University of Stavanger

Faculty of Science and Technology

MASTER'S THESIS

Study program/ Specialization: Data Science	Spring semester, 2021 Open / ~~Restricted access~~
Writer: Bjarte Botnevik	[illegible] (Writer's signature)
Faculty supervisor: Vinay Jayarama Setty	
Thesis title: Fake News Data Generation and Augmentation	
Credits (ECTS): 30	
Key words: Fake News, BERT, Transformers,Data Augmentation, Deep Learning	Pages: 61 + enclosure: ………… Stavanger, June 15 2021 Date/year

University of Stavanger

Faculty of Science and Technology
Department of Electrical Engineering and Computer Science

Fake News Data Generation and Augmentation

Master's Thesis in Computer Science
by

Bjarte Botnevik

Internal Supervisors

Vinay Jayarama Setty

June 15, 2021

“If you are the smartest person in the room, you are in the wrong room.”

- Unknown

Abstract

Fake news is becoming an increasingly more significant problem in today's society, especially on social media. The fact-checking field in Data Science is becoming more and more popular as people want to solve this. However, for low-resource languages, there is not much to do without training data. In this thesis, we suggest a way to generate multilingual data from a knowledge base to prevent the problem of low resources. We will use pre-trained deep learning models, like BERT to measure the quality of the generated data. Lastly, we will discuss if the data generation improved the models and if it is a feasible strategy to generate more data.

Acknowledgements

I would like to thank my supervisor Vinay Setty. His excellent guidance, inputs, and knowledge have been greatly appreciated throughout the project. I would like to thank Factiverse for providing office spaces, for letting me do my master's thesis for them, but most of all for giving me access to GPUs. I would also like to thank my fellow student Eirik Sakariassen for being a great discussion partner, friend, and colleague.

Contents

Abbreviations

Acronym	**W**hat (it) **S**tands **F**or
NLP	**N**atural **L**angauge **P**rocessing
HTML	**H**yper **T**ext **M**arkup **L**anguage
HTTP	**H**yper **T**ext **T**ransfer **P**rotocol
RDF	**R**esource **D**escription **F**ramework
BERT	**B**idirectional **E**ncoder **R**epresentations from **T**ransformers
RoBERTa	**R**obustly **o**ptimized **BERT** Pretraining **a**pproach

Chapter 1

Introduction

Fake news is becoming more and more accessible, both in the media and in social media. People read fake news without even knowing it. It is becoming harder to differentiate between fake news and real news. It can also have real-world consequences. An example of this is the Pizzagate conspiracy theory [1]. Research on fake news spreading on Twitter [2] shows interesting findings. The key finding includes that the fake news tweet is 70% more likely to be re-tweeted and reaches 1500 people, six times as fast as true news. Politics may be the most vulnerable field in the fake news spreading. Politics is where we can see the most drastic consequences of fake news in the real world. Stories might be completely false, or some parts of them are fake. In politics, this is done to try to shift the vote of the reader.

Consequently, fake news detection has gotten more and more popular in the field of Data Science. In our case, there are different ways of doing this, checking if the claim is supported by evidence. Fake news detection in Nordic languages is especially interesting to us, as we read most news in Norwegian. The challenge for doing fact-checking in Norwegian is duo to the low resources available. There is not much, if any, labeled training data in Norwegian. The language models have not been as much developed and tested as the English models.

The good part about fact-checking not being as popular in Norwegian is that there is lots of room for improvement. The bad part is that it is hard to tell how well our models perform as there is nothing to compare it to. In order to show the effectiveness of the datasets we create, we will also provide a strong baseline.

In this thesis, we will present a method of generating training data for fact-checking. The data that will be presented and tested are in English and Norwegian. However, this method will be available for most languages, including the other Nordic languages.

We decided to focus on English and Norwegian, as these are the most relevant for us. We aim to try different models, trained on a combination of datasets, to find the best performing one.

1.1 Motivation

The motivation for this thesis is to generate training data for multilingual fake news detection automatically, in particular fact-checking. Especially for Nordic languages as these have, in general, fewer natural language resources available.

1.2 Problem Definition

Low resource languages, like the Nordic languages, have fewer and lower quality resources. Detecting fake news using Deep Learning needs lots of good quality data. Today, there is no data for training a language model for detecting fake news written in the Nordic languages.

Therefore, the goal is to create such data to train a model to automatically fact-check.

1.3 Usecases/Examples

The data that is being generated can be used to train fake news detection models. Because we can generate multilingual data, the models can be trained for many languages. It can be used when doing fact-checking for Factiverse AS. This way of generating data can also be used in other topics, e.g., question answering.

