, or in the case of this project, which will be discussed later in 4.1.5, comparing two-word embedding vectors to ascertain the context similarity of two strings.

2.1.8 Logistic Regression

Logistic regression is a machine learning classifier and the penultimate machine learning model that was planned for use in this project. It is a probability-based solution for both binary and multiclass classification (multinomial logistic regression). Logistic regression derives its base concept from linear regression, where continuous predictions are produced by implementing a linear boundary through the data, but adds a new function in the form of what’s called a "logistic function" or "sigmoid function" (*Pant*, 2019).

The use case of the Sigmoid function is to map any input value to somewhere between 0 and 1 which in turn also makes the predictions non-continuous. Using this function in addition to a given threshold, we can take an input value of -2 and for example, map it to a probability value of 0.4. Based on the value of the threshold this probability value might be either classified as the 0-class or 1-class. In the case of the figure 2.4 where the threshold is 0.5 any X value of 0 or below would be below the threshold and classified as the 0-class while any value above would be classified as 1-class. In the case of a real-life task 0 could refer to the class "Women" and 1 to the class "Men" in the case of a gender classification task where the input could be some variables such as weight, height, and/or blood volume etc.

Now to find the optimal placement for the line within the data, a combination of a cost-function and gradient descent can be used. The cost function is in essence a function for calculating how well our line is placed, and by minimizing this function we can determine the best placement. To accomplish this task, gradient descent can be used to minimize the cost-function. Gradient descent is run on each parameter and can be thought of as standing on a mountain and feeling for a slope. By following the slope downwards at some point you will find some minimum where you get stuck and you are only surrounded by upwards slopes. This is in essence how gradient descent works with the cost-function. It calculates some minimum cost-function by taking iteration-

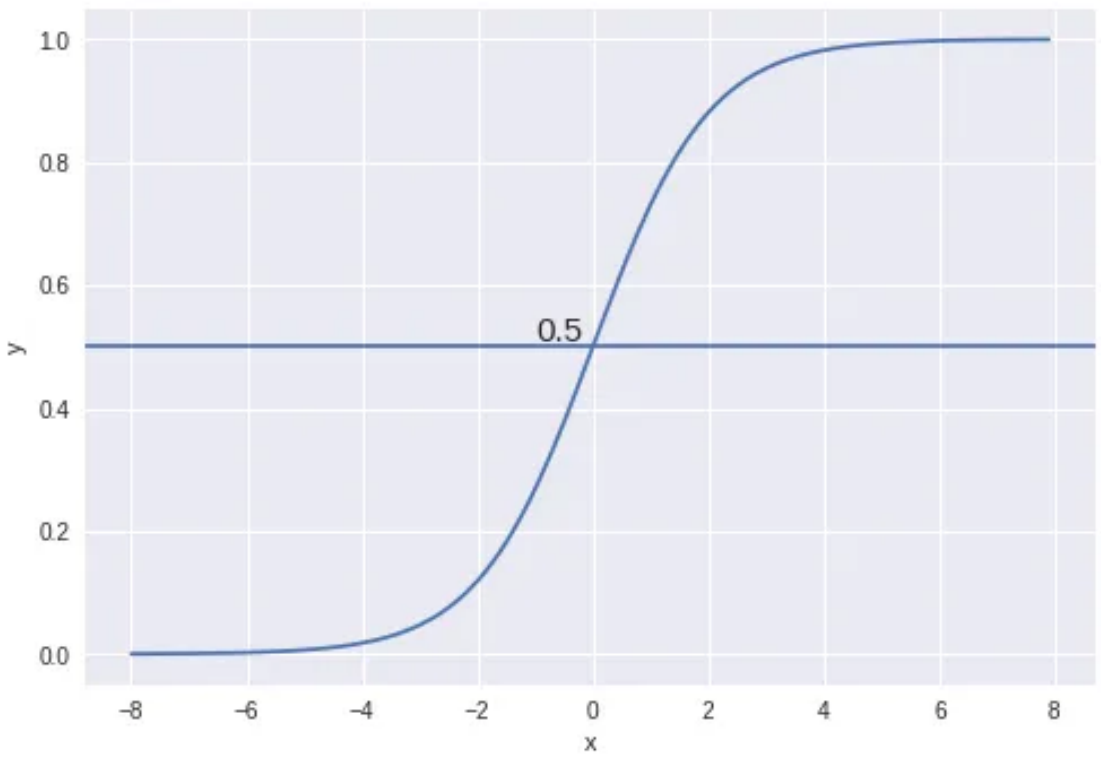

Figure 2.4: Example of logistic function with threshold 0.5. Source: (Pant, 2019)

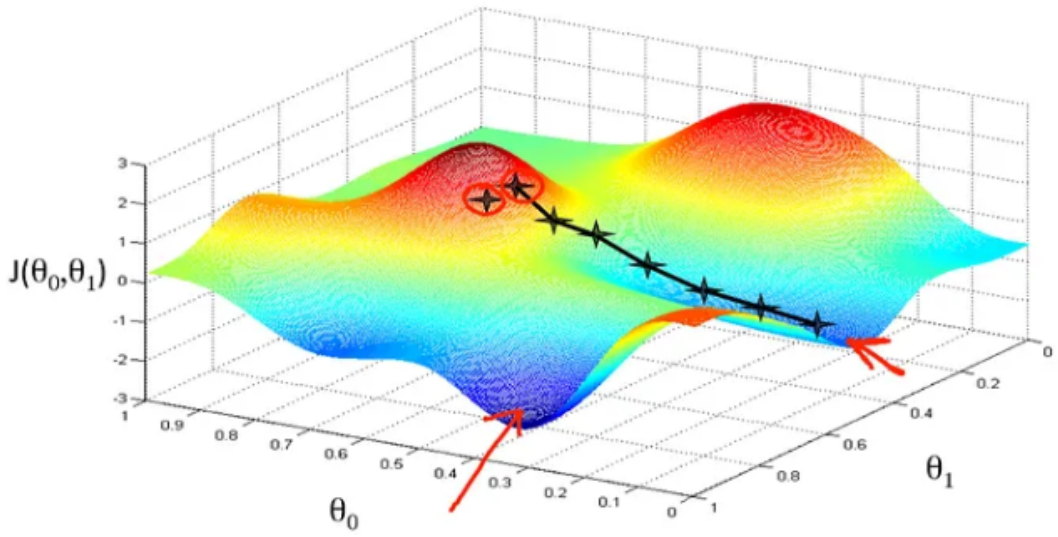

Figure 2.5: Example of Gradient Decent with a local and global minimum. Source: (Pant, 2019)

steps over the parameters, and by following the slopes down it finds the best values for the cost-function by minimizing it with each step. In figure 2.5 we can see gradient descent in effect where it lands on a local minimum.

2.1.9 Random Forest

The final type of machine learning model that will be used in this project is a "Random Forest Classifier". This classifier is based upon a conglomeration of multiple decision trees 2.6 which individually makes predictions before being combined into a single prediction 2.7. The important aspect of the random forest classifier is for each individual tree to produce uncorrelated predictions. The intuition behind this is that uncorrelated trees will protect each other from making the same mistakes (*Yiu*, 2019). This means that while some trees in the forest might make the wrong predictions, as long as the majority do not, the forest as a whole will produce the correct predictions.

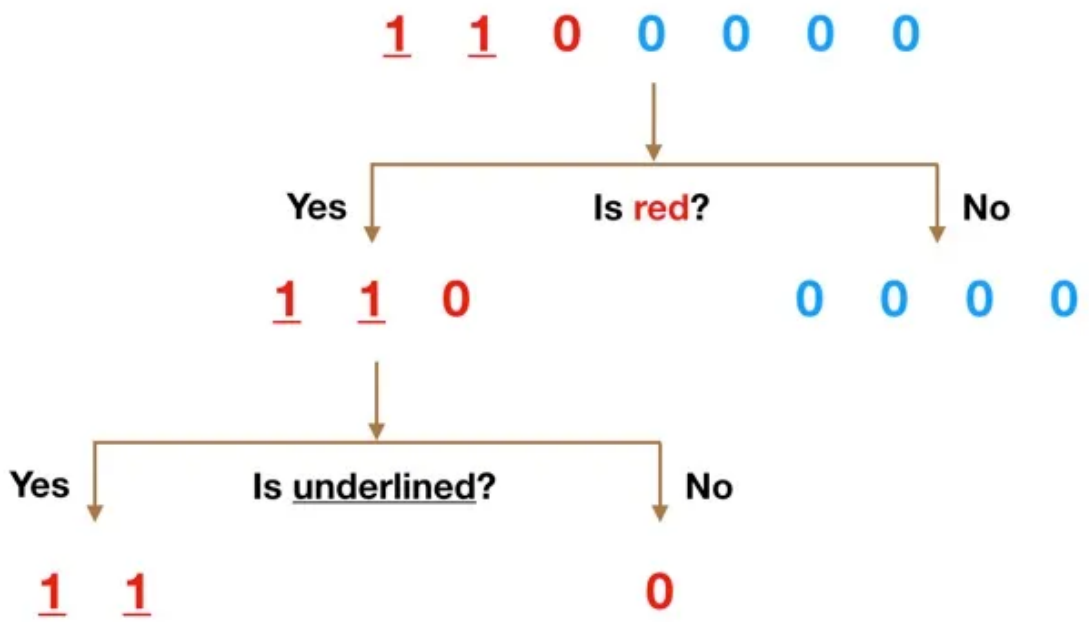

Figure 2.6: Example of a single decision tree. Source: (Yiu, 2019)

To achieve these conditions the random forest classifier uses bagging and feature randomness. Bagging is the process of how each individual decision tree samples from the training data. Instead of each tree being trained on different chunks of the training data, they sample the same amount of data from the training data which can result in overlapping data points. Feature randomness on the other hand refers to how each decision tree makes its splits. In the case of feature randomness, these splits are made by selecting from a random set of sub-features within the data. This is to induce more variation between the trees and as such hopefully result in less correlation between the decision trees resulting in better performance.

2.2 Automated Fact Checking

When it comes to automated fact-checking there exists a few common concepts which are commonly used and explained in literature. In this section I will describe the most encountered concepts, what they are, and some difficulties with them according to the literature.

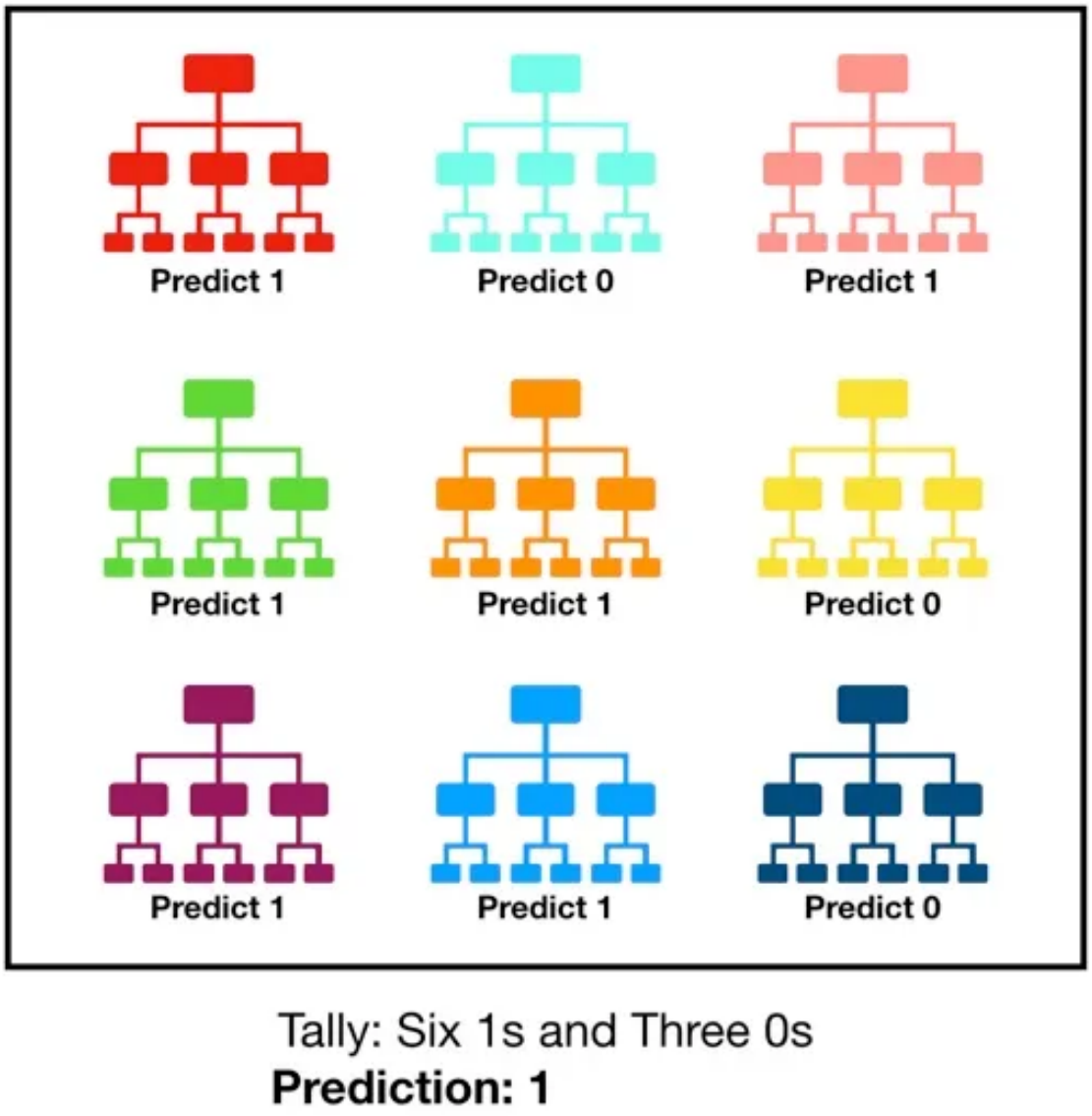

Figure 2.7: Example of an ensemble of decision trees creating a random forest. Source: (Yiu, 2019)

2.2.1 Claim Detection

When conducting any kind of automated fact-checking we are often talking about the process of labeling claims as either True (Supported), or False (Not Supported). To do this we need claims to label. In theory, one could just collect any claim from anywhere and start working from there. However, all claims are not created equal, and as such it is the job of "Claim Detection to determine which claims to verify. We can categorize claims as check-worthy and as not-check-worthy (*Guo et al.*, 2021; *Zeng et al.*, 2021). The types of claims considered to be check-worthy could be claims of public interest e.i. "Covid is more deadly than the common flu. While we often categorize not-check-worthy claims into claims of personal experiences, opinions, and claims concerning subjective statements (*Guo et al.*, 2021). The reason the latter are considered not-check-worthy is that for claims of personal experience and opinion, there is often not enough evidence to be able to correctly verify, and for subjective statements, these are dependent on the person making them thus having no objective answer. Researchers

have also created more specific categories as non-factual sentences, unimportant factual sentences and check-worthy factual sentences to try and add a bit more nuance (*Lazarski et al.*, 2021)

There is no single definition of what makes a claim check-worthy, and considering that what is check-worthy can change over time this poses problems for developing good models. Claims about Covid-19 for instance are a lot more check-worthy than claims about the Spanish-Flu due to the difference in time and relevance to current events (*Guo et al.*, 2021). Furthermore, with the total amount of claims made every day across news media and social media, researchers have to innovate and find better ways of detecting said claims. Claim detection has as such been employed for rumor detection on social media (*Guo et al.*, 2021; *Zeng et al.*, 2021).

Considering that the claims chosen during claim detection are likely to be input for processes down the line this puts an emphasis on producing good and relevant claims (*Zeng et al.*, 2021). Additional difficulties come in the form of narrow domains, annotation issues, and imbalanced data sets (*Zeng et al.*, 2021). Most claims in current data sets pertain to political domain claims, and as such the cross-domain applicability is low for claim detection. There has however been an increase in work pertaining to health claims due to Covid-19, but the domain of claim detection is still narrow. Annotating claims as check-worthy or not, is done by non-experts and as such the data produced could be faulty, and the alternative of retrieving claims from fact-checking websites only produces check-worthy claims as these are the only claims such websites would bother fact-checking. Finally, the data sets themselves could be problematic as they are often imbalanced, containing more not-check-worthy claims rather than a more even split.

2.2.2 Evidence Retrieval

Before fact-checkers can make predictions about the veracity of claims they often need some kind of evidence to be able to form the correct conclusions, and this is no different for automated fact-checking (*Guo et al.*, 2021; *Zeng et al.*, 2021). While predictions can be made without any evidence (*Guo et al.*, 2021) it does not usually produce good results. Evidence retrieval is the process of gathering relevant evidence to correctly ascertain the veracity of a claim. This is often done in two steps namely, document retrieval and rational selection (*Zeng et al.*, 2021). Document retrieval is the process of finding relevant documents based on different metrics, to substantiate a given claim. This can come in the form of querying search APIs of something like Google or Wikipedia using entities in the claim, or by using similarity metrics between the claim and some document. Once a document is found the “rational selection starts which is the process of retrieving the relevant parts of the document, often sentences, to use in claim verification (*Zeng et al.*, 2021). There is an additional way of evidence retrieval, called “stance detection, which is used in the case of having limited evidence (*Guo et al.*, 2021).

One problematic consequence of gathering evidence in this manner is that one has to assume all the evidence sources contain trusted information which is not always the case (*Guo et al.*, 2021). One can limit the evidence to only include encyclopedias or academic papers, which can help but might not be enough information in some cases. This also leads to a problem where if many data sets base their evidence exclusively

on something like Wikipedia, you can end up with a situation where Wikipedia might for all intents and purposes be the only source of all information. In addition to the textual evidence presented so far, there is also evidence in the form of pictures, tables, metadata, and knowledge bases. However, in the case of knowledge bases complete information is not often obtainable.

2.2.3 Claim Verification

Claim verification refers to the process of utilizing a given claim and often the corresponding evidence to determine the veracity of the claim (*Guo et al.*, 2021; *Zeng et al.*, 2021). The simplest form of this is binary classification of True and False. However, many fact-checking organizations often use a degree system in how claims are classified, and furthermore using true and false when describing claims might be too strong of a statement. A common practice is to use supported, refuted and maybe include a third option as lack-of-evidence (*Guo et al.*, 2021). Additionally, in the cases where there are multiple label choices e.g. True, False, half-true, etc. these categories are often condensed down into only a few, as it is extremely difficult for a machine to identify multiple categories especially if only some parts are true (*Zeng et al.*, 2021). Some recent models have seen success in aggregating multiple pieces of textual data (evidence) to solve larger more complex verification tasks. This also helps in the cases where multiple sources of evidence are found, but not every one of the sources are relevant (*Guo et al.*, 2021). These methods using external sources of evidence are often considered the best (*Lazarski et al.*, 2021).

When it comes to the actual process of verifying claims, there exists a few other methods of note. "Language Analysis is one method that analyzes the actual wording or language of a given claim (*Lazarski et al.*, 2021). Depending on what type of language is used, often referring to very subjective or hyperbolic word choice, it can be a predictor for unsupported claims. Using this method to determine the veracity of claims has had some success for binary classification, but has had a higher degree of difficulty for multi-class classification. In addition, depending on how sources are cited, the veracity of claims can be impacted. Seeing as a source citing another confirmed truthful source will increase the likelihood of the original source being truthful.

Another method is "claim matching, the process of comparing claims found during claim detection with a database of claims, or a database of already checked claims (*Guo et al.*, 2021; *Zeng et al.*, 2021). This can free up computational resources as in some cases sentence-level similarity could be used instead of the usual approaches. However, some downsides include the fact that for this approach to work a similar claim as the one being checked has to exist and be stored somewhere. In addition, there have to be appropriate tools to query for these similar claims, and if found these claims would hopefully have been manually annotated. Since if they are not manually annotated this process could result in the wrong classifications for all models querying the same database. This does however leave us with the same problem that automated fact-checking tries to solve, the need for alternatives to human labor (*Lazarski et al.*, 2021).

2.2.4 Justification Generation

Having insight into why a given model predicts a claim to be supported, refuted or lacking in evidence is extremely important if people are going to have any trust in the outcome. One can imagine that if Politifact only posted their result without an explanation of how they came to their conclusion people would not take the organization seriously. In automated fact-checking "Justification Generation is the process of gaining some insight into the reasoning of the models and how they arrive at their conclusions (*Guo et al.*, 2021). This has some similarities with "Factual Error Correction which we will look at in the next section. While this can be done for any part of automated fact-checking in terms of justification for claim detection or evidence retrieval, it is most often used for Claim Verification due to the importance of this step.