1.4 Challenges

1.4.1 Few Norwegian resources

We faced several challenges working with languages with fewer resources and later when generating claim-evidence pairs in this project. Most NLP tools do not work on Nordic languages, so many current English approaches are not accessible in this project.

Most of the data that is currently available for English is hand-labeled, using, for example, a crowd-sourcing platform like Amazon Mechanical Turk [3]. This is not an efficient way of creating training data in Nordic languages, mainly for two reasons. One, there are a relatively small number of Norwegian-speaking people. The amount of workers we would be able to hire is low. Two, Norwegian-speaking people do generally have a higher average wage. They would ask for more money than other nationalities, but for the same amount of work. Therefore this was not a feasible method in this project.

1.4.2 Claim generation/Language model

Another challenge was when deciding how to generate the claims and evidence. We sometimes want the model to be as precise as possible, where one word can change the outcome of the prediction. At the same time, we do not want the model to learn that, e.g., "not" means a false prediction.

1.4.3 What part of the claim to check

The decision of what part of the claim the model should check was also a challenge. As a complex claim can consist of multiple fact-checkable claims, the model needs to decide what claims should be checked. In table 1.1 we can see a claim that can be hard for the model to predict. In this claim, it is one main claim to fact-check. That is if it is 7.9 billion people on earth or not. However, there is also a secondary claim, if Donald Trump really said it or not. Even though that may not be the most interesting part to fact-check, it may be that the user wants to check that. Therefore, the model may struggle on what parts of the claim to focus on.

Donald Trump said "There are 7.9 billion people on the earth".

Table 1.1: Example of a claim having a secondary claim

1.5 Contributions

In this thesis, we suggest a way to generate multilingual data for fact-checking automatically. We will also showcase results from the models trained on this data to see if the models improved.

1.6 Outline

Chapter 2 will introduce related work on this topic to show what has been done and what we can do differently.

In chapter 3 the necessary background knowledge of this thesis will be introduced.

Chapter 4 will showcase some of the data explorations that were done. Both on the baseline data and the newly generated data.

In chapter 5 the solution approach is presented. It includes how the claim-evidence pairs were created, and it will showcase some of them and their complexity.

Chapter 6 is the experimental evaluation. It will present the results the models achieved and showcase what testing was done to achieve them.

Chapter 7 will provide some discussion around the decisions that were made during the project and how these decisions may have impacted the results. We will also discuss some alternatives to the decisions that were made.

Lastly, in chapter 8 we provide a conclusion and discuss further directions for developing the project.

Chapter 2

Related Work

This chapter will present some of the related work that is done. It includes fact-checking in English and Norwegian.

2.1 Norwegian fact-checking

Faktisk.no is a company that does Norwegian fact-checking. Their fact-checking process is manual, and they can only do a few fact-checks a week. All though, the number of fact-checks can vary quite a bit. The manual process is time-consuming, as they will need to understand the claim, research the topic and then write an article presenting the evidence to the readers. One of the main reasons we want to do this automatically is that the user can decide what news they wish to fact-check and can do so on the fly.

2.2 Norwegian resources

2.2.1 LTG Oslo

Language Technology Group (LTG) Oslo [4] is a group that works with Norwegian NLP and Computational Linguistics. This is a group from the University of Oslo, which actively works with the Norwegian language in computers. Their work includes sentiment analysis and language analysis. They have also released a BERT model with the code and data used to train this model, which we tried in this project.

2.2.2 Norwegian NLP

Norwegian NLP [5] is a collection of Norwegian NLP resources. They have collections of pre-trained BERT models, data, word embeddings, and more. This was where we first started looking when searching for Norwegian resources. Even though they have lots of Norwegian resources, there were no resources for fact-checking. The fact that there were no resources available for fact-checking was also a part of the motivation in this project.

2.3 English fact-checking

Fact-checking in English has a lot more related work. They have fact-checking websites, like factcheck.org [6] and snopes.com [7]. These websites do manual fact-checking. One or more journalists or researchers research a claim or an article to check if it is correct or not. They have different labels, e.g. ”True”, ”Unproven” and ”False”. As these fact-checks are done manually, the user can not decide what facts to check. Another downside is that it requires a lot of resources and time to do these fact-checks.