There are three strategies of justification generation, namely, using attention weights, use of logic-based systems, and summarization (*Guo et al.*, 2021). Attention weight refers to the process of highlighting which parts of the evidence had the biggest impact on the verdict giving us some insight. Using logic-based systems could make the processes understandable for humans as a derivation of the veracity of the claim (*Guo et al.*, 2021). Finally, the system could produce textual explanations of their decisions. All justifications generated by the system could be evaluated based on readability, plausibility and faithfulness (*Guo et al.*, 2021). Readability references a human's ability to understand a given justification, plausibility references how convincing the justification is, and faithfulness references how accurate the justification is. The goal being to get more insight into the reasoning of a given model to build trust in the results. Some data sets provide gold explanations of results which can be used to train models in generating post-hoc justifications for the verdict (*Guo et al.*, 2021).

2.2.5 System Design and Modeling Strategies

When referring to an automated fact-checking system one often thinks about the presented concepts as parts of a pipeline (*Guo et al.*, 2021; *Zeng et al.*, 2021). The system can start by collecting claims, then do claim matching, followed by evidence retrieval and so on. This is often the case for most such systems, but it does not have to be. *Zeng et al.* (2021) argues that by making systems with multiple steps, the likelihood of errors occurring in each step increases, and leads to an overall worse system. Training models to complete all tasks involved in automatic fact-checking will lead to better-performing systems according to *Zeng et al.* (2021).

2.3 Factual Error Correction

A related approach as to what this project aims to do has been investigated before in "Evidence-based Factual Error Correction" (*Thorne and Vlachos*, 2020) as an extension of factual error verification. In addition to assessing whether a claim/factoid is supported, refuted or lacking in evidence. The aim of error correction is to also be able to "correct" the given claim based on evidence, and give an insight into fact-checking. Given a claim such as the one presented in the aforementioned paper of *"Brown recluse spiders do not bite"*(*Thorne and Vlachos*, 2020), and the evidence *"Similar to other*

recluse spider bites, their bite sometimes requires medical attention." The goal of the system is for it to both assess the veracity of the claim whether it is supported, refuted or lacking, but also make a correction in the form of *"The brown recluse spider's bite sometimes requires medical attention."*

2.3.1 Approach: Using Distant supervision

(*Thorne and Vlachos*, 2020) presents an approach using "distant supervision" for correction generation. "Distant supervision" was used to generate corrections due to a lack of data containing claims and their corresponding corrections. Masking was used to create masked-claims which could be corrected by the system. Masking in this case refers to the process of removing tokens from a given claim that are important for the claim to be considered supported. An example of this could be the claim "Bruce Wayne is Batman". The masker could then through use of the evidence determine which tokens to remove to still include the meaning of the claim while still leaving room for a correction e.i. "Bruce Wayne is [MASKED]" or "[MASKED] is Batman". The corrector could then use this masked-claim to generate corrections from evidence e.i. "Bruce Wayne is Batman".

Furthermore, the paper focused on three points when looking into the quality of the resulting corrections. They looked at "Intelligibility", whether or not the corrections produced were grammatically correct and understandable without supplementary evidence. They looked at whether or not the corrections were "Supported by Evidence", meaning the corrections have evidence to back up the claim. Finally, they looked at "Error correction" which refers to the idea that corrections should be targeting the part of the claim that produces the erroneous claim.

To determine how well their system did in regards to these three metrics, the researchers used a few different approaches of how to score the resulting corrections using both manual evaluation and automated evaluation. For the human evaluation looking at the three measures of correction quality mentioned above, the model using gold evidence performed the best. It got 98.9 in "Intelligibility", 88.9 in "Supported", and 68.9 in "Corrected" using a pre-trained T5 transformer. This model was meant to represent the ceiling of what could be possible with enough data, as there was not enough gold data to do this in a feasible way. As for the T5 transformer models using maskers, the best performing models were using random masking for training referring to masking random parts of the claims, and a heuristic for the testing. Heuristic refers to only masking the parts of the claim not found in the retrieved evidence. The respective scores for the three categories were 89.3, 57.9, and 40.0. It was found that automated evaluation using SARI corresponds well to the manual evaluation. The work concluded that there is a need for better automated evaluation as manual evaluation, as similarity to reference sentences is not always representative. In addition, they stressed the need for more research into better masking technology as the different masking techniques used had noticeably different results.

2.3.2 Approach: Using Error Correction for Summaries

Another type of factual error correction worth mentioning is "Factual Error Correction for Abstractive Summaries." Newer methods of generating summaries from article texts do not only extract text sentences directly but also generate completely new sentences based on the given article text (*Lee et al.*, 2022; *Wang et al.*, 2020). However, when generating these summaries that often contain novel text, there is a chance for the generated text to be factually inconsistent with the text in the source article. Factual error correction can be used as a post-editing tool to correct such errors. In the 2020 paper "Factual Error Correction for Abstractive Summaries Using Entity Retrieval" (*Lee et al.*, 2022) one such approach is discussed.

The proposed method consists of entity detection and correction. Given an erroneous summary and a set of relevant evidence sentences extracted from the source article, embeddings are created for the entities of both the summary and evidence. Thereafter the summary and evidence are concatenated. The system does an error detection calculation on the summary entities and gives an error score for each entity. If the error score is above a given threshold the entity is considered to be an error. The system then does an error correction calculation for the erroneous entities and tries to find the entities in evidence with the highest score to replace the erroneous entity. The resulting summary has then had all its erroneous entities replaced with the likely factual entities from the evidence. The paper notes that the achieved results are similar to other summary correction models but with much less computational cost. One point of failure presented for this approach is that if a correction candidate is not correctly identified by the named entity recognition (NER) tagger, it wont be represented as an option for the model.

2.3.3 Approach: Unsupervised Question Answering

The third to last approach worth mentioning is more akin to what this project aims to do and can be found in "Unsupervised Question Answering for Fact-Checking." (*Jobanputra*, 2019) Using a somewhat similar concept as (*Thorne and Vlachos*, 2020) of masking text, the researchers created a three stage pipeline for solving the task through unsupervised question answering. The first stage being question generation. Using a NER-tagger, claims from the FEVER data set are tagged. From these tagged claims, words of the type PERSON, CITY, DATE, are masked and these masked claims are used as questions. For question answering, BERT is used to predict the missing tokens in the generated questions, giving an answer for each question generated from a claim. Finally, the last stage is label classification. Based on the amount of questions correctly answered a score is given. The score is then compared to a given threshold of how many questions out of the generated ones need to be correctly answered for the claim to be either labeled as supported if above the threshold, or in need of manual review if the score is below the threshold. In the results, the researchers found that a threshold of correctly answering 3/4 of all questions generated for a given claim lead to the best results.

2.3.4 Approach: Combined QA for Summary Correction

It is however possible to combine approaches and do question answering for the previously mentioned summary correction (*Wang et al.*, 2020). QAGS is a three-step framework consisting of question generation, question answering, and similarity computation between answers. Unlike (*Jobanputra*, 2019) this method does not end with a label classification step and uses Question Generation models(QG) instead of masking for QG. A QG model is trained on a data set consisting of a source text and answers, and different heuristics are used to filter out low-quality questions. The goal of this step is to produce high-quality questions from the summary text we want to correct. The next step of question answering (QA) attempts to answer the generated questions. This is done using both the source article and the summary as evidence, producing two sets of answers. For the last step of "comparing similarity", the answers in both sets are checked against each other. The goal is for the answer set produced using the summary to contain the same answers as the answer set produced using the source text, as this will give a high similarity score. Also given similar answers, it is more likely that the summary is factually correct and has not gained any erroneous elements during summarization. The QAGS score is the final metric for evaluating how well the full model has done and takes the average of all the similarity scores for answers and returns a score. In addition, as with the other factual error correction approaches we also get insight into what the errors were when looking at the discrepancy in answers given using source and summary.

This method showed great results when looking at human-judged consistency scores, producing 54.53 on the CNN/DM-data set which contains relatively simple abstractive summaries, and 17.49 on XSUM-dataset which contains much more abstractive summaries, meaning more novel text. While other automated evaluation metrics produced 29.68 for CNN/DM and 13.22 on ZSUM at best. Note these scores were relative to human judgment on the same summaries.

2.3.5 Approach: Explainable fact-checking through question answering

Finally, the last approach to be discussed and a late find in the overall scheme of the project was the approach described in EXPLAINABLE FACT-CHECKING THROUGH QUESTION ANSWERING (*Yang et al.*, 2021). The approach described in the paper is very similar to some of the work this thesis aims to implement. It first starts with generating multiple questions and corresponding answers in parallel for a given claim. Next, the generated questions are answered using a question-answering model using related evidence. This equips the researchers with a combination of questions, and two types of answers; one using the claim as context and one using provided evidence as context. As will be described later this is very similar to our approach so far, but at this point, it deviates. The next step is to compare the answer pairs and predict some form of veracity depending on the comparison results. The paper proposed a few baseline methods for accomplishing this, however, the main method to focus on is the proposed answer comparison transformer model. By using the questions and answers generated, the researchers trained a model that could attend to questions. The attention

mechanism aims to be a solution to the problem of differing importance of generated questions. Such a model also has the added benefit of being able to handle answers with differing words yet the close semantic similarity. Compared to state-of-the-art models for factual claim verification the model performed comparatively at 75.55% for dev accuracy and 73.43% for test accuracy, only about 1.5 percentage points lower than a baseline blackbox approach.

The benefit of this proposed method is, however, its explainability. The explainability takes a few forms, the first being question importance. By knowing which questions are important they gain more insight into how the model predicts veracity. Second, in the case of both right and wrong answers, by using the inherent function of question-answering they can look at the answers to get insight into which part of the claim might be wrong. Finally, the last aspect of explainability comes in the form of the pipeline approach used. By breaking fact-checking into multiple steps the researchers can get more insight into the inner workings of the model as opposed to the alternative which are often blackbox models.

Chapter 3

Methodology

3.1 Design Science Criteria

No.	Guidelines	Compliance
1	Design science research must produce a workable, practical artefact in the form of a construct, model, method, or instantiation	It can be used according to the original purpose, by the intended users. Be careful not to over promise. Be careful to promise the right things.
2	Ensure that the artefact produced is relevant and important	Has anyone tried to solve it before? Why hasn't it been solved before? How important can it be? Is it too difficult?
3	Rigorously evaluate the artefact produced	How do you know you accomplished what you wanted? Don't just ask people if they like it. Analytically using a mathematical model. Empirically using field study or experiment
4	Produce an artefact that makes a research contribution.	Solve a previously unsolved problem. Show that an artefact can be produced when it was previously unclear that this is possible.
5	Follow rigorous construction methods.	The method must be rigorous and replicable
6	Show the artefact is the outcome of a search process	This is done after you're finished
7	Clearly communicate the research process and outcome	Say a little about your thesis, any conference papers planned

Table 3.1: The seven guidelines for rigorous design science and how the work reported in this thesis fulfils them.

3.2 Design Science Implementation

Design Science Research is a methodology developed for use in relation to the domain of Information Systems. The goal of this methodology is to both produce and evaluate an artifact, in addition to contributing something new to the field, as science often does. For this master thesis, the work will be of the type of Design Science, and in this section, I will be discussing how the seven guidelines of rigorous design science as seen in table 3.1 are to be fulfilled.

1. The aim of the work is to produce a pipeline method for factual error correction and expand on the concept of explainable fact-checking. The goal of such a system is to take any given factual claim as input and produce corrections when needed, in addition to verifying a claims truth value. Furthermore, corrections made by the system should be based on related evidence for the given claim. The goal of the pipeline will be to help in combating misinformation by correcting misinformed claims. Users of a fully developed and idealized version of this system could be anyone from journalists wanting to fact-check sources or their own work, to the general public wanting to be certain of some information they have discovered. However, due to the nature of this being early research, it is more likely that the artifact produced would need to be further tested and developed before such a use case becomes possible. With this in mind, it is more likely that other researchers would be the main users of the artifact and further develop and expand on it.

2. As noted earlier in this paper more robust methods of fact-checking are needed to help combat the detrimental societal effects of misinformation and disinformation. As such research into possible methods that can help with these problems is both relevant and immensely important. An artifact using the specific approach I aim to use has not been completed before, but a similar approach has been used for summaries instead of claims in (*Wang et al.*, 2020) and for claims in (*Yang et al.*, 2021). Why this approach has not been completed thus far is unknown to me, however, there are many possible culprits. Effective methods of machine learning approaches such as question generation and question answering might not have been available if the task had been attempted earlier, or there could have been a lack of relevant datasets available for the task. It is likely impossible to know exactly the reason, seeing as it is a relatively complex task with many possible components.

3. When it comes to the evaluation of the artifact produced we will look at whether or not it is able to make the needed corrections and verifications required of it. To achieve this we will look at the accuracy of refuted claims turned into supported claims through corrections and supported claims turned into refuted in the case of miscorrections. We will also look into what makes for a high-quality question and try to understand the mechanisms behind this. A more extensive look concerning this point can be found in the evaluation subsection of methods 4.2.

4. If the research is successful the artifact produced will represent a new approach for factual error correction by using question generation and question answering

for factual error correction. By virtue of being a new approach its existence will have value and as such expand on current research. In addition, the planned investigations into understanding question generation in relation to claims will also provide contributions to the research domain. For a more specific idea about the contributions as a whole see 1.3.

5. In the making of this system, a pipeline will be created using rigorously planned-out methods of question-generation, question-answering, similarity metrics, corrections and verification systems. Each part of the pipeline builds upon models that can be implemented by others for replicability, and further research. The code will also be provided for better availability to replicate the results. The methods for this project will be further discussed in the methods section 4.1. Our methods of evaluation as described in 4.2 were also planned-out and something that is replicable.

6. When it comes to the "being an outcome of a search process" criterion in the guidelines list, this is something that can be observed throughout the thesis. From our original motivations to our end results, the entire process was built upon previous research and ideas of potential alternative solutions. Finding either the best approaches by reading literature, or testing conditions through trial and error was a major part of the work in this thesis. For our system, multiple different combinations of conditions were tested to get the results that were deemed to be the best, and as such the results can be considered the outcome of a search process.

7. Currently, there are no broader plans to communicate the research in this thesis which is to the detriment of good design science research. However, due to the nature of this being work done as part of a master's thesis, the work produced will be read and evaluated by researchers within the same field, and potentially looked at by fellow master's students. The communication criteria might not have the best execution of the requirements, but it is also not something that was ignored.

Chapter 4

Methods

4.1 Implementation

This section will explain how the proposed pipeline was implemented. It will begin with a general explanation of the pipeline before going into more detail about the individual parts themselves. It is important to note that one part of the pipeline is optional and will be labeled as such.

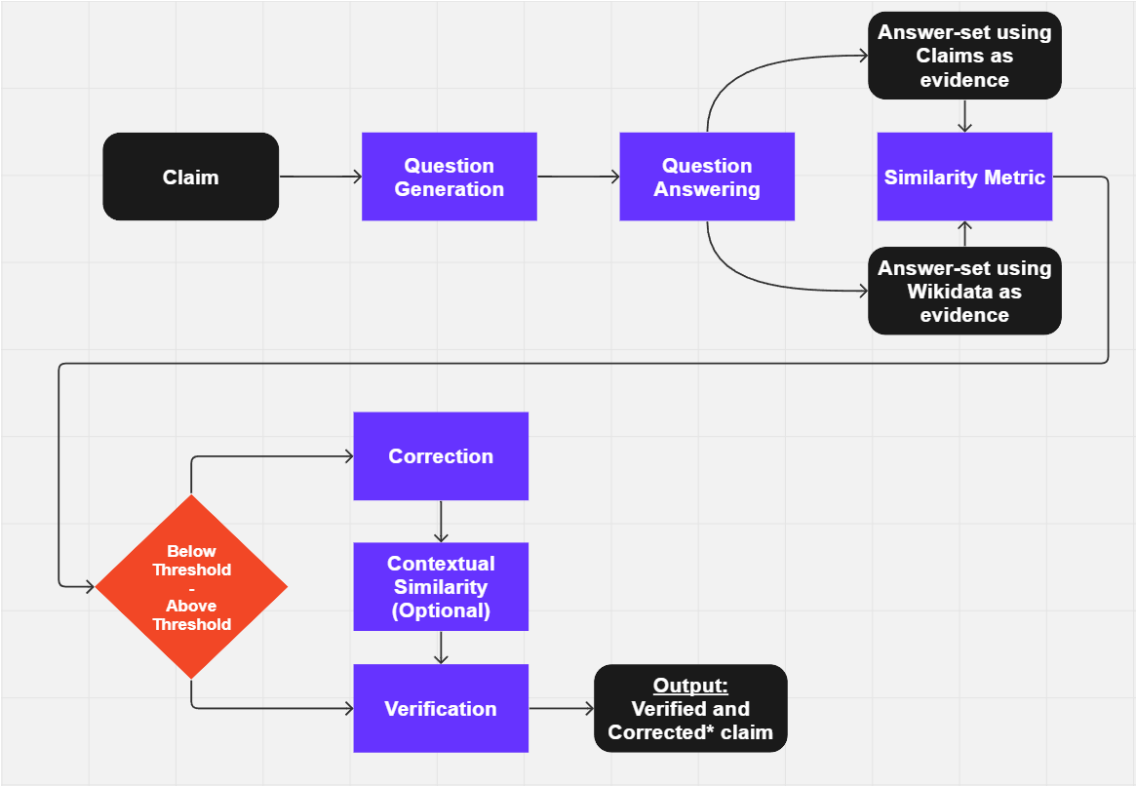

Figure 4.1: Example of the full pipeline
**Not all output claims are corrected*

4.1.1 General Overview

The pipeline starts out with extracting claims from the FEVER-dataset (*Thorne et al.*, 2018). These claims are then given as input into a Question Generation Model which creates questions for each of the input claims 4.1.3. Following this, the newly created questions are answered using a Question Answering model 4.1.4. For each question there are two answers generated, one using the claim as evidence, and the other using wikidata sentences as evidence, which are provided by a database-file included with the FEVER-dataset 4.1.2. Note that for the work in this project the wikidata evidence is viewed as true information and answers using wikidata sentences will be seen as correct in the case of differing results between the two evidence types. This implies that if a particular input question generates two different answers, the one using wikidata as evidence will be deemed to be the correct one.

To figure out whether or not the answers are the same or not, the pipeline employs a similarity metric method as its next step 4.1.5. If the answer similarity is below a certain threshold the pipeline will make the needed corrections. It is important to note that if the answer similarity is above the threshold no correction is needed and this step will be skipped. However, when a correction is needed part of the original claim is replaced with the answer using wikidata as evidence to make the correction. After this, there is an optional step of checking the contextual similarity between the original claim and the corrected claim to determine if the correction is within reason 4.1.5. Finally, all claims are given to a verification model and values of Supported, Contradicted, and Indeterminate are given 4.1.6. With this final step, the pipeline has completed its operation of correcting a claim and verifying it, and with this, we have achieved some explainability in the form of the correction in addition to the veracity prediction.

4.1.2 The FEVER Dataset

To achieve the stated goals of this project the FEVER data set was used in all the testing for the purposes of achieving said goals. The FEVER dataset or as it is also known "Fact Extraction and VERification" is a publicly available data set for verification against textual sources (*Thorne et al.*, 2018). The data set was originally created to tackle the lack of expansive data sets seeing as some research relied on datasets of around 300 claims while the FEVER data set boasts up to 185,445 claims. All the claims within the FEVER data set are manually annotated using Wikipedia introductions and classified as either Supports, Refutes or NotEnoughInfo. For the supported and refuted claims, the relevant Wikipedia introductory sentences were also returned. For the claims themselves, they were created by human annotators retrieving and periodically mutating the same introductory sentences in sometimes meaning-altering ways. Finally, this leaves us with a data set containing textual claims, their ids, an annotated label of either Supports, Refutes or NotEnoughInfo, and finally the corresponding evidence query for retrieving evidence sentences as seen in figure 4.2.

Of the 185,445 claims within the FEVER data set 145,449 were allocated to the train set in the form of 80,035 Supported, 29,775 Refuted and 35,639 NotEnoughInfo. Furthermore, both the dev set and test set contained 9,999 claims each split into 3,333 in each of the label categories. Lastly, 19.998 claims were reserved for other uses by

```
{
  "id": 89891,
  "label": "REFUTES",
  "claim": "Damon Albarn's debut album was released in 2011.",
  "evidence": [
    [
      [107201, 120581, 'Damon_Albarn', 17]
    ]
  ]
}
```


Figure 4.2: Example claim from FEVER data-set with corresponding evidence query.

the researchers. For this project, we choose to work with the Supported and Refuted claims which were a part of the dev-set, the reasoning for which were two-fold. Firstly, the provided version of the test set did not provide the required labels and evidence needed for the work, and on the other hand, while the train set was larger we had no plans to use the FEVER data set to train any models. Second, the choice to not include NotEnouthInfo claims was a deliberate one. Seeing as one goal of this project was for the system to take a misinformed claim as input and correct it. Thus using claims that were uncertain in the form of NotEnouthInfo-type claims seemed counter productive. Especially since such claims could by definition not be corrected with the given evidence provided by the data set. However, there exists the potential to create a system that would be able to identify such claims and remove them rather than explicitly removing them from the data set based on provided labels.

For the evidence part of the data set, it can be important to understand how the formatting works. For each claim, the evidence is given as list elements. These list elements refer to what queries to search up in a provided database file and which of the returned sentences are relevant. Each claim has at a minimum one corresponding evidence sentence, however, in many cases, a given claim could have multiple corresponding evidence sentences. The evidence could in some cases also point to the same sentence which was sometimes needed for internal testing. For the purposes of my work, duplicate sentences were also ignored.

4.1.3 The First Step: Question Generation

The first step of the pipeline is question generation using one of the existing T5 transformer models found within the repository provided by (*Suraj*, 2022). The models provided were all trained using the SQuADv1 dataset and uses the pre-trained T5 transformer model. The goal of this step is to take the claims from our data set and generate questions for each of the claims, which the chosen repository provides several different ways of accomplishing. One can employ an answer-aware model, a multitask QG-QA model, or an end-to-end question generation model which is answer-agnostic. For the purposes of this project, the answer agnostic end-to-end model was chosen for its capa-

bility to produce multiple questions for a single sentence without the need to rely on a separate answer span extraction method which in turn reduces computational complexity. See section 2.1.5 for a more detailed description of Question Generation.

While the multitask qg-qa model provides QA capabilities which we will need later, it was not suited for our task due to our need to provide different types of evidence for each question. Furthermore, the answer-aware model would limit us by requiring an answer span extraction method in the form of named entity recognition, or other similar methods to generate questions. In this case, one such method is already implemented in the model provided, although this adds to the computational complexity of the model. Seeing as this is the case, another factor for choosing the answer-agnostic model is based on the research of (*Lopez et al.*, 2020), which shows that an answer-agnostic model can be equal or better than an answer-dependant model for generating questions while keeping the computational complexity of the model lower.

Claim: "*Damon Albarn's debut album was released in 2011.*"

Generated Questions:

- One: "*What year was Damon Albarn's debut album released?*"
- Two: "*What year did Damon Albaniarn release his debut album?*"

Figure 4.3: Example of Generated Questions for claim seen in 4.2 using answer agnostic QA-model.

4.1.4 The Second Step: Question Answering

Continuing on, the next step in the pipeline aims to answer the questions generated in the previous step. To do this we will use an existing question-answering model from hugging face (*Chan et al.*, 2023). This model uses the roberta-base model and fine-tunes it on the SQuAD2.0 dataset with the end goal of answering provided questions.

The QA process begins by taking any given claim and retrieving the related questions for the said claim. These questions are first answered using the claim itself as evidence. Next, the same questions are answered using the provided FEVER wikidata sentences as evidence for the QA model. Once we have both answers for a given question we create a list-element containing the answers, a confidence score for the answers using wikidata as evidence, and finally the span where the answer using the claim-based evidence was extracted from as seen in 4.3 "Output".

While the process of using claim-based evidence is very straightforward, the use of wikidata sentences as evidence is not as simple. The wikidata evidence is queried from the provided database on a claim-by-claim basis. Each claim can have multiple related evidence queries and these queries can in some cases point to the same exact evidence sentence. As such before any evidence sentence is retrieved any duplicate query is removed. After the correct queries are identified they are then executed, and the correct evidence sentences are extracted and cut down to only the sentence span. This cut-down process is used to remove tags continuing after the end of a given evidence sentence.

Claim: "*Damon Albarn's debut album was released in **2011**.*"

Evidence (Wikidata Query: "[107201, 120581, 'Damon_Albarn', 17]"):
"*His debut solo studio album Everyday Robots -- co-produced by XL Recordings CEO Richard Russell -- was released on 28 April **2014** and featured collaborations with Brian Eno , Natasha Khan and the Leytonstone City Pentecostal Mission Church Choir as well as sampling several rants by Lord Buckley .*"

Question One: "*What year was Damon Albarn's debut album released?*"
Answer(Claim): 2011
Answer(Wikidata): 2014

Question Two: "*What year did Damon Albaniarn release his debut album?*"
Answer(Claim): 2011
Answer(Wikidata): 2014

Output (shown as list-elements):
['2011', '2014', 0.9037638306617737, (43, 47)]
['2011', '2014', 0.8201749920845032, (43, 47)]

Figure 4.4: Example of Question Answering for questions seen in 4.3 where the claim answer is 2011 and wikidata based answer is 2014.

Finally, all evidence sentences are joined together to create one piece of evidence used in the QA-model.

4.1.5 The Third Step: Similarity Metrics and Correction

Once we have completed both the question generation and answering we can continue on to compare the answers we were given. Firstly, we need a way to determine whether or not the two answers for a given question are similar or not and to do this we use string matching in the form of Levenshtein Distance **(SEE: SECTION ON LEVEN)**. In addition, we need some way to better handle matching strings of varying lengths.

Both can be accomplished using the "fuzzywuzzy" library in python using the method "token_set_ratio()" (*Pykes*, 2023). This method calculates the Levenshtein Distance for the two answers and returns a number between 0 and 100 corresponding to similarity, in addition to removing common tokens which helps if the answers are of different lengths. Furthermore, it also does not take into account token order which is helpful since answers might not start in the same way but might still mention the same entity if it is an entity-type answer. While this approach has its limitations such as answers referring to the same instance of an object or person etc. using different words, it is still a very powerful tool for comparing strings. Other options that might better tackle the same entity with a different name problem could be some sort of knowledge-graph-based approach or some form of embeddings-based similarity metric.

Once all the numbers are calculated for the answer pairs the system can make corrections. The process of correcting starts with checking the similarity between answers for the given claim. If a claim has only one answer pair below the threshold, the answer using wikidata evidence will replace the span of text in the original claim where the answer using the claim itself as evidence was extracted from as seen in 4.5, and

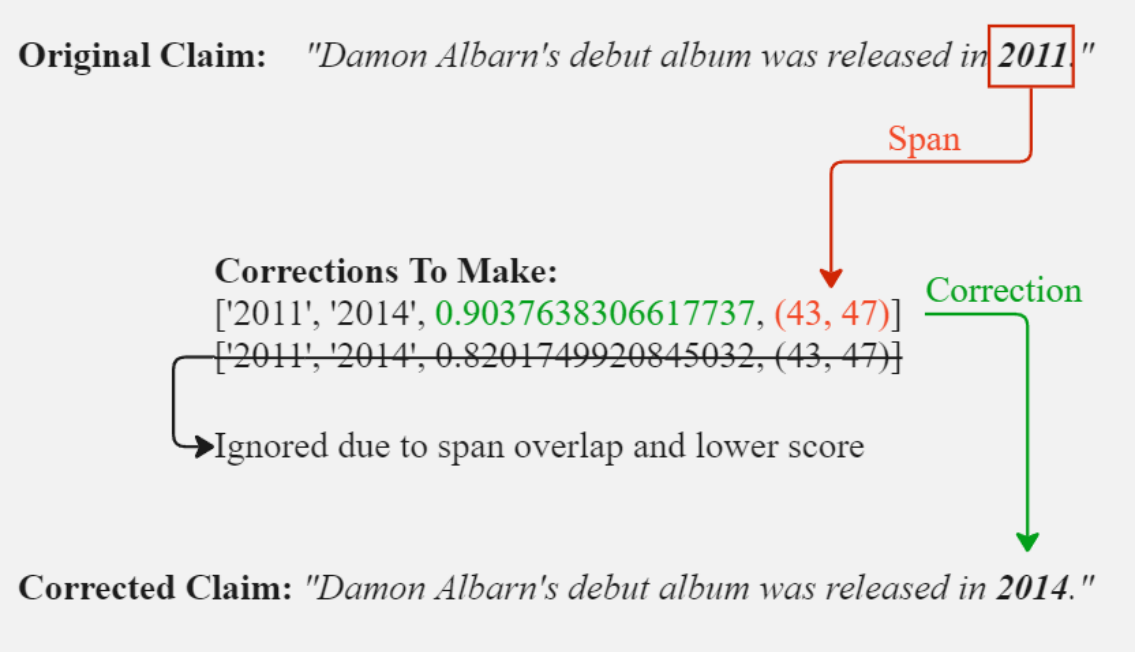

Figure 4.5: Example of Correction using answers from 4.4 where only one of two corrections are made due to overlapping spans. Correction to make was chosen based on score-output from QA-model. ***Note:*** *Simillarity Score between both answers were 75.*

with this, the claim is then considered corrected. In the case where there are multiple answer pairs below the threshold, the system checks if any of the claim-based evidence answers have overlapping spans. If none of the spans overlap the corrections will be made in the same way as the previous example. However, if the spans do overlap, the answer pair with the highest confidence score from the QA-step will be used for the correction also seen in 4.5.

Finally, we come to the last step of correction which is the optional step of measuring the similarity between the original claim and the corrected claim. This is done by calculating the cosine similarity of sentence embeddings. This can be used to find the contextual similarity between the original claim and the corrected claim which in turn can work as an extra measure in making sure our corrections are satisfactory. Furthermore, this can also empower us to be more forgiving with the thresholds used when doing corrections as we have an extra layer to the pipeline ensuring the corrections make contextual sense. To achieve this a huggingface sentence-transformer is used to get sentence embeddings for each of the claims (*Reimers and Gurevych*, 2020). We then employ a method to compute the cosine similarity between the embedding tensors which gives us a score between 0 and 100 corresponding to similarity. In the case where the similarity between the original claim and the corrected claim is below a given threshold, the original claim will be kept instead of the corrected claim.

4.1.6 The Fourth Step: Verification

Finally, we have come to the end of the pipeline where the goal is to take the claims produced earlier in the pipeline and ascertain their veracity. To do this we use the MultiVerS model. The MultiVerS model is a model for scientific claim verification

using the longformer model as the encoder (*Wadden et al.*, 2022). Unlike Bert and Roberta mentioned in section 2.1.3, the longformer is able to work with larger than 512-token inputs, and thus performs better on data sets with larger inputs. The model is a multi-task model predicting veracity and rationale. If the model predicts a claim as "Not Enough Info", the rationale will be blank, but if the veracity is predicted as anything else e.i. supported or refuted, the model will also predict the rationale. The rationale is any sentence that is considered responsible for making the decision either way.

The model was pre-trained on in-domain claim verification data sets such as EVIDENCEINFERENCE, and PUBMEDQA, in addition to more general-domain data sets such as FEVER 4.1.2. As for fine-tuning, SciFact, CovidFact, and HealthVer were used for training (*Wadden et al.*, 2022). Furthermore, the model also offers a few model-checkpoints depending on the prediction tasks the user wishes to attempt. The "fever-checkpoint" is for wikipedia or general text verification, "healthver-checkpoint" is for claims about covid-19, and "scifact-checkpoint" is for biomedical claims (*Wadden*, 2023). For our purposes, the FEVER-checkpoint was used. Overall, the model performs better than other state-of-the-art claim verification models such as VERT5ERINI, and GRAPHJOINT on verification tasks (*Wadden et al.*, 2022).

The MultiVerS model needs the data to be formatted into a JSON-lines format containing claims and a list of corresponding evidence ids. In a separate JSON-lines file called "corpus" the equivalent evidence ids and their evidence sentences can be retrieved. Both these files are given to the MultiVerS model as input and following this an output-file is created containing each claims veracity in the form of "Supported" or "Contradicted", in addition to the relevant evidence sentence(s) (rationales) for the given prediction. However, many of the claims will come back as neither, which for the purposes of this project is classed as "Indeterminate", and refers to where the model was not able to make a prediction.

The process of splitting the claims into claim and corpus files is done twice, first for all the supported claims and second for the refuted claims. This is done to be able to ascertain the full pipeline's performance on the two types of claims we are working with; supported and refuted. We want to be able to see how many refuted claims are predicted as supported after going through the pipeline while also seeing how any supported claims are predicted as contradicted (*Note: refuted and contradicted are essentially referring to the same veracity. One is given by the FEVER data set "refuted" the other is returned as predictions from the MultiVerS model "Contradicted"*). This gives us some insight into how well the model performs.

4.1.7 Logistic Regression and Random Forest Models

Somewhat outside the scope of the pipeline both a logistic regression and random forest classifier were also implemented with the goal of creating a model for identifying high-quality questions. The goal of these models would be to take the questions generated for each claim during the QG-step and any information from the following steps to determine what makes up a good question. Both models were trained using features gathered as a result of the down-stream tasks of the pipeline process which include;

- Word count for questions, both answers, and evidence.

- Named entity count of the type; Person, Organisation, Location, and Miscellaneous extracted form questions, both answers, and evidence.
- Question type split into what, who, and other type questions.
- Answer similarity (The same as the one determined in the correction step using fuzzy string matching)
- Cosine similarity between the original and corrected claim (The same as the optional step within the correction part of pipeline).

For the named entity extraction a Roberta-large model fine-tuned for NER on the conll2003 data set was used (*Polle*, 2023). This model achieved an f1-scores of about 97 on the original data set validation, and about 81 on other private NER data sets. The model extracts miscellaneous entities (MISC), a Persons name (PER), an organization (ORG), a location (LOC) and words outside of the named entity space (O) which we ignore for our purposes.

Finally, the label to predict using these features were either 1 or 0. 1 in the case of the MultiVerS model predicting "Supported" and 0 if predicting "Contradicted". This meant that all questions for a claim resulting in an "Indeterminate" prediction were not used. All in all, there were 26 features and one label to predict. The labels provided by the MultiVerS verifications were the same label predictions when no thresholds were used. This means all claims were corrected in some way, although for a number of claims, this meant they were corrected into the exact same claim as their original. Following this feature identification, all numerical features were scaled between 0 and 1. While the categorical features of which there was one e.i. question-type were kept as is.

The total questions contained within the data set used for training and testing were 5336 all derived from the FEVER dev data set using our pipeline. This was split into an 80/20 split where the train set contained 4249 questions 3236 of which were supported (1-labeled) and 1013 were refuted (0-labeled). The 20% remaining questions were part of the test set which included a total of 1087 questions 840 of which were supported and 247 refuted. Group shuffling was used to create the 80/20-split where questions for the same claim were kept within the same data set.

For the training of both the logistic regression classifier and random forest classifier, some form of feature selection, cross-validation and hyperparameter tuning was used. For feature selection, sklearns "SelectFromModel" method was used. As for cross-validation and hyperparameter tuning sklearnss GridSearchCV method was used with 10 folds focusing scoring on f1_macro. The reason for focusing on f1_macro was due to the inherent unbalance of the samples in the data set, and a need for our model to perform well on identifying all classes independent of how many samples each class has.

For logistic regression the features tested using gridsearch were; The penalty parameter for which we tested values of, None, l1, and l2 as potential penalties. Furthermore, we tested differing solvers which included, newton-cg, lbfgs, liblinear, sag and saga. In addition to these parameters, we also tested using different class_weights, in the form of "None" and Balanced, but refrained from using "None" after some testing provided undesired results. Finally, these parameters were all tested multiple times using three

different random states in the form of 0, 7, and 42. The best results were achieved by using class_weight "balanced", penalty of "None", random_state of 0 and the "lbfgs" solver.

As for the random forest classifier, max_depths of 10 to 50 was tested in increments of 10, before noticing that testing with increments of 1 between 5 to 10 performed better. n_estimators of 100, 200, and 300 were tested and class_weight was always set to balanced due to poor results using "None". Finally, these parameters were tested using 5 different randomly selected random states of 0, 2, 7, 13, and 42. In the end, the hyperparameters that were chosen in the end were max_depth: 9, with n_estimators: 100, and random_state of 13.

4.2 Evaluation

In this section, I will describe how the results in chapter 5 will be evaluated. A very brief idea of this was given in the previous subsection 4.1.6 on the implementation of the verification step. First, we will look at baseline veracity results. These are the results when running all the originally supported and refuted claims and their corresponding evidence through the MultiVerS model. After this, we will do the same for the claims that have gone through our pipeline and compare the results to the baseline results. We will look at the number of refuted claims that are predicted as supported for both the original and corrected claims, to give insight into whether or not the corrections are working as intended by correcting refuted claims. The same will also be done when looking at how many of the supported claims are predicted as refuted compared to the baseline to get an idea if the corrections are correcting already supported claims which is not the goal.

Furthermore, we will also evaluate question generation in relation to claims as part of the results. By comparing the types and length of questions, in addition to the types of entities in questions using methods such as bar charts and histograms. We can look into the possible aspects that make a given question better than another in the context of factual error correction. We can compare questions that result in successful corrections and unsuccessful corrections e.i. questions that lead claims that are originally refuted to becoming supported and vice versa. The same will also be done for the down-stream-processes by looking at answer pairs and correction similarity. Features of the evidence will also be investigated as the evidence is important in general as it affects multiple parts of the pipeline in both question-answering and verification.

Finally, we will evaluate the logistic regression and random forest classifiers. Both will be evaluated on how well they are able to perform when identifying both classes in terms of questions leading to supported and refuted claims. For this purpose, we will look at f1-macro and to some degree accuracy when determining the best versions of the models. In addition, to further investigate the importance of features that lead to certain veracity predictions we will look at feature selection and what features the models determine are the most relevant.

Chapter 5

Results

In this chapter, the results of the work carried out in this project will be analyzed and presented. The chapter will first look at the results of correction and how the veracity of refuted and supported claims are impacted in section 5.1. Next, the chapter will take a deep dive into how the question generation and question answering impact the performance of the pipeline and which features provide the best results in section 5.2, 5.3 and 5.4. Finally, we will present the results of our logistic regression and random forest classifier and discuss the results in general while also looking at how these results might inform our opinion on the question generation task as a whole in section 5.5.

5.1 Correction Results

Having now implemented the proposed pipeline this section will look to evaluate and analyze the correction results of the pipeline. This will be done in two main ways, a broad evaluation 5.1.1 of the results shown in table 5.1 and 5.2, in addition to a more in-depth 5.1.2 look at specific differences between the different thresholding methods and the individual niches between them. For the in-depth analysis, specific examples of corrections will be presented and discussed as a way to visualize these differences.

	Supported					
	Supports	**Change**	**Contradict**	**Change**	**Indeterminate**	**Change**
Original	1285 (38.55%)		41 (1.23%)		2007 (60.22%)	
fuzz75	1260 (37.80%)	-1.94%	53 (1.59%)	+29.26%	2020 (60.60%)	+0.65%
fuzz85	1274 (38.22%)	-0.85%	53 (1.59%)	+29.26%	2006 (60.18%)	-0.05%
fuzz100	**1399 (41.97%)**	**+8.87%**	**44 (1.32%)**	**+7.32%**	**1890 (56.70%)**	**-5.83%**
fuzz75/cos70	1351 (40.53%)	+5.14%	51 (1.53%)	+24.39%	1931 (57.94%)	-3.79%
fuzz85/cos70	1366 (40.98%)	+6.30%	50 (1.50%)	+21.96%	1917 (57.51%)	-4.48%

Table 5.1: Table showing veracity of originally supported claims after going through the MultiVerS verification model. The numbers shown refer to total claims predicted as supporters, contradict and indeterminate with a corresponding percentage. Change refers to % increase for the corrected predictions compared to original predictions. Best values are shown as bold text.

	Refuted					
	Supports	**Change**	**Contradict**	**Change**	**Indeterminate**	**Change**
Original	56 (1.68%)		1341 (40.23%)		1936 (58.09%)	
fuzz75	772 (23.16%)	+1278.57%	700 (21%)	-47.80%	1861 (55.84%)	-3.87%
fuzz85	806 (24.18%)	+1339.29%	680 (20.40%)	-49.30%	1847 (55.42%)	-4.60%
fuzz100	**889 (26.67%)**	**+1487.50%**	**670 (20.10%)**	**-50.04%**	1774 (53.23%)	-8.37%
fuzz75/cos70	522 (15.66%)	+832.14%	1033 (30.99%)	-22.97%	1778 (53.35%)	-8.16%
fuzz85/cos70	549 (16.47%)	+880.36%	1018 (30.54%)	-24.09%	**1766 (52.99%)**	**-8.78%**

Table 5.2: Table showing veracity of originally refuted claims after going through the MultiVerS verification model. The numbers shown refer to total claims predicted as supporters, contradict and indeterminate with a corresponding percentage. Change refers to % increase for the corrected predictions compared to original predictions. Best values are shown as bold text.

5.1.1 Broad Evaluation

The results shown in both tables 5.1 and 5.2 are the resulting prediction when using the MultiVerS model for claim verification on the 3333 supported and 3333 refuted claims respectively. The results are composed of first running the original "FEVER" claims through the verification model before doing the same with our corrected claims. Multiple thresholds for the "Levenshtein Distance" mentioned in section 4.1.5 were tested, starting with a threshold of 100 shown as fuzz100 in the tables. A threshold of 100 is the same as no threshold with the assumption that any threshold lower than 100 will increase how strict our corrections are.