2.4 LC-QuAD

Largescale Complex Question Answering Dataset (LC-QuAD) [8] is a dataset consisting of 30 000 pairs of a question and a SPARQL query that will provide the answer for that question. The general idea is that they generate a question answering dataset from a knowledge base. In this case, the targeted knowledge base is both Wikidata and DBpedia. They automatically generate what they call ”Template Question”. This is a question that contains some natural language but also some Wikidata/DBpedia properties. They used crowd-sourcing to verbalize the questions, i.e., it is not fully automatic.

2.5 DBNQA

DBpedia Neural Question Answering (DBNQA) [9] is similar to LC-QuAD in the sense that they both generate question answering datasets from a knowledge base. The dataset consists of 894 499 instances in total, so it is a lot bigger than LC-QuAD. However, DBNQA does only provide the SPARQL queries and does not provide the verbalized queries.

2.6 Event-QA

Event-QA is a dataset for Event-based Question Answering [10]. As opposed to the two previously mentioned Data set, Event-QA is generated from a knowledge graph, EventKG. EventKG is a multilingual knowledge graph, that is modelled in RDF. Event-QA also utilizes manual work to create the queries. The users got presented by a SPARQL-query, and then had to verbalize a question.

Chapter 3

Background

This chapter discusses how the language models were built and what technology were used. How the claim is defined is also mentioned here, as this is important to why this solution is relevant.

3.1 Knowledge base

Knowledge bases can be used to store both complex structured and unstructured data. It is often a collection of data or documentation published online, but it can also be an internal service.

3.1.1 Wikidata

Wikidata is a free and open knowledge base [11]. It acts as central storage for different projects, including Wikipedia. It is readable by both humans and machines. Using Wikidata, we can automatically gather the information that is mentioned in, e.g., a Wikipedia article. Wikidata offers an open query service. Using SPARQL, we can query large data in a small amount of time. From one query, we can get around 50 000 results in 60 seconds, depending on the complexity of the claim, which we can use as true claims. Wikidata is a Resource Description Framework database. RDF databases are based on the idea to ask for resources in a triple, often noted as s-p-o (subject–predicate–object). An example of such a triple can be Donald Trump's date of birth. The triple would then be (”Donald Trump” - ”birth date” - ”June 14th, 1946”). Using these triples, we are able to generate claims from the Wikidata knowledge base.

3.2 SPARQL

SPARQL Protocol and RDF Query Language(SPARQL) is an RDF query language [12]. RDF is a semantic query language for databases. As Wikidata is a Resource Description Framework database, we can use SPARQL to retrieve data from the database. Using SPARQL, we can easily modify the query to be as complex as we want. For example, we can make simple claims if we want to and make them more complex so that the model has a variety of claim-evidence pairs to train on.

3.3 Wikitext

Wikitext contains syntax and markup keywords to format a page used by MediaWiki [13]. Wikitext also contains some HTML elements in its code. An example of Wikitext is that the equal sign, ”=”, is the syntax for a header. So, ”= Header 1 =” becomes the biggest header available in Wikitext, with the content of ”Header 1”. This is useful for us because we want to remove this syntax to clean up the dataset. These keywords and this syntax might sometimes be interesting to see what the topic for a section is. In our case, this was not interesting, so we decided to remove them.

3.4 Paraphrasing

Paraphrasing is a way of changing the wording of text by using various methods of modifying the text while still keeping its meaning. Examples include switching words with synonyms, adding adjectives, and adding and deleting random words. The method of paraphrasing is a well-known way of augmenting the data already available and thus creating more training data. We wanted to try this to see if it improved the models, and if so, how much it improved.

3.5 Hugging Face

Hugging Face provides open-source NLP technologies [14]. Through transformers, they host pre-trained models for anyone to use. By fine-tuning these models to fit our use-case, we can take advantage of these models. They also provide datasets for NLP and their own tokenizer, amongst other things. Their tagline on GitHub is ”Solving NLP, one commit at a time”. Companies like Facebook AI, Microsoft, and Google AI use Hugging

Face and make some of their models public. Thus, Hugging Face showcases already pre-trained language models that we can re-use use for our task.

3.6 Transformers

Transformers provides general-purpose architectures for Natural Language Understanding [15]. The general-purpose architectures in our case are BERT-multilingual and RoBERTa. A great feature of Transformers is that it is accessible to entry, meaning it is easy to fine-tune a pre-trained model, and thus we can quickly test our data. In our case, this is helpful as we can test the quality of our generated data by measuring how a model trained on it performs. We are then able to test lots of different models.