In addition to threshold = 100 both 85 and 75 were the next thresholds tested. 75 was chosen as the strictest threshold since anything lower than 75 was determined to be too strict, as it did not allow for claims where a given year was incorrect to be corrected in the case of a single-digit difference in the year-date. 85 was chosen as a midway point between 100 and 75 to better visualize the effects of stricter versus more lenient thresholds.

After having tested both 75 and 85, we added to the complexity of our thresholding by including another criterion for correction, namely cosine similarity between the original claims embeddings and the corrected claim. The value chosen here was cos70 based on a manual inspection of the effects of using higher and lower values on a small subset of the data.

Now having an understanding of the thresholds chosen and the layout of our result tables, we can start by analyzing and discussing the first column in table 5.1 **"Supported/Supports"**. Here we can immediately see the effects of thresholding. The stricter the threshold was set the worse our corrected claim predictions performed compared to the original of not correcting anything. No threshold produced the best results of +8.87% compared to the original data while the strictest option of fuzz75 produced worse results in a negative trend of -1.94% compared to the original. The assumption is that by adding a threshold to the correction we decrease the number of questions taking part in our corrections which in this case leads to worse results. On the other hand, we can see that the inclusion of contextual similarity in the form of fuzz75/cos70 and fuzz85/cos70 improves our results compared to only using the base threshold of fuzz75 and fuzz85, but does not perform better than no threshold fuzz100. The assumption

here is that the no-threshold (fuzz100) method produces corrections for every claim based on the corresponding answers using evidence. Furthermore, by including the contextual similarity rules we ignore corrections that are too far outside the norm leaning to a balance where we correct many claims, yet only keep the contextually most accurate corrections leading to better results than just using base thresholds.

As for the **"Supported/Contradicted"** column in the data, we see little change between the different methods. This is due to the low amount of claims predicted as contradicted. The small amount of samples does lead to large increases percentage-wise from just a few predictions, seeing as some changes are as high as +29.26%. However, all in all, the changes here are less relevant when determining the viability of our methods. Same as what the previous column discussed, fuzz100 produces the best results with a +7.32% (3 total claims) increase in contradicted claims compared to the original claims which are still antithetical to our goals of minimizing contradicted claims. Furthermore, the use of contextual similarity also produces better results than just the baseline thresholding methods, however, the difference is very slight.

Finally, we can look at the claims that are not able to be classified by the MultiVerS model as either supports or contradicts. For the column of **"Supported/Indeterminate"**, we do not see much difference in terms of the performance between the different methods, with fuzz100 producing the best results and the contextual-based methods being the next best. However, the changes as compared to the original claims do show a positive result. For the combined contextual similarity and thresholding methods, and the no threshold method there is a decrease in the number of claims that are predicted as indeterminate which is a positive outcome as a whole. The more claims that are determinable the better we can consider the performance of our methods.

Now we can start to discuss the more interesting part of the results namely how the refuted claims look after having been corrected compared to the original FEVER claims. **"Refuted/Supports"** refers to the column we want to maximize the most with our corrections. As has been a common theme so far within this section the method of fuzz100 performs the best with an increase of +1487.50% when compared to the original claims. The reason for such a large increase is mostly due to the low number of original claims that are predicted as supported, however with a total of 833 claims being corrected the results are still good. Contrary to how the performance was on supported claims using contextual methods, for the refuted claims the contextual methods perform worse than the base thresholding. We can observe a general increase in performance the more lenient our thresholds are, while the context-based methods perform the worst overall. Still, all our methods do perform some corrections with the implication that the more we restrict the corrections by implementing thresholds or conditions the worse our results will be.

Looking at the next column **"refuted/contradict"** the case is the same as before. Fuzz100 performed the best while fuzz75 and fuzz85 performed the next best, with the contextual methods being a bit further behind comparatively. The model's performance was quite good seeing as at its best we were able to reduce the amount of "contradict" predictions by -50.04%, meaning around half of all contradicted claims could be corrected in some way.

For our last column of "refuted/indeterminate" there is not much more to comment on other than fuzz75 and fuzz85 performing the worst compared to fuzz100, fuzz75/cos70 and fuzz85/cos70 where fuzz85/cos70 performed the best at -8.78%.

However the difference between fuzz100, fuzz75/cos70 and fuzz85/cos70.

Generally speaking, when we look at these results with the goal in mind of minimizing the negative effects on supported claims and maximizing the effects on refuted claims. The best method overall looks to be fuzz100 where we use no thresholding and correct all claims using all the questions and evidence-based answers. Implying that the methods with fewer restrictions perform the best when correcting claims.

5.1.2 In-depth Evaluation

To start with this section we will first discuss some examples of how the different methods of correction impact refuted claims before looking at some more examples of supported claims. We will discuss what successful corrections are, and what types of mistakes our pipeline is making by inspecting some examples from our data.

Method	Claim	Veracity
Original	Murda Beatz's real name is Marshall Mathers.	REFUTES / INDETERMINATE
fuzz75	Murda Beatz's real name is Shane Lee Lindstrom.	SUPPORT
fuzz85	Murda Beatz's real name is Shane Lee Lindstrom.	SUPPORT
fuzz100	Murda Beatz's real name is Shane Lee Lindstrom.	SUPPORT
fuzz75_cos70	Murda Beatz's real name is Shane Lee Lindstrom.	SUPPORT
fuzz85_cos70	Murda Beatz's real name is Shane Lee Lindstrom.	SUPPORT

Figure 5.1: Showing original claim and corresponding corrections and veracity based on the method of thresholding. Original claim veracity is given as Fever-label / MultiVerS prediction

Method	Claim	Veracity
Original	Matteo Renzi is German.	REFUTES / CONTRADICT
fuzz75	Matteo Renzi is an Italian politician.	SUPPORT
fuzz85	Matteo Renzi is an Italian politician.	SUPPORT
fuzz100	Matteo Renzi is an Italian politician.	SUPPORT
fuzz75_cos70	Matteo Renzi is an Italian politician.	SUPPORT
fuzz85_cos70	Matteo Renzi is an Italian politician.	SUPPORT

Figure 5.2: Showing original claim and corresponding corrections and veracity based on the method of thresholding. Original claim veracity is given as Fever-label / MultiVerS prediction

In figure 5.1, we can see an example of correction working as intended. "Murda Beatzs real name is Marshall Mathers" is the original claim that is labeled as refuted by the Fever data set. All our methods, in this case, make the same corrections over

the entity of "Marshall Mathers", and correct this to "Shane Lee Lindstrom". Using the MultiVerS model we can see that the original claims veracity is predicted as "INDETERMINATE" which is not ideal seeing as it is a refuted claim. However, the corrected claims are all predicted as "SUPPORT" which is the best case. For the "INDETERMINATE" prediction it is the case that the MulitVerS model was not able to find supporting or contradicting rationales in our evidence. On the other hand figure 5.2 shows a similar good correction where the veracity of the original claim is successfully identified.

Method	Claim	Veracity
Original	Brad Wilk helped co-found Rage in 1962.	REFUTES / INDETERMINATE
fuzz75	Tom Morello and Zack de la Rocha Wilk helped co-found Rage in August 1991.	SUPPORT
fuzz85	Tom Morello and Zack de la Rocha Wilk helped co-found Rage in August 1991.	SUPPORT
fuzz100	Tom Morello and Zack de la Rocha Wilk helped co-found Rage in August 1991.	SUPPORT
fuzz75_cos70	Tom Morello and Zack de la Rocha Wilk helped co-found Rage in August 1991.	SUPPORT
fuzz85_cos70	Tom Morello and Zack de la Rocha Wilk helped co-found Rage in August 1991.	SUPPORT

Figure 5.3: Showing original claim and corresponding corrections and veracity based on the method of thresholding. Original claim veracity is given as Fever-label / MultiVerS prediction

Another type of situation found within the corrections can be seen in figure 5.3. In this case, all the corrections are somewhat right yet, also a bit deceptive. The desired corrections would be to turn the claim "Brad Wilk helped co-found Rage in 1962" into "Brad Wilk helped co-found Rage in August 1991". Only changing the date is required to turn this claim into a corrected claim, however, instead, our correction methods also correct "Brad Wilk" into "Tom Morello and Zack de la Rocha". All three people are members and founders of Rage Against the Machine, and as such this is a case of our system overcorrecting, seeing as "Brad Wilk" would have been correct. In addition, we can see the inclusion of the last name "Wilk" in all corrections which is not desired. While the corrections are still predicted as "SUPPORT" the addition of "Wilk" may lead a potential user of this system to be confused by the correction which is not ideal.

```
Question 1:  When did Brad Wilk co-found Rage?
Claim Answer:  1962
Wikidata Answer:  August 1991
Simillarity Score:  27

Question 2:  Who helped co-founded Rage in 1962?
Claim Answer:  Brad Wilk
Wikidata Answer:  Tom Morello and Zack de la Rocha
Simillarity Score:  24

Question 3:  What was the name of Wilk's co-founder?
Claim Answer:  Brad
Wikidata Answer:  Tom Morello and Zack de la Rocha
Simillarity Score:  11
```

Figure 5.4: Questions, Answers, and similarity scores between answers used in corrections of claims in figure 5.3

As for why "Wilk" is still included in the correction, it has to do with how our

question-answering model extracted the answer spans from the claim itself. The two questions relating to "Brad Wilk" as seen in figure 5.4 are; "Who helped co-founded Rage in 1962?", and "What was the name of Wilk's co-founder?". The first of the two questions perform as desired, by extracting the full name of "Brad Wilk" while the second question only extracted "Brad" leaving out "Wilk". When comparing the confidence scores (not the same as the similarity score shown in figure 5.4) of both the answers using wikidata sentences as evidence, the second answer had a much lower score compared to the third answer, due to answering different questions, and as such we end up with only the span of "Brad" being replaced with the wikidata answer.

Method	Claim	Veracity
Original	I Kissed a Girl was only recorded by Donald Trump.	REFUTES / CONTRADICT
fuzz75	I Kissed a Girl was only recorded by Katy Perry.	CONTRADICT
fuzz85	I Kissed a Girl was only recorded by Katy Perry.	CONTRADICT
fuzz100	I Kissed a Girl was only recorded by Katy Perry.	CONTRADICT
fuzz75_cos70	I Kissed a Girl was only recorded by Katy Perry.	CONTRADICT
fuzz85_cos70	I Kissed a Girl was only recorded by Katy Perry.	CONTRADICT

Figure 5.5: Showing original claim and corresponding corrections and veracity based on the method of thresholding. Original claim veracity is given as Fever-label / MultiVerS prediction

Wikidata Evidence (1*): `` I Kissed a Girl '' is a song recorded by American singer Katy Perry for her second studio album , One of the Boys -LRB- 2008 -RRB- .
*Number of sentences

Figure 5.6: Showing corresponding evidence for claim corrections in figure 5.5

Continuing on it is also important to understand that the model we are using for verification has its limitations. In figure 5.5, we can see that the claim "I Kissed a Girl was only recorded by Donald Trump." is corrected to "I Kissed a Girl was only recorded by Katy Perry." Based on the evidence as seen in figure 5.6 you would expect such a claim to be true, however, this is not the case here. While the reasoning for why this is predicted as "CONTRADICT" is hard to know, such misclassifications are possible when using the MultiVerS model and it is important to understand that the model is not a be-all and end-all for veracity identification.

Moving a bit away from looking at the differences between the methods themselves, figure 5.7 shows an example of why fuzz100 might perform better in some correction cases compared to the other methods. While the other methods deem that no correction is needed (NCN) because the similarity of the answers are too close (see figure 5.8), fuzz100 makes such correction anyways, as we discussed in the broad evaluation earlier in section 5.1.1. The same can be observed in figure 5.9, with the same positive results.

However, fuzz100 is not always better at making corrections. Figure 5.10 shows how making corrections using the evidence-based answer on every single claim answer

Method	Claim	Veracity
Original	John Dolmayan was born on July 15, 1873.	REFUTES / CONTRADICT
fuzz75	NCN	CONTRADICT
fuzz85	NCN	CONTRADICT
fuzz100	John Hovig Dolmayan was born on July 15 , 1973.	SUPPORT
fuzz75_cos70	NCN	CONTRADICT
fuzz85_cos70	NCN	CONTRADICT

Figure 5.7: Showing original claim and corresponding corrections and veracity based on the method of thresholding. Original claim veracity is given as Fever-label / MultiVerS prediction

Question 1: Who was born on July 15, 1873?
Claim Answer: John Dolmayan
Wikidata Answer: John Hovig Dolmayan
Simillarity Score: 100

Question 2: When was John Dolmayan born?
Claim Answer: July 15, 1873
Wikidata Answer: July 15 , 1973
Simillarity Score: 92

Figure 5.8: Questions, Answers, and similarity scores between answers used in corrections of claims in figure 5.7

Method	Claim	Veracity
Original	T2 Trainspotting is a 2017 British comedy drama book.	REFUTES / CONTRADICT
fuzz75	NCN	CONTRADICT
fuzz85	NCN	CONTRADICT
fuzz100	T2 Trainspotting is a 2017 British comedy drama film.	SUPPORT
fuzz75_cos70	NCN	CONTRADICT
fuzz85_cos70	NCN	CONTRADICT

Figure 5.9: Showing original claim and corresponding corrections and veracity based on the method of thresholding. Original claim veracity is given as Fever-label / MultiVerS prediction

span, can lead to some unnecessary corrections. As with the "Brad Wilk" example, here "Queen" has been corrected to "Queen are a British rock band that formed in London in 1970 ." and "a Canadian rock band" replaced by "a British rock band". In addition, we have the entity artifact of "(band)" being out of place in the correction same as in the "Wilk" example. Looking at the corrections one can see how the other methods corrections are better and more precise compared to the original claim. It is also interesting to note that while all corrections are correct to some degree the MultiVerS model

is only able to ascertain the veracity of the fuzz100 claim correction as "SUPPORT", again showing some limitations with the model.

Method	Claim	Veracity
Original	Queen (band) is a Canadian rock band.	REFUTES / INDETERMINATE
fuzz75	Queen (band) is a British rock band.	INDETERMINATE
fuzz85	Queen (band) is a British rock band.	INDETERMINATE
fuzz100	Queen are a British rock band that formed in London in 1970 . (band) is a British rock band.	SUPPORT
fuzz75_cos70	Queen (band) is a British rock band.	INDETERMINATE
fuzz85_cos70	Queen (band) is a British rock band.	INDETERMINATE

Figure 5.10: Showing original claim and corresponding corrections and veracity based on the method of thresholding. Original claim veracity is given as Fever-label / MultiVerS prediction

Lastly, when it comes to the refuted claim corrections we will look at the intuition behind what the cosine similarity method does. It is important to understand that for fuzz75_cos70 and fuzz85_cos70 the job of these methods are to determine if the correction made by fuzz75 or fuzz85 are suitable corrections. In the cases where we are looking at refuted claims, cosine methods will never make better predictions than just base thresholding. This is due to the fact that their only function is in determining if a correction is contextually similar to the original claim. As seen in figure 5.11 here we have a claim where both the fuzz75 and fuzz85 have made some bad corrections, and when comparing these to the original claim the cosine methods have decided that no correction was needed. In the worst case the strictness of this method will lead to suitable corrections such as the one in figure 5.12 being deemed as no corrections needed. While the last example is undesired, the benefits of using cosine are better shown when looking at claims which were originally supported.

Method	Claim	Veracity
Original	Weekly Idol is hosted by Yoo Jae Suk.	REFUTES / INDETERMINATE
fuzz75	Jeong Hyeong-dong-don.	INDETERMINATE
fuzz85	Jeong Hyeong-dong-don.	INDETERMINATE
fuzz100	Jeong Hyeong-dong-don.	INDETERMINATE
fuzz75_cos70	NCN	INDETERMINATE
fuzz85_cos70	NCN	INDETERMINATE

Figure 5.11: Showing original claim and corresponding corrections and veracity based on the method of thresholding. Original claim veracity is given as Fever-label / MultiVerS prediction

When it comes to looking at how corrections affect supported claims there is not much too much to comment on. The type of corrections we want to see is "NCN" for all but the fuzz100 method, and for the fuzz100 to have a correction that is the same or very similar to the original claim. Figure 5.13 shows where this is the case. Furthermore as shown with the refuted claims previously, the MultiVerS Model is not always able to

Method	Claim	Veracity
Original	Kenneth Lonergan is the director of Pacific Rim.	REFUTES / CONTRADICT
fuzz75	Guillermo del Toro is the director of Pacific Rim.	SUPPORT
fuzz85	Guillermo del Toro is the director of Pacific Rim.	SUPPORT
fuzz100	Guillermo del Toro is the director of Pacific Rim.	SUPPORT
fuzz75_cos70	NCN	CONTRADICT
fuzz85_cos70	NCN	CONTRADICT

Figure 5.12: Showing original claim and corresponding corrections and veracity based on the method of thresholding. Original claim veracity is given as Fever-label / MultiVerS prediction

Method	Claim	Veracity
Original	Excuse My French is the debut album of an artist.	SUPPORTS / SUPPORT
fuzz75	NCN	SUPPORT
fuzz85	NCN	SUPPORT
fuzz100	Excuse My French is the debut album of an artist.	SUPPORT
fuzz75_cos70	NCN	SUPPORT
fuzz85_cos70	NCN	SUPPORT

Figure 5.13: Showing original claim and corresponding corrections and veracity based on the method of thresholding. Original claim veracity is given as Fever-label / MultiVerS prediction

Method	Claim	Veracity
Original	Down With Love is a 2003 comedy film.	SUPPORTS / CONTRADICT
fuzz75	NCN	CONTRADICT
fuzz85	NCN	CONTRADICT
fuzz100	Down With Love is a 2003 comedy film.	CONTRADICT
fuzz75_cos70	NCN	CONTRADICT
fuzz85_cos70	NCN	CONTRADICT

Figure 5.14: Showing original claim and corresponding corrections and veracity based on the method of thresholding. Original claim veracity is given as Fever-label / MultiVerS prediction

correctly predict supported claims as seen in figure 5.14. One suspected reason in this instance is that based on the corresponding evidence for the claims as seen in figure 5.15 the model differentiates between concepts such as comedy and romantic comedy, seeing them as mutually-exclusive when this is not necessarily true.

Wikidata Evidence (1*): Down with Love is a 2003 romantic comedy film .
*Number of sentences

Figure 5.15: Showing corresponding evidence for claim corrections in figure 5.14

Method	Claim	Veracity
Original	Christian Gottlob Neefe was an opera writer.	SUPPORTS / SUPPORT
fuzz75	5 February 1748 -- 28 January 1798	INDETERMINATE
fuzz85	5 February 1748 -- 28 January 1798	INDETERMINATE
fuzz100	Christian Gottlob Neefe was an opera writer.	SUPPORT
fuzz75_cos70	NCN	SUPPORT
fuzz85_cos70	NCN	SUPPORT

Figure 5.16: Showing original claim and corresponding corrections and veracity based on the method of thresholding. Original claim veracity is given as Fever-label / MultiVerS prediction

Looking at supported claims we also notice the benefits of using cosine similarity as a method. Figure 5.16 shows where the baseline methods have generated a wrong prediction that does not meet the contextual requirements compared to the original claim, and as such the cosine methods have determined that no correction was needed. Unlike what is the case with refuted claims, supported claims can benefit from stricter methods of correction seeing as it can avoid the already supported claims from being unintentionally corrected into incorrect or unidentifiable claims.

Method	Claim	Veracity
Original	Jack Falahee is a person who acts.	SUPPORTS / SUPPORT
fuzz75	Jack Falahee is an American actor.	SUPPORT
fuzz85	Jack Falahee is an American actor.	SUPPORT
fuzz100	Jack Falahee is an American actor.	SUPPORT
fuzz75_cos70	Jack Falahee is an American actor.	SUPPORT
fuzz85_cos70	Jack Falahee is an American actor.	SUPPORT

Figure 5.17: Showing original claim and corresponding corrections and veracity based on the method of thresholding. Original claim veracity is given as Fever-label / MultiVerS prediction

Finally, the last type of correction that will be discussed is the type of precision correction. As the term implies these are claims that might already be supported, but

Method	Claim	Veracity
Original	Trollhunters was produced by an animation company.	SUPPORTS / SUPPORT
fuzz75	Trollhunters was produced by an DreamWorks Animation.	SUPPORT
fuzz85	Trollhunters was produced by an DreamWorks Animation.	SUPPORT
fuzz100	Trollhunters was produced by an DreamWorks Animation.	SUPPORT
fuzz75_cos70	Trollhunters was produced by an DreamWorks Animation.	SUPPORT
fuzz85_cos70	Trollhunters was produced by an DreamWorks Animation.	SUPPORT

Figure 5.18: Showing original claim and corresponding corrections and veracity based on the method of thresholding. Original claim veracity is given as Fever-label / MultiVerS prediction

Method	Claim	Veracity
Original	Alice Cooper has been working in his field for over 50 years.	SUPPORTS / INDETERMINATE
fuzz75	Alice Cooper has been working in his field for five decades.	SUPPORT
fuzz85	Alice Cooper has been working in his field for five decades.	SUPPORT
fuzz100	Alice Cooper has been working in his field for five decades.	SUPPORT
fuzz75_cos70	Alice Cooper has been working in his field for five decades.	SUPPORT
fuzz85_cos70	Alice Cooper has been working in his field for five decades.	SUPPORT

Figure 5.19: Showing original claim and corresponding corrections and veracity based on the method of thresholding. Original claim veracity is given as Fever-label / MultiVerS prediction

the corrections correct them in such a way as to make them more precise. This is an added benefit in many cases for our correction methods and the pipeline as a whole, and while it was not an intended goal it is a welcome one. Figures 5.17, and 5.18 are examples where the original claim is supported and predicted as "SUPPORT", but some corrections are made in which the claims are still predicted as "SUPPORT" but they are more precise. There are also instances of precision corrections leading to claims predicted as "INDETERMINATE" by MultiVerS showing as "SUPPORT", e.i. figure 5.19. However, this points more to the MultiVerS model not being able to handle the claim rather than the precision correction being especially well formed.

5.2 Question/Evidence Analysis

Considering how important the questions we ask are for our resulting corrections, this section aims to understand more about the questions we generate, in addition to how some features of questions and corresponding evidence might affect the outcome of our corrections. We will investigate word counts for questions and corresponding wikidata evidence, entities within the generated questions and the same evidence, and finally which types of questions are asked. We will only be investigating the questions that either lead to a "SUPPORT" prediction or "CONTRADICT" prediction which totals 5336 questions corresponding to a total of 1443 originally supported claims and 1559

refuted claims, the rest being predicted "INDETERMINATE" and as such ignored, see section 5.1.1 for a recap of the distributions for fuzz100.

5.2.1 Question/Evidence Word Count Analysis

In this section we will start with a word count analysis of the generated questions, and evidence before investigating if there are any differences between the categories of "Supported/SUPPORT", "Supported/CONTRADICT", "refuted/SUPPORT", and "refute/CONTRADICT". "Supported/SUPPORT" being originally supported claims that are predicted as "SUPPORT", "Supported/CONTRADICT" being claims that originally were supported but were predicted as "CONTRADICT", etc.

Question Word Count

Starting off we look at word count-related features for questions. We start with a look at the data as a whole before going into the aforementioned comparisons. Firstly when looking at the data there are as mentioned a total of 5336 questions that each have an average of 6.8 total words. The question with the fewest words contains a total of 2 while the largest has a total of 27. While 6 was the median word count for the 50th percentile, and 8 was the 75th-percentile median count and what was used in determining the range of histograms generated. Figure 5.20 shows the total distribution of the questions' word count.

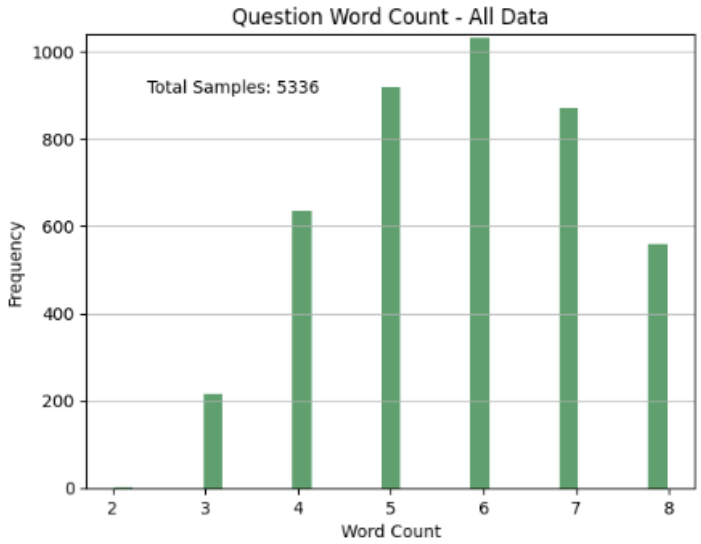

Figure 5.20: Histogram featuring word count distribution for Questions

Having looked briefly at the data as a whole, we will now compare the same measures for the correction categories. Figure 5.21 shows the different word count distributions for these categories. The takeaway from looking at the word counts is that our method of question generation is very consistent in how it generates questions across the data as a whole, with no correction combination leading to a vastly different distribution of counts. Only "Supported/CONTRADICT" had a different distribution towards the end of the plot, however, it is also a much smaller sample size and more prone to small differences within the data.

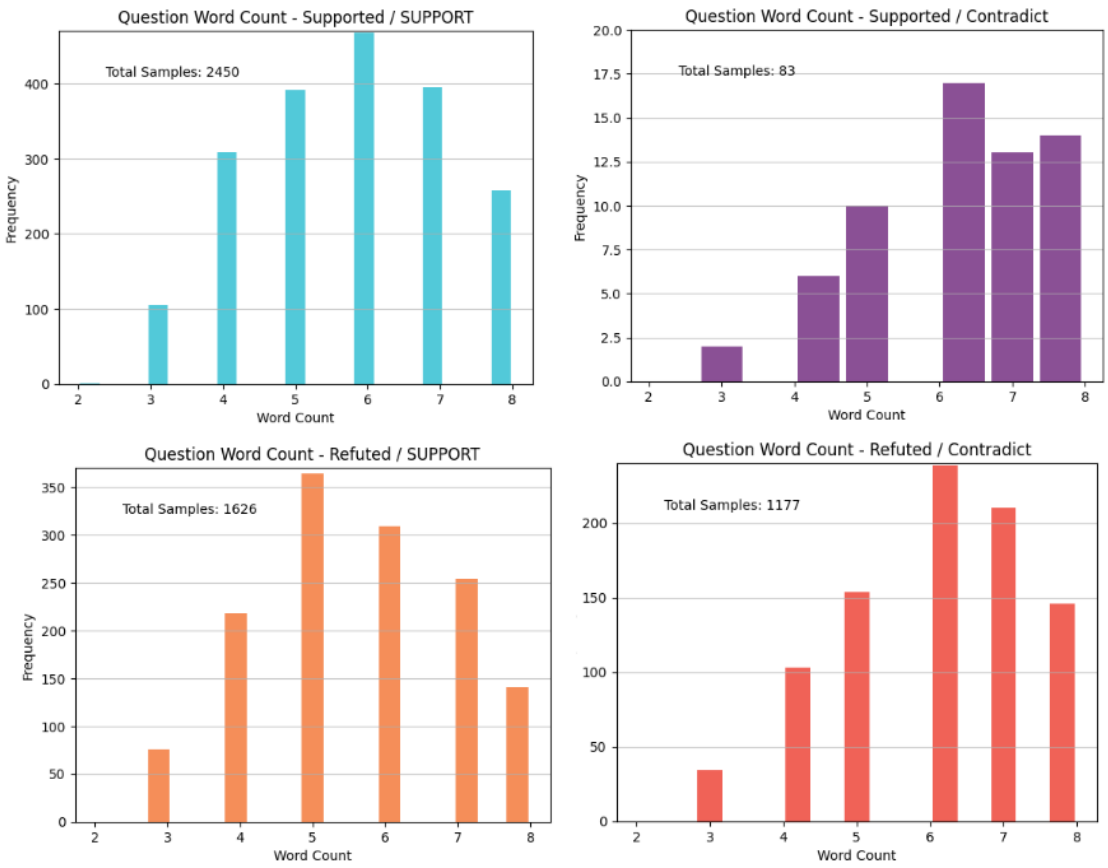

Figure 5.21: Histogram featuring word count distribution for Questions based of subsets of total data.

Evidence Word Count

On the other hand, the verbosity of the evidence might have more of an impact on corrections compared to question length and due to this possibility, we will also investigate the same for our wikidata evidence. Firstly the average word count is 52.8 words with a minimum of 7 words and a maximum of 1494 words. The high number of the maximum evidence word count comes from a claim with 52 corresponding evidence sentences which is outside the norm. The mean of the 50th percentile is 33 words and 164 for the 95th percentile. In figure 5.22 the distribution of all questions corresponding evidence is given within the range of 7 being the minimum and 164 was set as the max creating a 95% inclusion of the data.

Figure 5.23 shows the same type of comparisons as with question word count. We observe that all distributions follow the general distribution of the total data, and there is little to no difference between the distributions other than what can be attributed to sample size artifacts. As such we can conclude that the length of the questions and corresponding evidence have little to no impact on the resulting prediction down the line. One would maybe assume that more evidence would equal better predictions, but in the case of the FEVER data set all evidence is gathered by its relevance to the claim, and this likely makes both large and small pieces of evidence equally relevant to their corresponding claim.

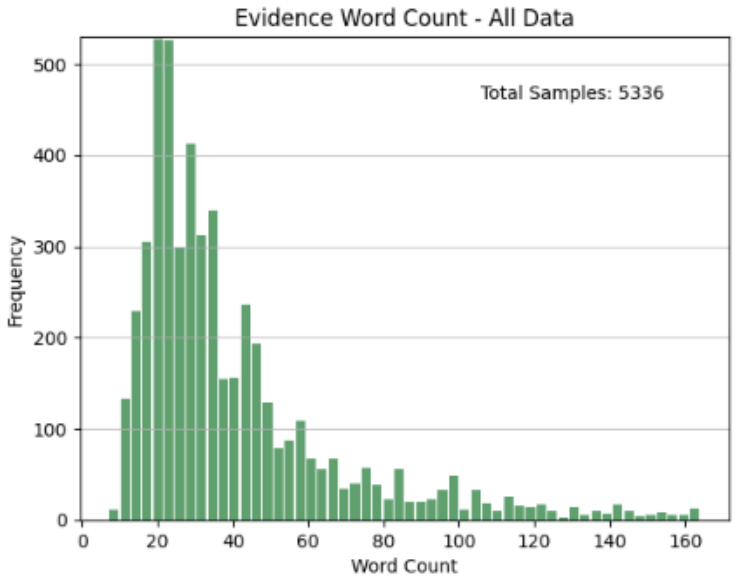

Figure 5.22: Histogram featuring word count distribution for Evidence

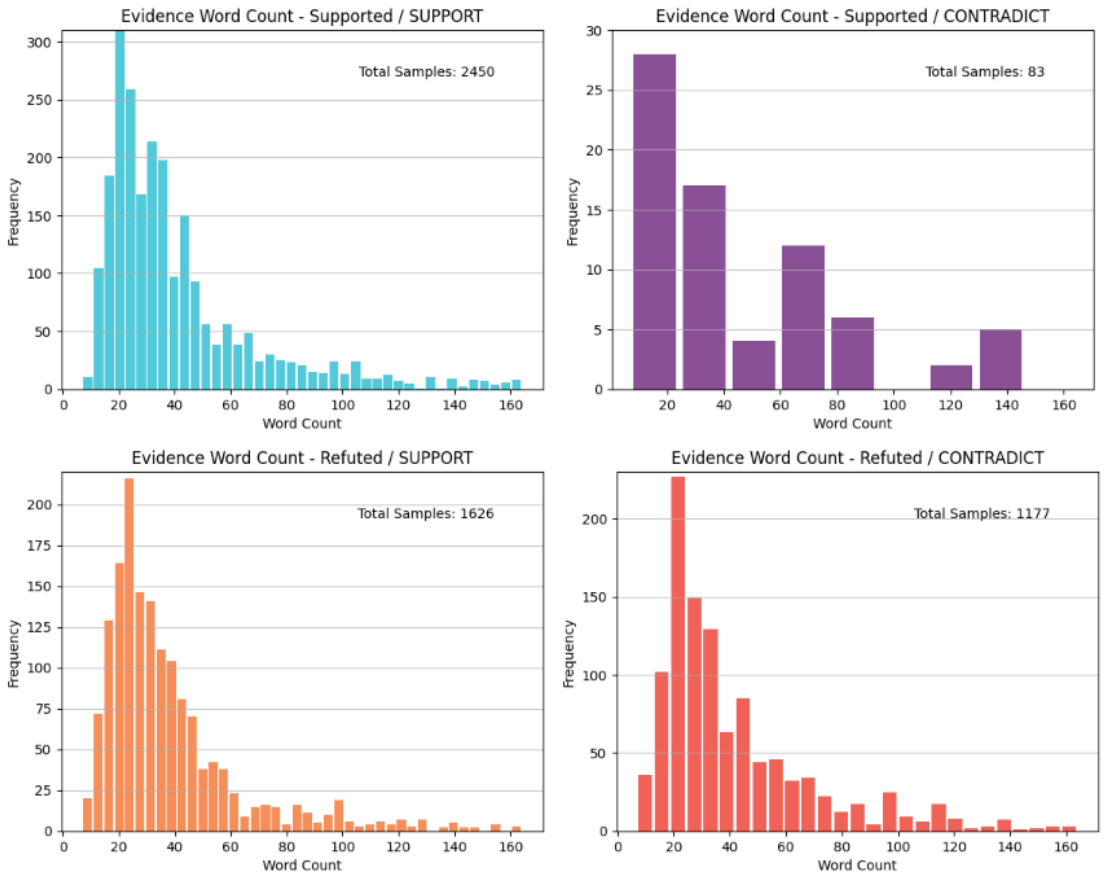

Figure 5.23: Histogram featuring word count distribution for wikidata evidence based of subsets of total data.

5.2.2 Question/Evidence Entity Analysis

Following our investigation into word counts we will do the same type of investigation into entities within the question and wikidata evidence text. The entities to investigate

are organization (ORG), person (PER), location (LOC), and miscellaneous (MISC), same as discussed in 4.1.7 for use with our logistic regression and random forest classifiers which will be discussed later in section 5.5.

Question Entities

When it comes to entities within the question text, there are very few instances present within the text. Most questions will have at most only one type of the same entity present, but as figure 5.24 shows the vast amount of questions will have no entities other than the one mentioned. In total, there are 1252 questions where no entity could be recognized, while the rest of 4084 had at least one or more. We also observe when an entity is present most of the time it will be of the type person, while in cases where multiple of the same entities are present, it is most common for the entity to be miscellaneous. Figure 5.24 also shows the distribution using a bar chart, where we can see the fact that most common entities are of the type person.

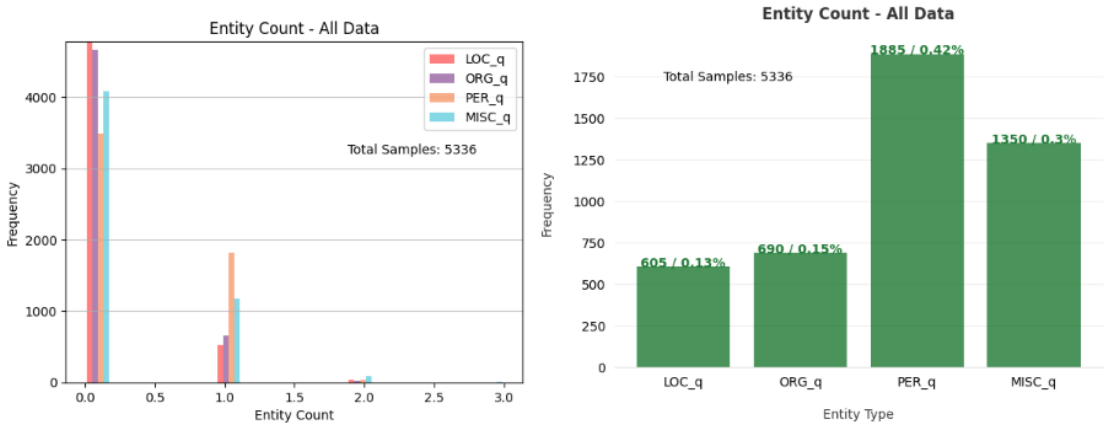

Figure 5.24: Histogram / Bar chart featuring entity count distribution for Questions for all data.

Using the same bar chart setup we can compare if there are any differences between resulting corrections the same way as we did for word counts. Figure 5.25 show all total and percentage values for the entities, and there appears to be little to no difference between them. It seems that the types of entities same as with word count have very little to do with the resulting predictions.

Evidence Entities

Finally, we will do the same with the evidence text, by identifying entities and comparing. From figure 5.26 we can see that evidence texts with only one entity of the same type tend to favor misc-entities, however as the entity count increases the more "person"-entities tend to go up. All in all, it is very close between the two as the bar chart in figure 5.26 shows, and both MISC and PER are extremely close in total occurrences.

As for when we do our comparisons, by investigating resulting corrections the same as before, there is little difference between the charts as seen in figure 5.27. The only difference of some significance appears between "Refuted/SUPPORT" and "Refuted/-CONTRADICT". In this case, it seems like evidence with either more location entities

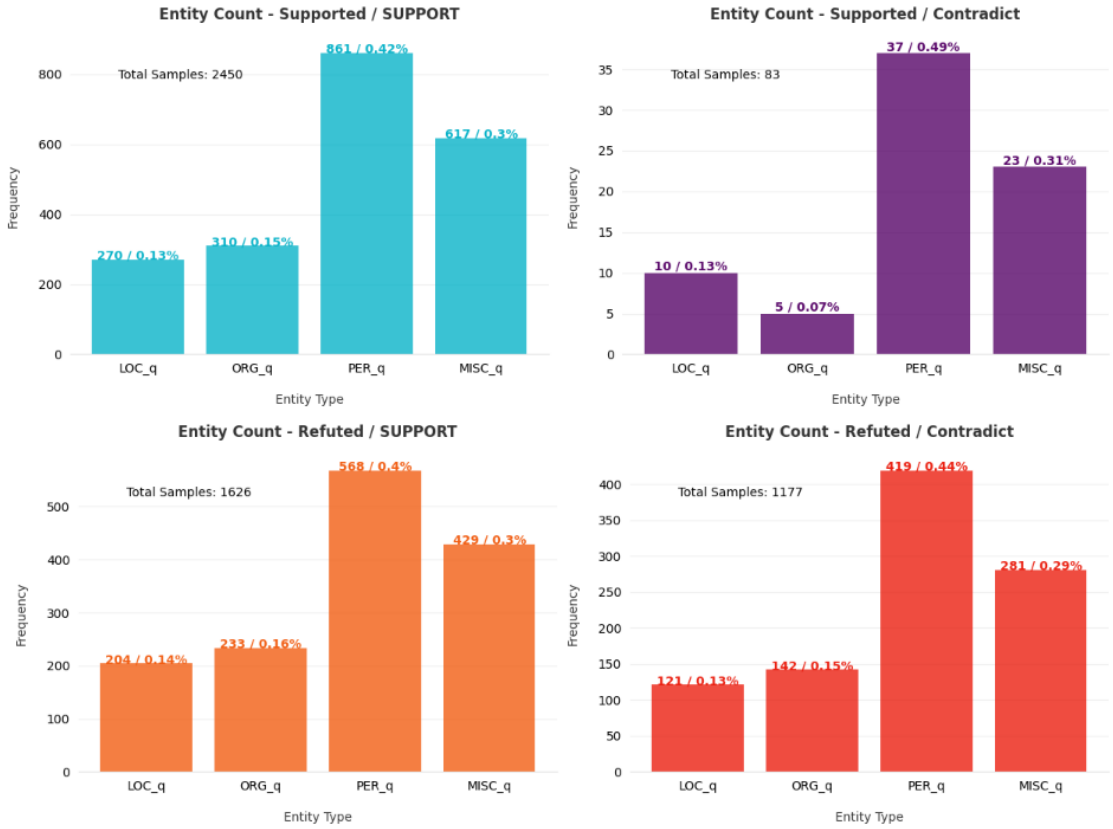

Figure 5.25: Bar chart featuring entity count distribution for Questions based of subsets of all data.

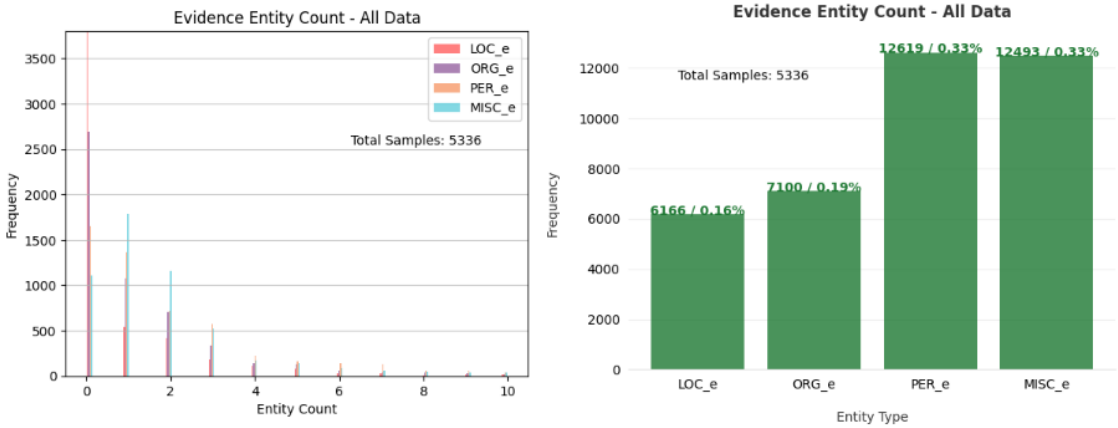

Figure 5.26: Histogram / Bar chart featuring entity count distribution for Evidence for all data.

or fewer MISC-entities leads to better results when correcting refuted claims although the differences are very slight, and might not be indicative of a broader trend.