3.7 BERT

Bidirectional Encoder Representations from Transformers (BERT) [16]. BERT is a language representation model made by Google. The goal for BERT is to understand natural language, meaning human language, to better the performance of various NLP tasks. In our case, BERT is pre-trained on data from 104 languages. What we need to do is to fine-tune the model to fit our task. The fine-tuning was done on the generated data. Using BERT, we can easily re-use general language models and fine-tune them to a specific task. Training more models with different datasets to see which combination performs best is effective using BERT. In figure 3.1 below we can see an example on how BERT tokenizes a sentence.

3.8 RoBERTa

Robustly Optimized BERT Pretraining Approach (RoBERTa) builds on BERT's language masking strategy [18]. Some of the tasks that BERT can solve, RoBERTa disregards in order to make it perform better on the specific task at hand. RoBERTa was trained on a larger amount than BERT. RoBERTa is, in general, a proposed way of training BERT models to make better-performing models.

3.9 Simple Transformers

Simple Transformers is a library based on Transformers [19]. Simple transformers support text classification, amongst other tasks. It makes it easy to train and evaluate

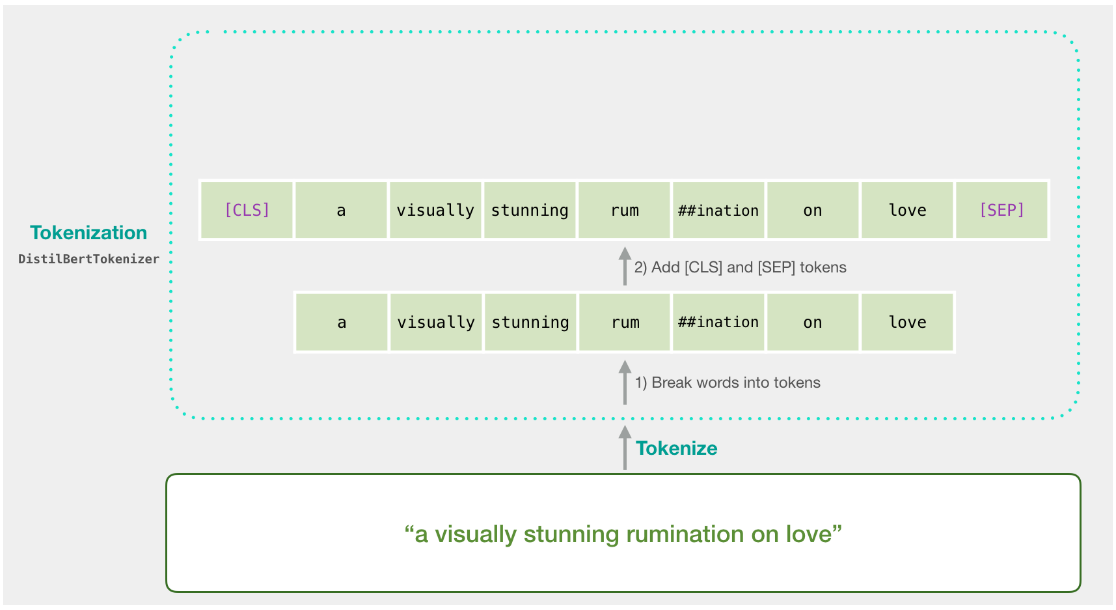

Figure 3.1: An example of the BERT-Tokenizer [17].

Transformers models. This fits our project well, as we want to test models trained on different combinations of the datasets we have generated. We can easily and quickly see how the data impacts the model. Simple transformers also support parameter tuning.

3.10 Sentence Transformers

The main difference between BERT and RoBERTa, and sentence transformers is that sentence transformers do sentence-level embeddings, as compared to word-level embeddings in BERT and RoBERTa [20]. As the overall meaning of the sentences in the claim and evidence are sometimes more important than the meaning on a word-level, we wanted to try this model using sentence-level embeddings.

3.11 Stance Detection

The task of fact-checking can be boiled down to stance detection. In our case, we want to check if the evidence supports a claim or if it is disputing it. Stance detection is known as one of the core parts of fake news detection and fact-checking. Figure 3.2 showcases how stance detection work in our case. The input is a claim and a piece of evidence. The models will then predict whether it is true or not. True meaning that the evidence supports the claim, and false if the evidence disputes the claim.

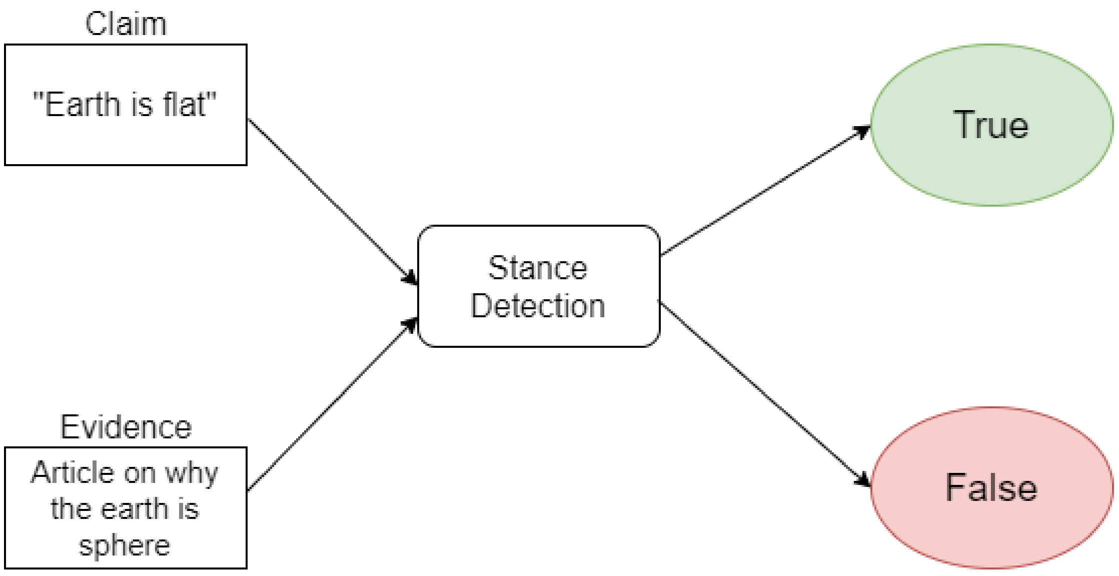

Figure 3.2: Demonstrates how stance detection can be used for fact-checking.

Chapter 4

Data Exploration

This chapter will explore the data that was used to train the baseline model. It will also explore the new data that was generated during this project. It is interesting to compare the two datasets, as it makes it easier to see why new data is valuable. We will also look at features of the claim and evidence pairs, specifically their length in words, to see if there are any interesting statistics.

4.1 Baseline Data

The baseline data is data that we already have gathered in the earlier stages of Factiverse. This is data that has been used to train language models in earlier stages of the project. We use this data to train a model that will be the baseline model.

4.1.1 English

Fever Fever is a dataset for Fact Extraction and VERification[21]. It contains claim-evidence pair that is classified as ”supported”, ”refuted” or ”NotEnoughInfo”. Only the first two labels were used in this project.

SNLI SNLI data is the Stanford Natural Languag Inference (SNLI) Corpus [22]. This collection consists of 570 000 human-written claim-evidence pairs in English. These are all manually labeled. The labels are entailment, contradiction, and neutral. We converted them, so that true is ”entailment”, ”contradiction” is false, and ”neutral” is not sure. We did not end up using the ”neutral” label, but as we are able to generate data with three labels, this might be interesting to keep in the data.

MNLI Multi-Genre Natural Language Inference (MultiNLI) corpus is modeled on the SNLI Corpus [23]. The difference is that MNLI has more genres of spoken and written text. The labels are the same, and we convert them to our labels the same way.

WCEP WCEP is a dataset that consists of human-written summaries [24]. These summaries are about news events that are obtained from Wikipedia Current Events Portal (WCEP). The ”title” field is kept as the claim, and the ”wcep_text” is kept as the evidence. As all these news events happened, these are all labeled as true, meaning the evidence supports the claim.

4.1.2 Norwegian

Norwegian data consists of 50 000 claim-evidence pairs from Fever and 48 668 claim-evidence pairs, translated to make training data in Norwegian. Two methods were used to translate data. The first one was using DocTranslator [25]. It can translate up to around 5000 rows, depending on the length of the sentences. However, it did not keep the file formatting, so some data processing before and after the translation was necessary, e.g., splitting the claim and evidence into separate columns and storing them in separate files. It also occurred to be really buggy, and therefore time-consuming. One column of about 5000 lines could take up to 15 minutes, and more often than not, it crashed and could not translate.

In a previous project, document translation from Google showed to be the best method [26]. It could do up to 4000 rows per translation. The big difference is that it only took about 15 seconds per translation, and it kept the file format so that less processing was required. All the Norwegian training data was translated using this method. Early in this project, we decided to translate even more data to have more Norwegian training data. The total amount of Norwegian translated training data equaled 148 933 claim-evidence pairs when this was done.