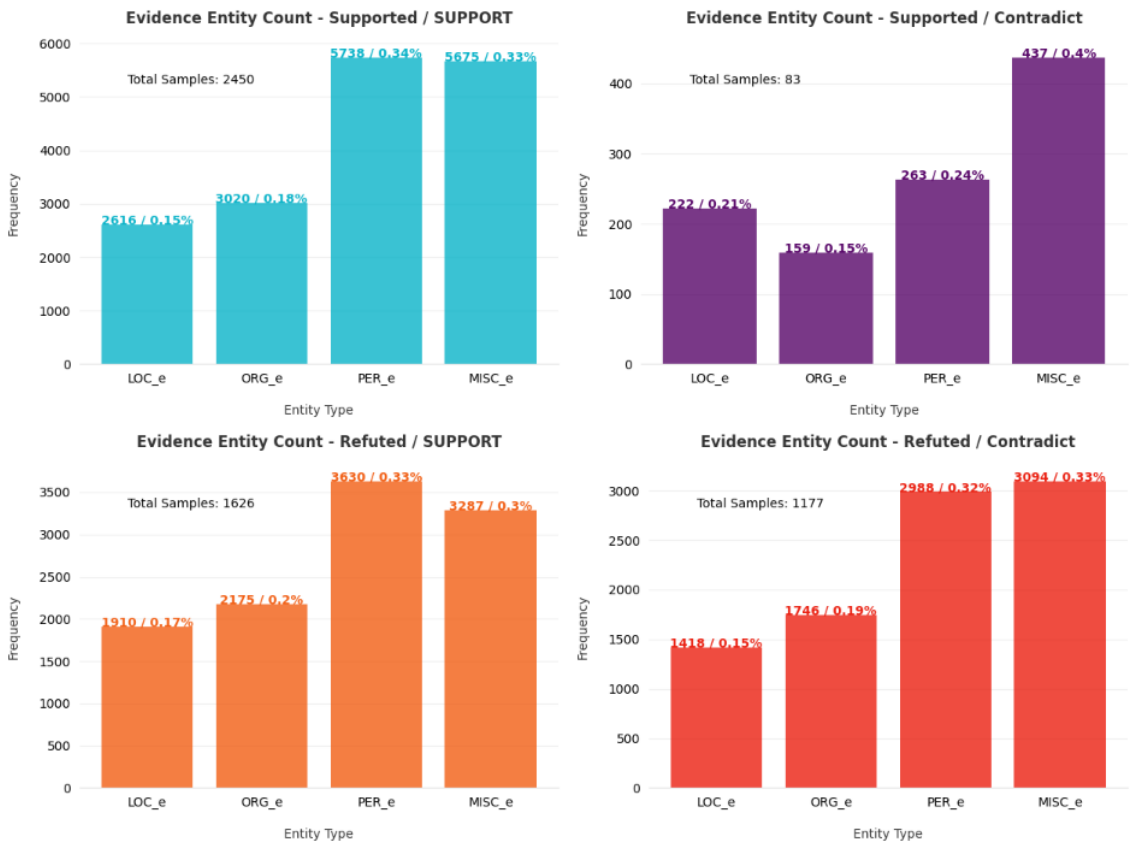

Figure 5.27: Bar chart featuring entity count distribution for Evidence based of subsets of all data.

5.2.3 Question Type Analysis

When generating questions our model generates a few different question types. What this section aims to investigate is whether or not the different types of questions we ask leads to different results. As for the question type the model prefers, there exists a clear winner in "what-type" questions. These are questions that start with the word what and can refer to questions about anything from time-related questions such as "What year did so and so win X-prize" or questions of "type" such as "What type of animal is X" etc. There is not a very clear line exactly what a "what-type" question is all the time. However for the next category of "who-type" questions it is easier to determine exactly what these types of questions entail. Who type questions for the most part involve some kind of reference to a person such as "Who won X-prize in 1999?". The aforementioned categories account for about 76% (49% + 27%) of all the generated answers when looking at the data set as a whole (this includes Indeterminate results).

As for the rest of the analysis, the remaining 24% have been grouped together as "other-type" questions. This broad category of "other" includes questions of the type; when, how, in, where, the, which, on, and different questions starting with some named entity such as a person or location-related information. The reason for these categories being grouped together was that as individual categories many of them represent a very small portion of the overall generated questions. They also often represent a more precise way of asking questions when compared to "what" and "who" -type questions. It is more likely that "when-type" questions are referencing some time or date, ver-

sus "what-type" questions which might be referencing both time or as mentioned a "what-category" type question. The same goes for "where-type" questions being about location in contrast again to what can be many different types of question subjects. "Who-type" questions are a bit more precise but not as precise as some of the questions in the "other-category" e.i. "Who was born in 1995?" vs. "John and which other member of X-group was born in 1995?". A question like the second might stop our question-answering model from answering "John", which is a possible answer for the first question given the case where we have evidence of two people being born in the same year. As such the "other-category" can be viewed as questions more specific to the original claim they are produced from, while "what", and to some degree "who" are more general. This is not a fact but is the general consensus when looking at the data.

Now looking at figure 5.28 we can see the general distribution of all the question types in the data set, not including indeterminate resulting questions. As we can observe the distribution is very close to the original distribution mentioned earlier, the only difference being between who types and other types by a few percentage points. We observe the dominance of the "what-type" questions, while the "who-type" and "other-type" questions are a distance behind as expected.

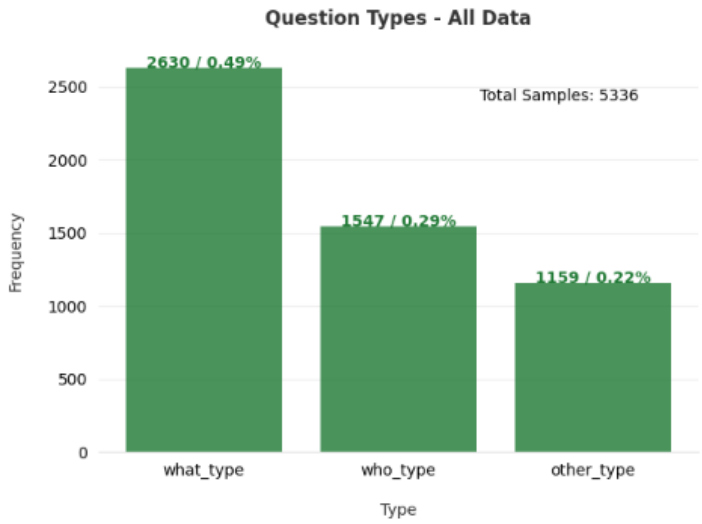

Figure 5.28: Bar chart featuring question-type distribution for Questions based of all data.

When comparing the question types for the different result combinations we can notice a difference between some of the distributions see figure 5.29. The difference between "supported/support" and "supported/contradict" is not great, but the "other-type" question category is more prevalent while the "what-type" category is less so. However, as with any other time "support/contradict" is mentioned it is hard to point to a trend due to the low number of samples within the data. On the other hand when comparing refuted/support and "refuted/contradict" we see a noticeable difference in distribution. Claims corrected from being refuted to supported seems to have a disproportionate increase in the number of "other-type" questions while the two most common categories are less so prevalent. This points to the idea that when we are correcting refuted claims more precise questions found within the "other-type" category lead to better-resulting corrections if we are to assume these questions are indeed more precise. So while the length and entities of and within a question has little impact on the resulting predictions,

the type of question generated seems to have some effect as observed here.

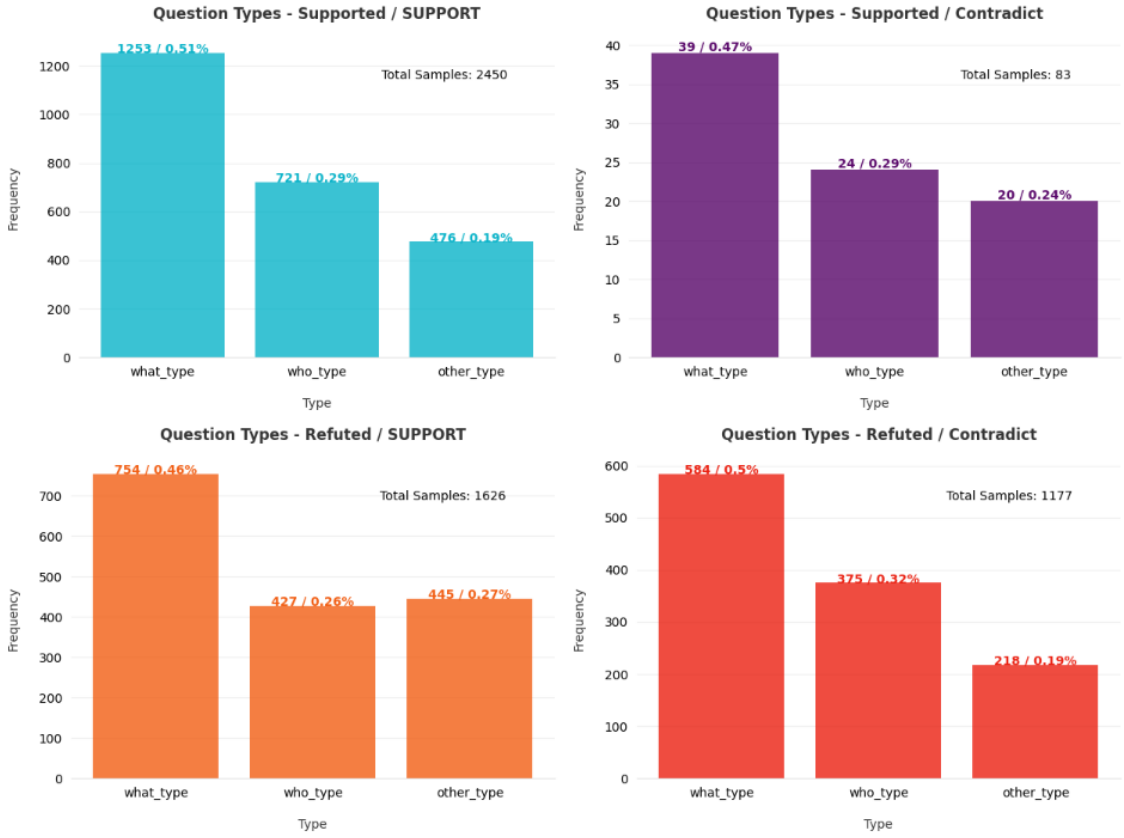

Figure 5.29: Bar chart featuring question-type distribution for Questions based of subsets of all data.

5.3 Answer Pair Analysis

In this section, we will look at how the answers to the questions impact down-the-line predictions. This section will focus on the same features as with questions in addition to an extra feature plotting similarity for answer pairs.

5.3.1 Answer Word Count Analysis

Claim Answers

When looking at the word count as it is related to answers we start with the answers using claim-based evidence. These types of answers are an average of 2.3 words long with the shortest and longest answers being 2 words and 13 words respectively. In figure 5.30 the distribution for all words within the 95th percentile of all words can be observed, and most words are single or dual answer words with a few samples having more than two.

As for comparing answers using the word counts impact on resulting predictions figure 5.31 shows the distributions. As it stands "refuted/support" contains more single-word answers compared to "refuted/contradict" and the same difference can be observed between "supported/contradict" and "supported/support". However, the differ-

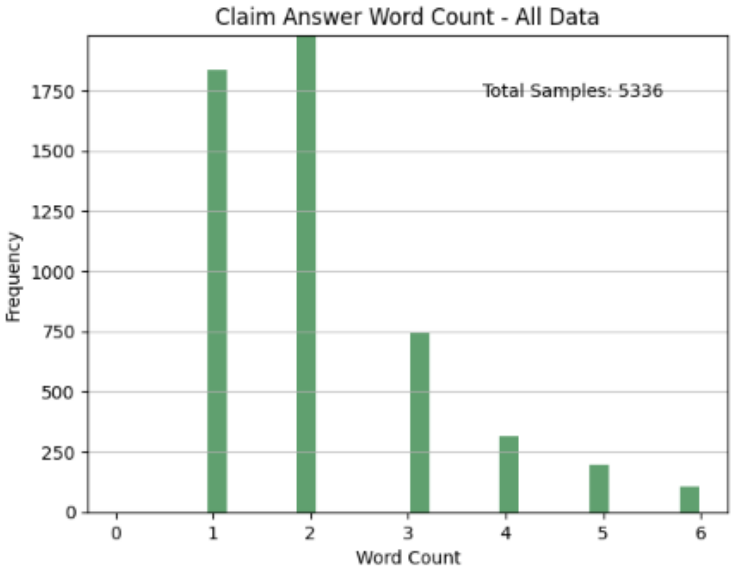

Figure 5.30: Word count distribution for Claim-based Answers for all data

ences between them are minuscule. It seems that likewise as with question length, the length of claim-based answers does not impact the resulting predictions too greatly.

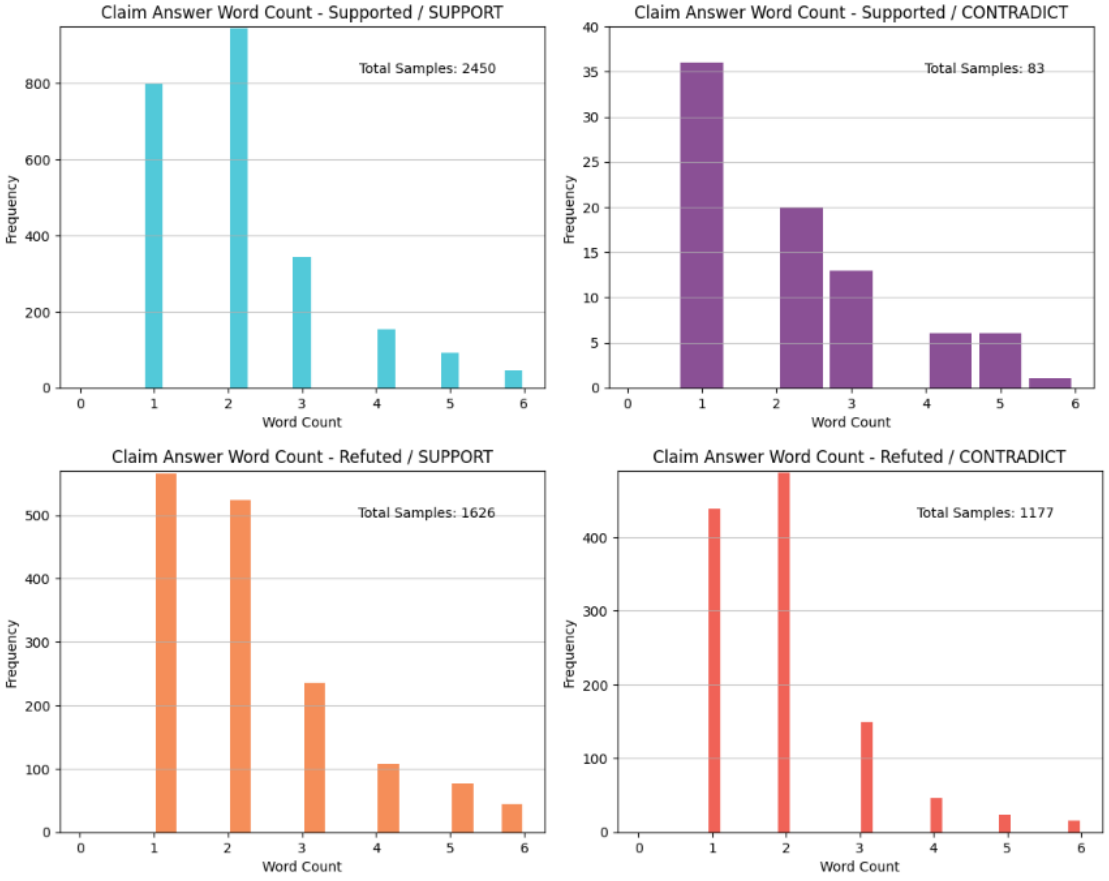

Figure 5.31: Word count distribution for Claim-based Answers for veracity sub-sections of all data

Evidence Answers

As for answers using wikidata based evidence, there is some difference compared to answers using claims. Firstly, the answers using evidence are longer on average at 3.1 words compared to claim-based answers 2.1 words, and the shortest to longest answers are 0 and 15 words respectively. In the case of zero-word answers, such answers are ignored within the pipeline. Figure 5.32 shows the distribution and one can observe that evidence-based answers definitely are more verbose when compared to claim-based answers.

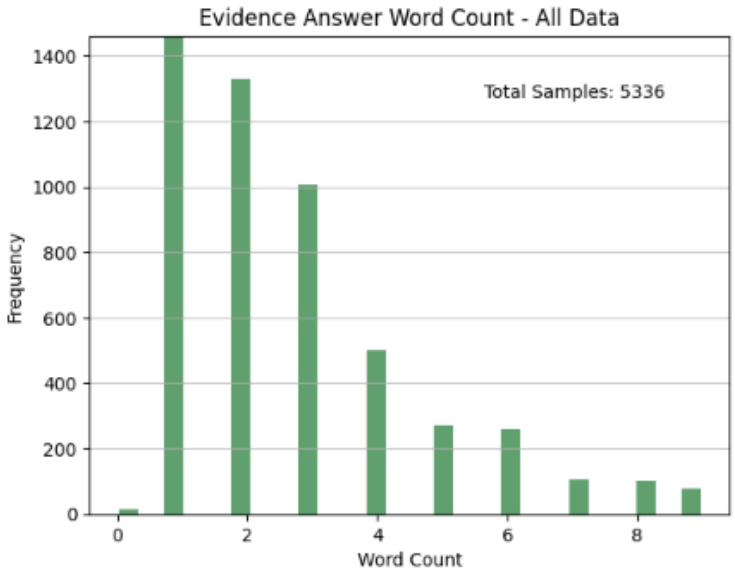

Figure 5.32: Word count distribution for Evidence-based Answers for all data

When looking at how the length of answers impacts predictions as we have done previously there is not much to observe. The distributions are again very similar between the different combinations as seen in figure 5.33, and as such there is little to comment on. There is however a greater amount of zero-length answers among the refuted claims, with most being within the "refuted/contradicted" distribution. It is however unlikely that these impart any major effects on down-the-line predictions.

5.3.2 Answer Entity Analysis

Claim Answers

Entities within claim answers are the next feature type we will take a look at. Same as with questions and evidence we will look at location, organization, person, and miscellaneous entities within the text. Looking at figure 5.34 we observe a similar layout to how the questions entities looked. There is a large number of person entities accounting for 43% of all entities within the claim answers. We also observe this to be the case when there is only one entity it is more likely to be of the type person. Which again is something we would expect if the questions also have a high degree of "person" entities.