Table 4.2 presents how some of the claim-evidence pairs are. As seen, the evidence is straight to the point. The evidence only mentions topics mentioned by the claim and nothing else. We will later in this chapter present why this might be interesting to compare with the generated data. When the evidence is concise like this, there is no way that the model might predict the wrong label by, e.g., focusing on another and irrelevant part of the evidence.

	Claim	Evidence
English training data	11.7	18.0
Norwegian training data	13.2	27.1

Table 4.1: Showing average length in words for claims and evidences in baseline data.

Claim	Evidence	Label
Vrenna and I both fought him and he nearly took us.	Neither Vrenna nor myself have ever fought him.	0
But that takes too much planning	It doesn't take much planning.	0
Well you see that on television also.	You can see that on television, as well.	1
How do we fix this?	Can we fix this?	1

Table 4.2: Examples of claim-evidence pair from basline data.

4.2 Generated Data

The generated data is all the data we have produced in this project. It includes simple claims and more complex claims, which are both explained more later in Section 5.2. In this project, we mainly focused on generating English and Norwegian data, but our proposed method could be used for most languages.

	Claim	Evidence
English training data	10.1	72.4
Norwegian training data	10.0	86.2

Table 4.3: Showing average length in words for claims and evidences in generated data.

4.2.1 English

The English data consists of 742 952 claim-evidence pairs. When we do paraphrasing, we have 1 965 267 claim-evidence pairs. This is made from mainly four combinations of properties. This showcases how much potential to create data when we managed to generate so many pairs with only four combinations. For every true claim, we generate five false claims. We chose five because even though there are only minor changes from the true and false claims, we wanted the models to be sensitive to those changes, as one word can change the correct label for a pair. Therefore, we thought five false claims for every true claim was a fitting amount.

If we compare table 4.1 and table 4.3 we can see how big of a difference there is in the datasets. The evidence in the generated data is almost four times as long as in the baseline data. The longer the evidence, the harder it becomes for the model to predict the correct label. Often the claim is answered in one or two sentences, and the rest of the evidence may not be relevant to the claim. Table 4.4 showcases how long the evidence can be. The claim of the evidence is "Alan Turing was the originator of the Turing Machine." As we can see, the sentence that contains the part is needed to support the claim consisting of ten words. In other words, the rest of the 349 words in the evidence is not needed to decide the correct label. If we compare this with table 4.2, we can see that these examples are very different. The evidence table 4.2 is the only text that is relevant to the claim.

Evidence	Total words
...The Turing machine was invented in 1936 by Alan Turing. ...	359

Table 4.4: Example of a hard claim to check form generated data.

There are also examples where the generated data is similar to baseline data. Table 4.5 showcases how the data can vary, and in this case, the generated data is similar to the baseline data. Here the length of the evidence is almost the same as mentioned in table 4.1.

Evidence	Total words
The principle of least constraint ... enunciated by Carl Friedrich Gauss ...	27

Table 4.5: Example of a easier claim to check, more like baseline data.

4.2.2 Norwegian

Data generated for Norwegian consists of 98 787 claim-evidence pairs. We could not perform paraphrasing on Norwegian data, as the package we used did not support it. The Norwegian data is made from the same combinations of properties as the English dataset. In the same way, as in the English dataset, we generate five false claims for every claim.

There is so much more English data than Norwegian data because the English language is more used. The main reason is that there are fewer Wikipedia articles available in Norwegian. Since English is a more widespread language, it will be more Wikipedia articles created and maintained.

In table 4.6 we can again see how different the generated data is from the baseline data. The claim, in this case, was translated to English and is "HTML was originally created in 1989 by Tim Berners-Lee". We can see that in nine words, we have sufficient enough evidence to say that the evidence supports the claim. The other 241 words are strictly not needed in this case. These are all examples of how the difference in datasets can be challenging.

Evidence	Total words
...HTML ble opprinnelig definert i 1989 av Tim Berners-Lee. ...	250

Table 4.6: Example of a tough claim to check in Norwegian.

4.3 Benchmark Data

The dataset we use as a benchmark is a dataset created in a former project. The benchmark consists of fact-checks that were scraped from websites and companies like Politifact.com and AFP Fact Check. Claims and evidence are scraped from their website and generated into claim-evidence pairs to use as a benchmark set. The benchmark data also includes some handmade claim-evidence pairs. These are generated by manually writing claims and then choosing snippets from google as evidence. The google snippets were chosen as these would most likely be the closest fit to the fact-checking data in production. As shown in table 4.7 the length of the evidence in the benchmark is somewhere in the middle of the baseline data and the generated data. This indicates that the benchmark may be a good test set to measure the performance of both datasets.