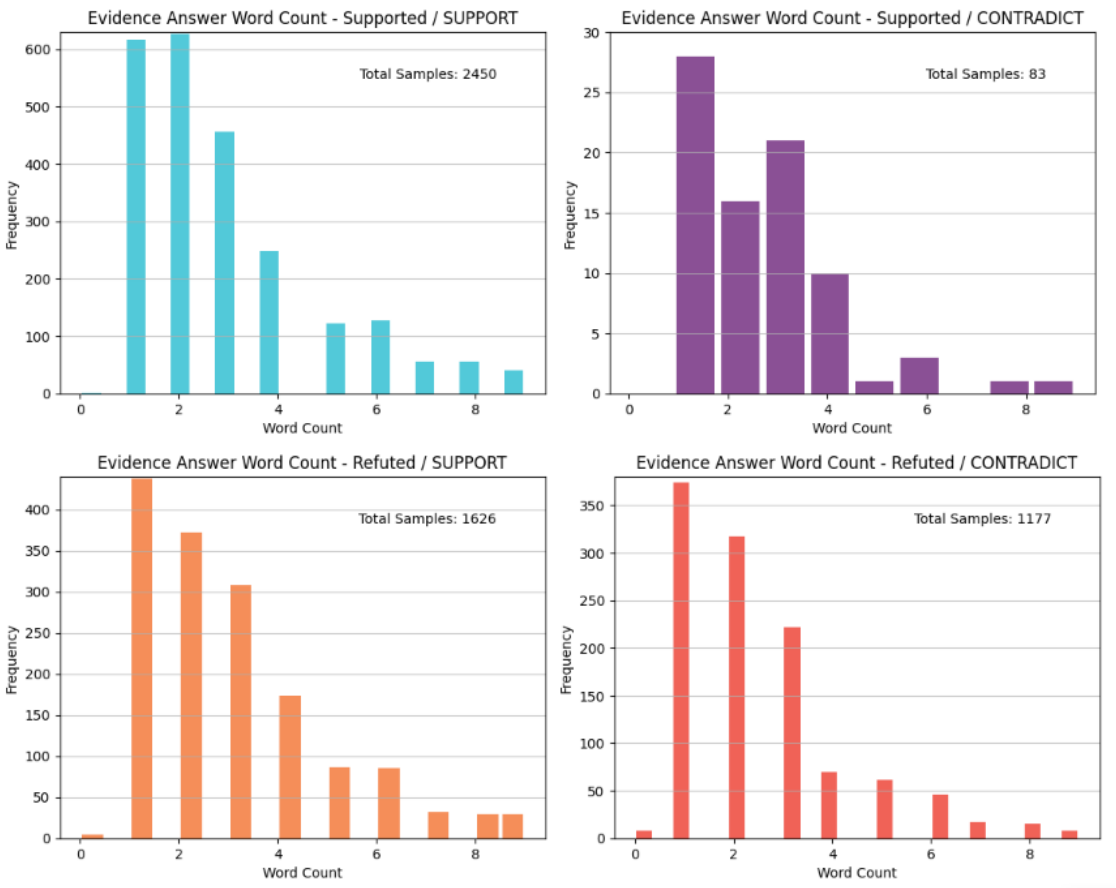

Figure 5.33: Word count distribution for Evidence-based Answers for veracity sub-sections of all data

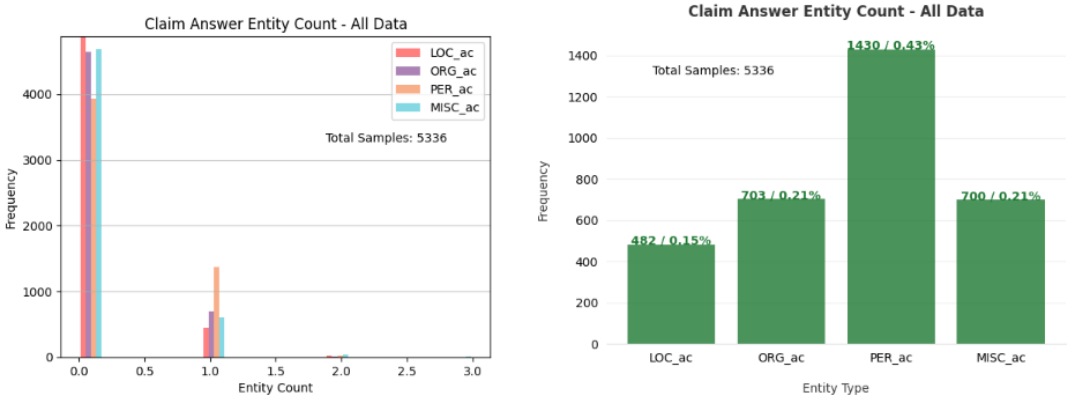

Figure 5.34: Entity count distribution for Claim-based Answers for all data

As for when we are comparing between entity counts looking at resulting predictions as seen in figure 5.35. There is a small difference in the location entity count between "refuted/supported" and "refuted/contradict" implying that if the claim-answer in an answer pair for a refuted claim refers to a location, then it is more likely that the resulting corrections will result in supported claims.

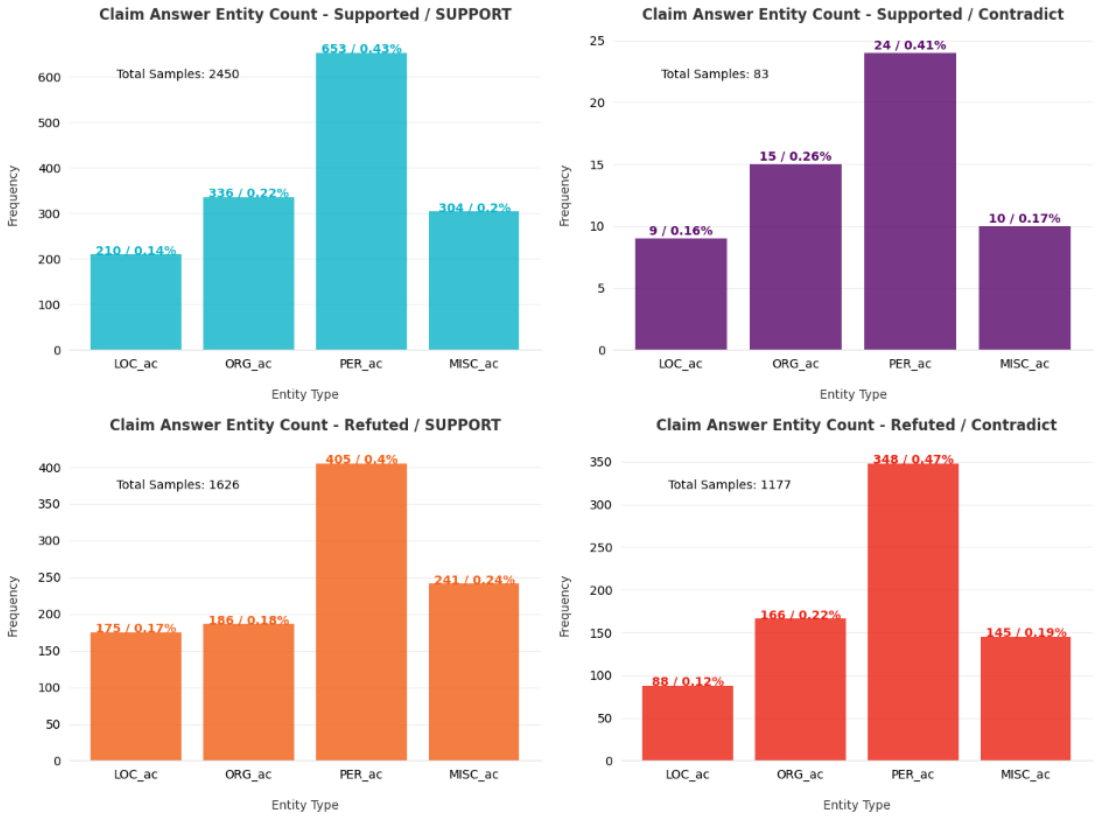

Figure 5.35: Entity count distribution for Claim-based Answers for veracity sub-sections of all data

Evidence Answers

Now looking at evidence-based answers we can notice the same distribution as claim-based evidence answers (see. figure 5.36), but this is likely due to the larger word count of evidence-based answers. These types of answers contain on average more identifiable entities as can be observed when we sum up the totals of the bar chart plotting these values when compared to claim-based answers. One interesting fact to note is that evidence-based answers follow the question entity distribution closer as we noted, and not the evidence-based entity distribution. This is positive to see as even though there is a higher density of misc entities in the evidence, the question-answering model still manages to extract the right type of entities from the evidence in most cases based on the question given.

As for comparing resulting predictions using the entity feature for evidence-based answers, there is almost no difference between the result combinations (see figure 5.37). The only result combination with some out-of-the-norm values is the "supported/contradict" combination with the misc-entity count being lower than expected. Although, as we have stated before, there are too few samples in this plot to make a statement about its significance on the corrections, and as such no broader implications for these features can be ascertained.

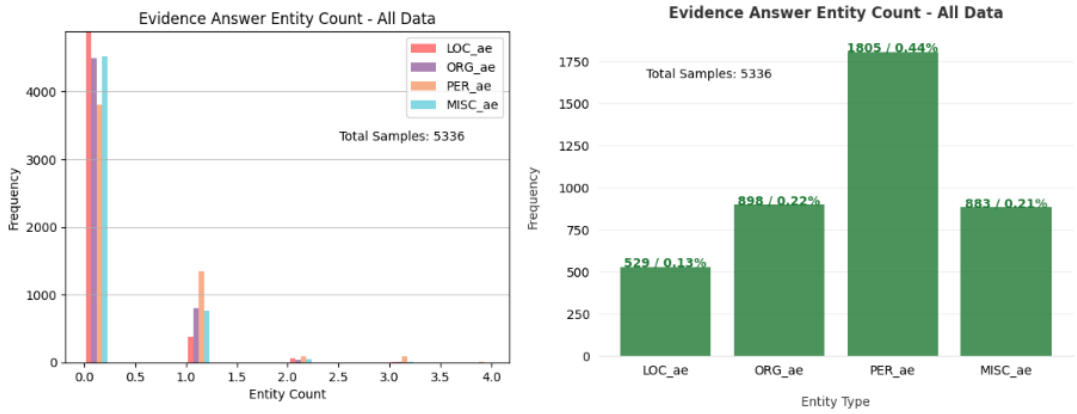

Figure 5.36: Entity count distribution for Evidence-based Answers for all data

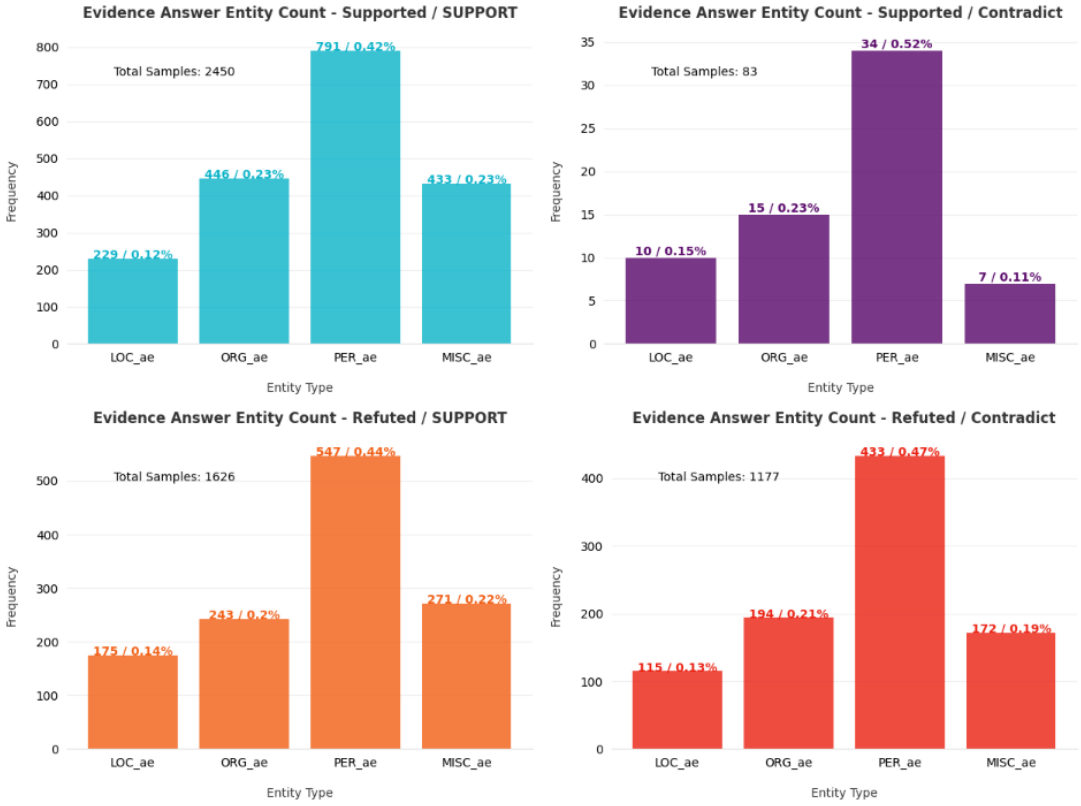

Figure 5.37: Entity count distribution for Evidence-based Answers for veracity sub-sections of all data

5.3.3 Answer Similarity Analysis

In this section the answers' similarity as given by the string matching method described in 4.1.5 of "Levenshtein Distance" will be plotted and analyzed. Figure 5.38 shows the similarities for every answer pair with most values being either 1.0 meaning identical or around 0.0 to 0.4 meaning very different. Furthermore, the fewest instances are found within the range of 0.4 to 0.9. This points to the idea that answers are either very similar or very different, which is not all that surprising. It is likely that answers

found in evidence will either be the same as the ones found using the claim, especially for supported claims or very different if another answer is found, which is what we observe in the data.

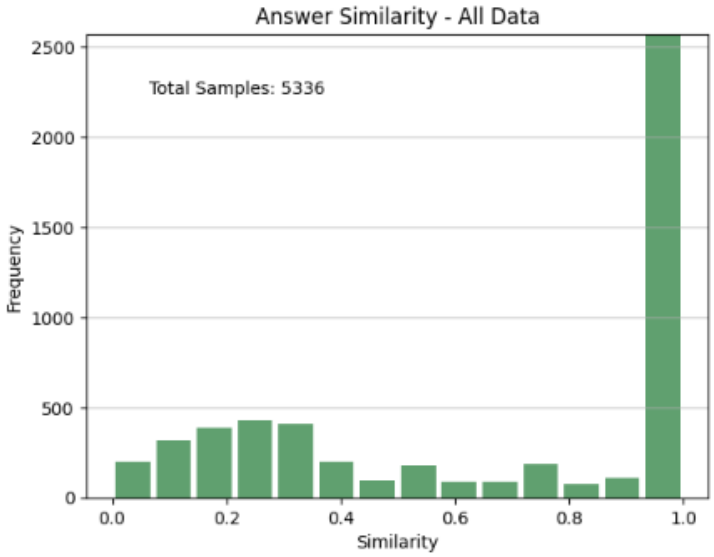

Figure 5.38: Answer pair similarity score for all data

Now when looking at the resulting prediction combinations we can observe a few things in figure 5.39. Firstly there is an immediate difference between supported claims and refuted claims as a whole. The answer pairs for refuted claims are on average less similar than those of supported claims. This makes sense as the claim and wikidata evidence will likely produce the same answer most of the time given the same question. On the other hand, the refuted claims are more likely to have differing answers between the answer pairs. This is due to the fact that for a refuted claim evidence-based answers are more likely to produce different answers than the answers using the refuted claim as evidence. Furthermore, we can observe that even within the refuted category "refuted/support" has a lower similarity score distribution compared to "refuted/contradict" which implies that when evidence and claim-based answers are different it is more likely for the resulting corrections to be predicted as support and as such be considered well formed. Finally, one point to make about the large number of 1.0 similarity claims within the "refuted/contradict" is that some claims might be only partially incorrect. Multiple questions can be generated for such a claim which target different sections of the claim. As such multiple answer pairs are produced with only the pairs relating to the question targeting the incorrect section of the original claim being of a low similarity, while the rest might produce identical answer pairs even for a refuted claim if targeting a correct section of a claim. As for the large number of 1.0s in the "refuted/-contradicted" distribution these are instances where the correct answers were probably not found within the evidence, which might be the fault of the questions posed or the question-answering model used.

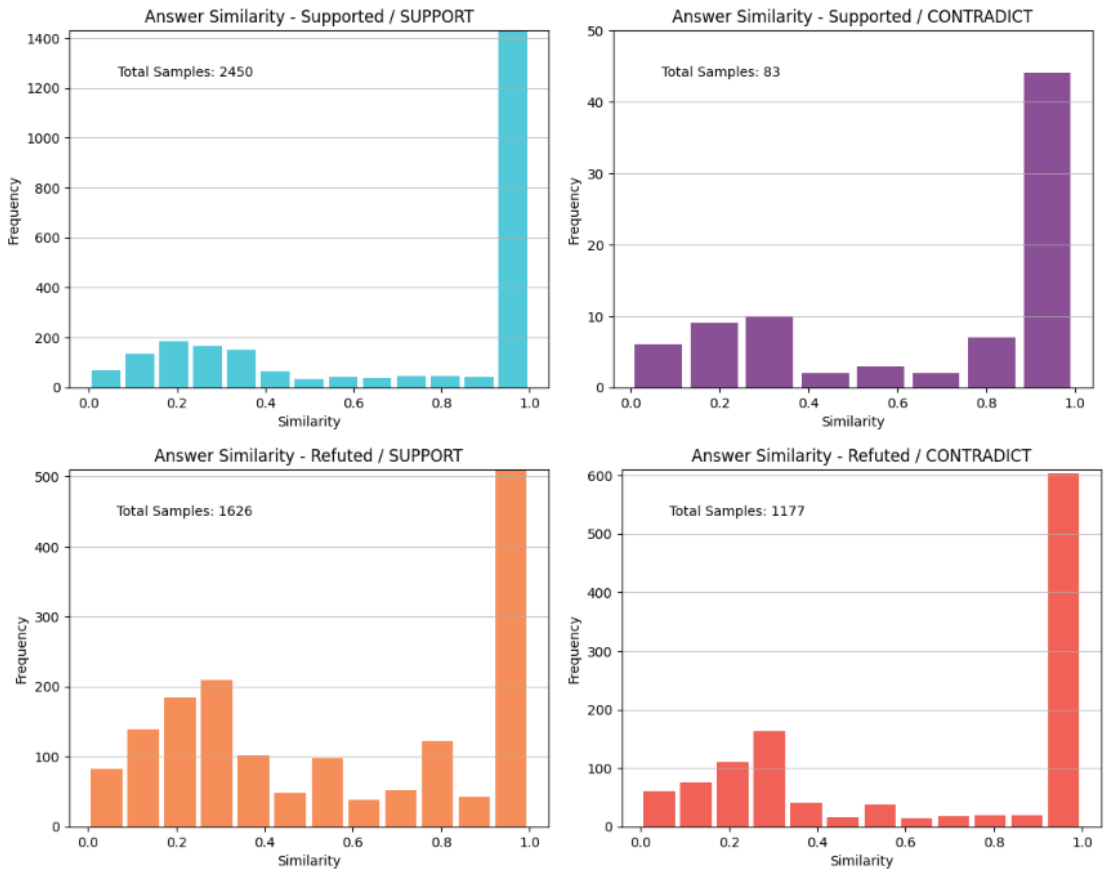

Figure 5.39: Answer pair similarity score for veracity sub-section of all data

5.4 Correction Similarity Analysis

Finally, we will shift our focus from questions and answers over to the final feature we are going to take a look at which is how similar corrected claims are to their original claims by plotting the cosine similarity and investigating any differences between the result combinations. Firstly looking at figure 5.40 we observe the distribution of similarity scores across all data. We can immediately notice that the most prevalent similarity score is 100% or 1.0. Other than this the rest of the scores for the most part decrease linearly as they progress towards 0 with an average of 79% across all claims.

Now looking at figure 5.41 comparing all the distributions based on results, there are a few things we can observe and draw conclusions from. Firstly when looking at the higher end of similarity between "refuted/support" and "refuted/contradict" claims we observe a difference in how similar claims are. The distribution of "refuted/support" is shifted a bit to the left compared to the refuted/contradict and all the other combinations. The most reasonable explanation for this begins by explaining both "supported/support" and "refutes/contradict". One would assume for the "supported/-support" results to have a high similarity score since when we correct an already supported claim, the correction would either not change anything, or as is often the case with fuzz100 which is being evaluated here, we would assume the claim is corrected to the same claim or something very very similar. As for "refuted/contradict" claims it would also make sense for there to be a high number of claims that are very similar

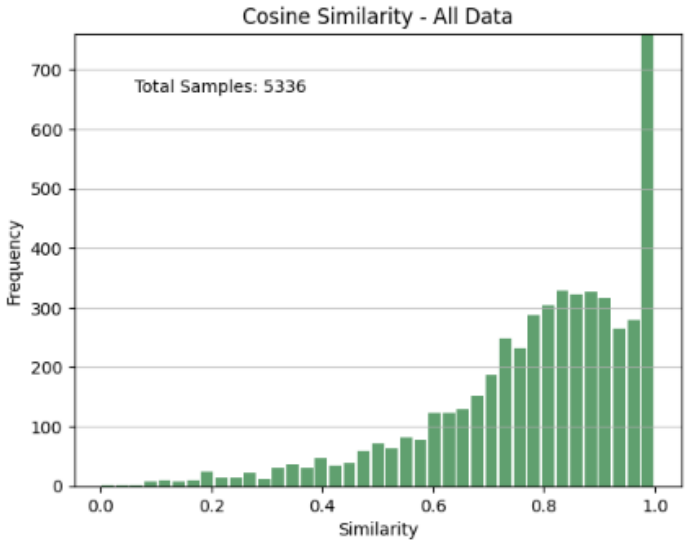

Figure 5.40: Histogram featuring cosine similarity distribution for claims and corrections based of all data.

between the original refuted and the predicted contradict claim. Both these categories distributions make sense since if a claim is originally of a given type then if they have a high similarity score the MultiVerS model is likely going to predict them as the same given category.

On the other hand "refuted/support" shows us that when correcting refuted claims we are likely to get new corrected claims that are not 100% similar, but still pretty similar. This makes sense as a corrected refuted claim should never be too similar to the original as then it would be classified as the same originally refuted claim. To make a claim correction often some small changes are necessary and thus the "refuted/support" claims are very similar to the original but not 100% as with the other combinations. The outlier in all of these distributions is the "supported/contradict" distribution. One would assure the same type of distribution here as with "refuted/support" however this is not the case. Many of the corrections are similar to the original claim while still not being classified as such. The most logical reason for this is inherent to the MultiVerS model we are using for corrections. If the claims are supported and we have corrections that are 100% similar, then these could be considered misclassifications by the MultiVerS model if they are not predicted as such.

5.5 Logistic Regression and Random Forest Results

In this section, we will present our results from our dual-purpose logistic regression and random forest classifiers for questions. The goal of these models was to first produce good results using the features discussed in the previous sections by identifying questions that lead to supported and contradicted predictions aka. well-formed/high-quality questions. Additionally, while also giving us some insight into the features by using feature selection to investigate which features are important when generating well-formed questions. As stated in 4.1.7 we are using the data set which is the same as what was also investigated in the previous sections. Firstly I will present the results

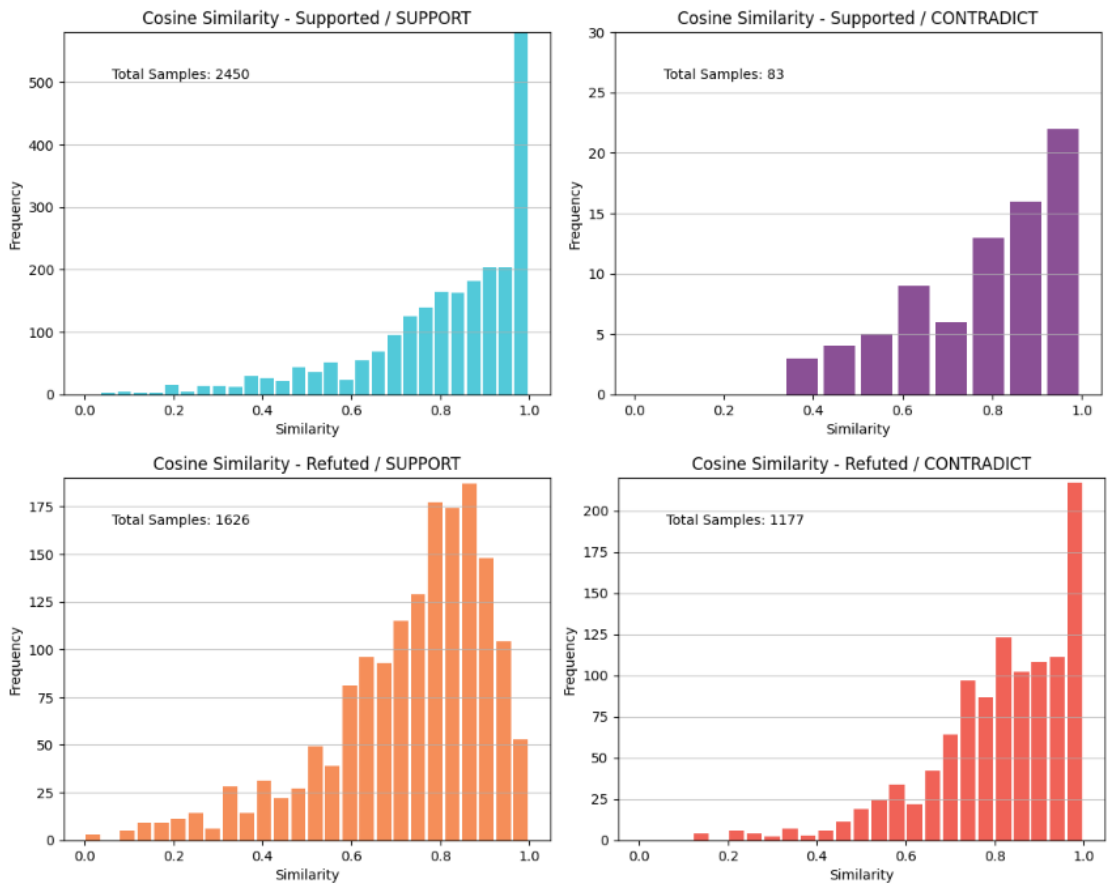

Figure 5.41: Histograms featuring cosine similarity distribution for claims and corrections based of subsets of all data.

as we obtained them, and then look at the feature selection results.

Results

Figure 5.42 shows our results using logistic regression. The first matrix shows results when not using "balanced" class_weight parameter in addition to the solver being different from the best ones discussed in section 4.1.7 when discussing the best parameter found by gridsearch. The second matrix uses the best parameters found using gridsearch with our focus being on the "balanced" class weight.

For the first matrix the results if we were to only look at accuracy is actually very good at 77%, however, this is of course not the best way of evaluating these results. We want a model that tells us when a question is good (1) but also when a question is bad (0), and this is not possible with the model producing the results related to matrix one. We should in our case look at the macro average instead. Macro average gives us 44%, and is better at telling us which model produces the best results in this situation. When inspecting matrix one we can see that this model almost exclusively only predicts the 1-class and this being the reason for its relatively good scores. Out of 247 instances of the 0-class the model only predicts 0 three times two of which were correct. This led to an abysmal recall of 0.01% for this class and an unusable model. To combat this the "class_weight" parameter was set to balanced for further testing and we got better results.

Logistic Regression Classifier Results

```
Confusion matrix Test Set
 [[  2 245]
 [  1 839]]
              precision    recall  f1-score   support

           0       0.67      0.01      0.02       247
           1       0.77      1.00      0.87       840

    accuracy                           0.77      1087
   macro avg       0.72      0.50      0.44      1087
weighted avg       0.75      0.77      0.68      1087
```

```
Confusion matrix Test Set
 [[158  89]
 [429 411]]
              precision    recall  f1-score   support

           0       0.27      0.64      0.38       247
           1       0.82      0.49      0.61       840

    accuracy                           0.52      1087
   macro avg       0.55      0.56      0.50      1087
weighted avg       0.70      0.52      0.56      1087
```

Figure 5.42: Matrices showing results for Logistic Regression Classifier

The second matrix shows the results using "class_weight" balanced, and immediately we can see an increase in the 0-class predictions and an overall jump in the macro average score from 44% to 50%, although the overall accuracy of our model became significantly worse going from 77% to 52%. While these new results are still better in our opinion compared to matrix one, they are not as good as one would hope. Thinking maybe that maybe the problem was not linearly divisible or maybe logistic regression was just not the best model for the task we then attempted another classification method of random forest.

Figure 5.43 shows the results of training a random forest classifier on the same data. Matrix one in the new figure shows the results of using no class balance and max_depth 10, 300 n_estimators and random state 42. While matrix two shows the results as described in 4.1.7. While it performs ever so slightly better the model using no class balance still has the same problem as the logistic regression model of the same type. The 0-class has almost no predictability except for a few instances.

However, matrix two is a lot more promising. The overall accuracy of the model is a lot higher compared to the best logistic regression model 69% compared to 52%, and the same goes for the macro average 61% up from 50%. Compared to the logistic regression model our forest model does however have a lower recall for our 0-class, but on the flip side, the precision is 10 percentage points higher. All in all the random forest model performs acceptable and the best out of our tested models at predicting whether a given question will result in a supported or contradicted prediction. This also informs us it the question is of high-quality is it leads to 1-class predictions and low-quality in the case of 0-class predictions.

Random Forest Classifier Results

Confusion matrix Test Set
[[10 237]
[3 837]]

	precision	recall	f1-score	support
0	0.77	0.04	0.08	247
1	0.78	1.00	0.87	840
accuracy			0.78	1087
macro avg	0.77	0.52	0.48	1087
weighted avg	0.78	0.78	0.69	1087

Confusion matrix Test Set
[[133 114]
[224 616]]

	precision	recall	f1-score	support
0	0.37	0.54	0.44	247
1	0.84	0.73	0.78	840
accuracy			0.69	1087
macro avg	0.61	0.64	0.61	1087
weighted avg	0.74	0.69	0.71	1087

Figure 5.43: Matrices showing results for Random Forest Classifier

Feature Selection

Now to get even more insight into what makes a question lead to different resulting predictions, this section will look at the features retained after feature selection and discuss how these might impact down-the-line predictions. Using the default select-from-model of sklearn, in addition to the estimator corresponding with the model that is to be trained e.i. logistic regression estimator for when logistic regression was about to be trained, and the random forest estimator for when random forest was about to be trained, we were able to identify which features had the most impact on down steam predictions.

For **"Logistic Regression"** the features with the most impact was determined to be;

- *Word count for questions, claim-based answers, and evidence-based answers.*
- *Person(PER) entity count for both claim-based and evidence-based answers.*
- *Location(LOC) entity count for evidence-based answers and evidence.*
- *Organization(ORG) entity count for evidence.*
- *Miscellaneous(MISC) entity count for evidence.*
- *Correction similarity between the original claim and the corrected claim.*

For the **"Random Forest classifier"** the features that had the most importance were;

- *Word count for questions, evidence, claim-based answers, and evidence-based answers.*
- *All entity counts of (PER), (LOC), (ORG), and (MISC) in evidence.*
- *Answer Similarity*
- *Corrected similarity.*

Looking at the features that produced the best results, many of the ones included here were not the same ones found to impact the results when we analyzed the features in the previous sections. There is however a difference, and that is that while we analyzed the general look of the features earlier, feature selection lets us investigate these features as they relate to other features. We had previously only investigated these features in a vacuum divorced from each other. A certain word count feature might not tell us much on its own, but together with a certain entity counts feature it seems like they do have some impact on the resulting predictions.

As for the features that seem to definitely have some impact on the results on their own, answer similarity and to a larger degree correction similarity seems to be in this category of important features. Correction similarity (cosine similarity) was identified as having differing distributions between resulting predictions when we discussed it earlier and seems to also be important for our classifiers here in determining the correct predictions. This would then imply that there is a strong correlation between how similar a resulting correction is to the original claim in making correct predictions. As for the second important feature, "Answer Similarity", much the same applies but not to the same degree, seeing as it was only used in the logistic regression classifier. All in all, the features we have looked into tell us something about what a good question might be, given that we understand that all the other features used are downstream from the questions we ask.

Chapter 6

Conclusions and Future Work

This chapter concludes our thesis and aims to summarize the findings. Firstly, in this thesis we investigated the domain of explainable fact checking using as far as we know a novel approach. This approach consisted of creating a pipeline method using question generation, question answering, similarity metrics, correction, and finally verification in an attempt to provide alternative corrected claims for claims that are misinformed. Systems like this would ideally contribute to lessening the workload of human fact checkers, in addition to being implemented as an automatic method of fact checking in social media spaces such as fact checking tweets on twitter.

When we started our work on the master thesis we also set out with the goal of answering a few research questions, and now that the results have been analyzed we can accomplish this.

- R1: "Can an unsupervised question-answering system be used for factual error correction of claims?" Using off-the-shelf question generation and question answering we show that a high ratio of originally refuted claims can be corrected by applying simple similarity-based methods. Furthermore, the quality question detection system using random forest can be used to automatically detect questions that contribute to the overall corrections with acceptable performance.

- R2: "Is it possible to generate high-quality factoid questions from claims?" As for this question, we addressed it in three phases. Firstly, we did an in-depth feature analysis on a set of comprehensive features extracted from questions, answers, evidence, and corrected claims in the form of similarity. Secondly, our use of feature selection to detect the most discriminative features and the development of the quality question classification model with the purpose of being able to detect good questions gave us some insight. We learned that for the individual features there wasnt really a single feature that determined a high-quality question from a low-quality question. However, the results of the random forest classifier, and the corresponding feature selection shows that word count for questions were among the discriminative features, and that for down-steam tasks such as question answering, entity types became more important. Additionally down-stream similarity metrics also provided good indicators in detecting a high-quality question. All in all, this shows that questions generated from claims can be high quality.

- R3: "Does question-answering lead to better verification results?" The results of our correction model based on different thresholds and contextual similarity confirms that QA-based correction improves the verification results without impacting the supported claims too much. Using fuzz100, our most promising method, the model is able to reduce the amount of "refuted/contradict" claims by about 50%, and correct about 65% of all the "refuted/contradict" claims if we look at refuted claims originally predicted as "contradict" and compare to claims predicted as "support" after correction. In addition, correction also reduces the total amount of indeterminate claims in both the supported and refuted sets by about 7% in total. This shows that our model using question-answering produces better verification as compared to verifying claims and not correcting anything.

- R4: "Does question-answering lead to better error correction scores compared to transformer-based distance learning?" According to (*Thorne and Vlachos*, 2020), a fully-supervised system with gold evidence generates corrections for 69% of the refuted claims. This indicates that there is a limit to the performance of distant-supervision models relying on retrieved evidence. Their best-performing distant learning system corrects 24% of the refuted claims when using retrieved evidence. In this work, we assumed the ground truth corrections do not exist. We do not use any ground truth corrected claim to train our models. The proposed pipeline relies on the gold evidence as well as the existing models trained to perform the tasks in the proposed pipeline. It is worth mentioning that the percentage of the corrected claims in this study is not directly comparable to the one reported by Thorne and Vlacos, but it plots a picture of the effectiveness of the proposed method that corrects 65% of the claims as mentioned earlier.

Limitations

Even though the results described are promising, as it stands now there are still some limitations in our approach that should be discussed.

- Firstly, as described by *Zeng et al.* (2021) using a pipeline approach can lead to errors at each step. This could be observed in our in-depth analysis of the general results where our question-answering using claim-based evidence did not capture the complete answer span leading to corrections where artifact entities were left behind.

- Second, our model was not fine-tuned for the specifics of our project and as such this could limit the results we got. It could be possible that with some fine-tuning of either the question generation, and/or question answering, that we could get better results and less errors at each step of the pipeline.

- Finally, it is important to understand the correction limitations of the system. The system is composed of language models, and has no inherent reasoning skills such as what a logic program might have. As such the corrected claims are a result of a language model finding likely corrections, but not through reasoning, but rather text extraction. This means that while the model might be confident in its corrections, these corrections might still be incorrect despite what the model

thinks. Following this, our corrections are also only as good as our evidence. Our model would not be able to correct a claim if either no evidence, or the wrong evidence is provided. This also points to another problem inherent to the domain of fact-checking in what evidence we should trust. For this project evidence in the form of wikipedia/wikidata sentences were provided which is a reasonably trustworthy source of information, but there is still a human element within the process that could lead to incorrect results. Furthermore, how we choose between conflicting pieces of evidence is a problem in general when it comes to fact-checking and not something the work in this thesis attempts to solve, however, it is something important to keep in mind.

Future work

While a lot has been done in this thesis, there is still a lot more that can be done. This section aims to expand on possible further contributions and future work that can be done.

- To continue the point made in limitation, some form of fine-tuning and/or few-shot learning could be implemented and tested to see if this produces better corrections.
- The models currently used within the pipeline are somewhat modular and using the same base concept one could attempt to do the task using different models. Additionally if one were to use an answer dependent question generation model, this would open up the possibility of not being limited to only using extractive question answering models as the answer dependent model would extract the required spans when creating questions.
- Another possibility to improve the already existing approach would be to introduce our random forest classifier as part of the pipeline itself, in a way that would allow it to weed out potentially low-quality questions.
- As the results show, stricter thresholding produces worse corrections, however, this opens up the possibility to investigate other methods of thresholding and as an extension similarity. Investigating the use of alternative ways of measuring similarity between answers such as knowledge-graphs might lead to better or comparable overall correction results.
- Over the past year AI has gotten a larger spotlight in the mainstream media and especially social media with the introduction of GPT-4 and ChatGPT. The idea of incorporating AI into one's workflow is no longer a totally alien concept to more people than ever. As such it could be interesting to investigate GTP-4s verification and correction capabilities, and compare to more baseline models such as the one proposed in this thesis. It might be that some form of prompt engineering would produce better results than the conventional methods of verification and correction.

Bibliography

Brenan, M. (2021), Americans' trust in media dips to second lowest on record. 1.1

Chan, B., T. Möller, M. Pietsch, and T. Soni (2023), roberta-base for qa. 4.1.4

Chiusano, F. (2022), Two minutes nlp quick intro to question answering. 2.1.6

Devlin, J., M.-W. Chang, K. Lee, and K. Toutanova (2019), Bert: Pre-training of deep bidirectional transformers for language understanding. 2.1.3

Giacaglia, G. (2019), How transformers work. 2.1.1

Gottfried, J., and J. Liedke (2021), Partisan divides in media trust widen, driven by a decline among republicans. 1.1

Guo, Z., M. Schlichtkrull, and A. Vlachos (2021), A survey on automated fact-checking, doi:10.48550/ARXIV.2108.11896. 1.1, 2.2.1, 2.2.2, 2.2.3, 2.2.4, 2.2.5

HuggingFace (2023), Question answering. 2.1.6

Issa, A. (2023), Transformer, gpt-3,gpt-j, t5 and bert. 2.1.3

Jobanputra, M. (2019), Unsupervised question answering for fact-checking, doi:10.48550/ARXIV.1910.07154. 2.3.3, 2.3.4

Kostadinov, S. (2019), Understanding encoder-decoder sequence to sequence model. 2.1.1

Lazarski, E., M. Al-Khassaweneh, and C. Howard (2021), Using nlp for fact checking: A survey, *Designs*, *5*(3), doi:10.3390/designs5030042. 2.2.1, 2.2.3

Lee, H., C. Park, S. Yoon, T. Bui, F. Dernoncourt, J. Kim, and K. Jung (2022), Factual error correction for abstractive summaries using entity retrieval, doi:10.48550/ARXIV.2204.08263. 2.3.2

Liu, Y., M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov (2019), Roberta: A robustly optimized bert pretraining approach. 2.1.3

Lopez, L. E., D. K. Cruz, J. C. B. Cruz, and C. K. Cheng (2020), Transformer-based end-to-end question generation. 2.1.5, 4.1.3

Marshall, C. (2019), What is named entity recognition (ner) and how can i use it? 2.1.4

Mishra, P. (2020), Understanding t5 model : Text to text transfer transformer model. 2.1.3

Myre, G. (2021), Intelligence report: Russia tried to help trump in 2020 election. 1.1

Nam, E. (2019), Understanding the levenshtein distance equation for beginners. (document), 2.1.7, 2.3

Pant, A. (2019), Introduction to logistic regression. (document), 2.1.8, 2.4, 2.5

Polle, J. B. (2023), roberta-large-ner-english: model fine-tuned from roberta-large for ner task. 4.1.7

Prabhakaran, S. (2018), Cosine similarity understanding the math and how it works (with python codes). 2.1.7

Pykes, K. (2023), Fuzzy string matching in python tutorial. 4.1.5

Raffel, C., N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu (2020), Exploring the limits of transfer learning with a unified text-to-text transformer. 2.1.3

Reimers, N., and I. Gurevych (2020), all-minilm-l6-v2. 4.1.5

Roberts, A., C. Raffel, and N. Shazeer (2020), How much knowledge can you pack into the parameters of a language model? 2.1.6

Singh, P. (2020), A simple introduction to sequence to sequence models. (document), 2.1, 2.1.1

Stencel, M. (2019), Number of fact-checking outlets surges to 188 in more than 60 countries. 1.1

Suraj, P. (2022), Question generation using transformers. 2.1.5, 4.1.3

Sutskever, I., O. Vinyals, and Q. V. Le (2014), Sequence to sequence learning with neural networks. 2.1.1, 2.1.1

Thorne, J., and A. Vlachos (2020), Evidence-based factual error correction, doi:10.48550/ARXIV.2012.15788. 2.3, 2.3.1, 2.3.3, 6

Thorne, J., A. Vlachos, C. Christodoulopoulos, and A. Mittal (2018), FEVER: a large-scale dataset for fact extraction and VERification, doi:10.18653/v1/N18-1074. 4.1.1, 4.1.2

Vaswani, A., N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin (2017), Attention is all you need. (document), 2.1.2, 2.2

Wadden, D. (2023), The multivers model. 4.1.6

Wadden, D., K. Lo, L. L. Wang, A. Cohan, I. Beltagy, and H. Hajishirzi (2022), Multivers: Improving scientific claim verification with weak supervision and full-document context. 4.1.6

Wang, A., K. Cho, and M. Lewis (2020), Asking and answering questions to evaluate the factual consistency of summaries, doi:10.48550/ARXIV.2004.04228. 2.3.2, 2.3.4, 2

Yang, J., D. Vega-Oliveros, T. Seibt, and A. Rocha (2021), Explainable fact-checking through question answering. 2.3.5, 2

Yiu, T. (2019), Understanding random forest. (document), 2.1.9, 2.6, 2.7

Zeng, X., A. S. Abumansour, and A. Zubiaga (2021), Automated fact-checking: A survey, doi:10.48550/ARXIV.2109.11427. 2.2.1, 2.2.2, 2.2.3, 2.2.5, 6