	Claim	**Evidence**
English benchmark data	12.2	44.1

Table 4.7: Showing average length in words for claims and evidences in benchmark data.

Chapter 5

Solution Approach

The overall idea of our method is that a fact is to be generated from a knowledge base. Then a Wikipedia page is used to gather enough evidence to support the claim. So the true claim needs to be generated, such that a single Wikipedia page can support it. When this is done, it is easy to generate a false claim by augmenting the true claim by, e.g., changing a name or a date. Thus the evidence will no longer support the claim.

5.1 Overview of Claim Generation

Our goal for generating claims was to generate lots of data for multiple languages. In this thesis, we will focus on English and Norwegian. The claims need to be written so that it is easy to extract evidence from a Wikipedia page. One of our goals was to generate both true and false claim-evidence pairs efficiently.

5.1.1 Complex Claims

Complex claims need some definitions and guidelines on how they should be fact-checked. Our definition of a complex claim is if it contains two or more facts to check in the claim. In these complex claims, the model might have a hard time predicting the correct outcome. We have mainly two different types of complex claims.

Fact-check Worthy

In table 5.1 we can see a complex claim where all facts are fact-check worthy. This claim consists of mainly three sub-claims. 1. Darren Turret was a journalist, 2. Darren Turret was from the United Kingdom, and 3. Darren Turret was born 1965-06-02. All these claims are fact-check worthy, and they all need to be true for the model to predict true. If one of these sub-claims is false, the model should predict the claim to be incorrect. Thus, if the claim says he is born 1965-03-02, it should predict false, even when the other two sub-claims are true. In this example, every sub-claim is worth checking, as they are essential to the claim. As we want the model to predict true if every sub-claim is true, we decided to generate true claim-evidence pairs this way.

Darren Tulett was a journalist from the United Kingdom. Darren Tulett was born 1965-06-02.

Table 5.1: Example of a complex claim.

Not Fact-check Worthy

Table 5.2 shows a complex claim where it contains a sub-claim that is not fact-check-worthy. The first claim is that there are 7.9 billion people on the earth. The second claim is if Donald Trump really said that it is 7.9 billion people on the earth. It is mainly the first claim that is interesting to us. However, it may also be more interesting for some to check if Donald Trump actually said it or not. In these cases, it may be categorized as subjective of the claim is a complex claim or not. Therefore we decided to discard these claims as complex, to not make it too difficult of ourselves.

Donald Trump said ”There are 7.9 billion people on the earth”.

Table 5.2: Example of a complex, but not fact-checkw worhty claim.

5.2 Generating Claims

Before we started generating any claims, we had to define what claims would be the easiest to generate while still having high quality. The number of claims that could be generated from that fact was also a significant factor. We want to have Wikidata properties that we can create lots of claims from, and in table 5.3 we can see the top ten properties. The interesting properties are those that have values that are easy and interesting to fact-check. Some analyzing of the properties needed to be done, to see if they were suitable for this task. The more careful we were when picking the properties for generating the claims, the less work had to be done when generating the evidence.

Times occuring	**Property names**
71393700	author name string
44571399	instance of
36672332	cites work
17111198	publication date
17084898	title
16855516	published in
16136807	PubMed ID
15998850	page(s)
15946901	volume
14356774	issue

Table 5.3: Ten topmost occurring properties.

5.2.1 Non-promising Properties

The description for ”author name string” from Wikidata is: ”string to store unspecified author or editor name for publications; use if Wikidata item for author (P50) or editor (P98) does not exist or is not known”. So this property is useless in our case. Since it is only used when a Wikipedia page does not exist, it means that it is possible to gather evidence for that claim, and therefore useless for us. The next property is ”instance of.” This is a property that every object has. All humans have the property ”instance of: human.” Every country has the property and value ”instance of: country.” The property is a good way to filter out the unwanted entities but is not helpful for us when generating the claim.

5.2.2 Promising Properties

Promising properties are those that can quickly generate a large number of claim-evidence pairs straight from the query. The key point is that there are many results from the query and that the properties are mentioned in the Wikipedia abstract. The property ”date of birth” is a good example of a promising property. The birth date is almost always mentioned in the abstract of a person’s Wikipedia abstract. Choosing these promising properties, there is no need for doing any post-processing. Another promising property we found was ”occupation”. If we choose specific occupations, we can generate lots of claim-evidence pairs. Especially the occupation ”actor” is an example of why it is a promising property. Actors are often famous, or at least well-known, people. Thus their occupation is almost always mentioned in the Wikipedia abstract. Other occupations, like football players, were also a great property to use to generate data.

5.2.3 Simple Claims

We wanted to try with different complexity of claims to see how it impacted the model. The claims could also be of varying complexity when it was running in production. We started with simple claims, such as nationality and occupation. We define these simple claims as claims that have one or two facts in them. In table 5.4 we can see examples of these simple claims. There is no confusion on what the model should be doing the stance detection for. There are also fewer words in the simple claims compared to the more complex claims. This also reduces the complexity in data for the model.

Erna Solberg is the Prime Minister of Norway.
The capital of Norway is Oslo.
Christopher Hitchens worked a journalist.

Table 5.4: Examples on simple claims.

5.2.4 Complex Claims

In table 5.5 we see examples of what we define as complex claims. Generating complex claims can mainly be done in two ways. Either we can combine two or more properties, or word the sentence so that its complexity increases. In the first two examples shown in the table, we can see how combining two properties can increase the complexity of a claim. We combine both "date of birth" and "occupation" in one sentence, and they both need to be true for the model to label them as true correctly. If one of them is false, the model should classify them as false. In the last example in the table, we can see how the wording of the claim can increase the complexity. Here we used the property "discovered or invented by". Therefore, the model should predict true if either object was discovered by, or invented by, the subject. Even though only one property is used, we can word it to be classified as a complex claim. The subject in the last example also has a title, "3rd Earl Stanhope", which makes it even more complex.

Henry Fielding was born 1707-04-22 and worked as a journalist
Albert Camus was a journalist from France and was born 1913-11-07
Stanhope lens was discovered or invented by Charles Stanhope, 3rd Earl Stanhope.

Table 5.5: Examples on complex claims.

5.2.5 Generating Claims with Three Labels

An additional upside to the way we are creating claims is that we can generate claims with three labels. Some of the mentioned related work have a third label often mentioned as ”Not Enough Information.” An example of this is if we change a person’s name in the claim and keep the original evidence. If we change the name in the first example in table 5.5 to ”Donald Trump” and keep the original evidence, the label would now be ”Not Enough Information”. As the evidence is not mentioning Donald Trump, there is no way of knowing if he is, or is not, what the claim is saying. Thus, we could use this as a third label. The third label was never used in this project, but this section was written to showcase how the project could be expanded to generate data with more than two labels.

5.3 Extracting Evidence

The evidence comes from the Wikipedia abstracts relevant to the claim, and it is often the subject. We carefully chose the properties with a high chance of the claim being supported or disputed in the abstract. The more careful we were when choosing these properties, the less post-processing we had to do. An example of this is the properties mentioned in section 5.2.2. We found that a person’s occupation was often mentioned in its abstract. That is also the main reason why we chose to have ”occupation” as one of our main properties. The evidence is then gathered from Wikipedia. We do this through an HTTP request, where we get the abstract of said Wikipedia page as a result. In some of the properties, we checked if the evidence contained the fact that was going the be checked. Birth dates were one of those that were easy to check if it were in the evidence or not. For other properties, it was harder, as the property was sometimes mentioned in the evidence but not related to the subject.

5.4 Paraphrasing

We wanted to try paraphrasing, as this is a useful way to increase the amount of training data we have. We used eda_nlp [27] to paraphrase our data, and this method was only available for English data in our case. The goal for the paraphrasing was to increase the amount of data while still keeping it high quality. In table 5.6 we can see an example of how paraphrasing ruins the meaning of a sentence. ”Television” turns into ”idiot box”. When this bad example was created, the paraphrasing script was run with the parameter for ”replace word with synonyms” being too high. When we found it more fitting, the paraphrasing was not as bad as showcased here.

Even though performing paraphrasing caused some noise in the dataset, that might not be a bad thing. As we generate our data with few property combinations, it might be good to add some noise to the data. For every false claim we generate, we use the same evidence as in the original true claim. Thus, variance in the evidence could improve the performance of the models

Original	Dennis Farina was an American actor, television presenter and former police officer.
Paraphrased	Dennis Farina was an american language actor, idiot box presenter and former police force officer.

Table 5.6: Examples on how the paraphrasing may impact the quality of data.

5.5 Multilingual

One of the most promising findings in this project is how easy it is to create multilingual data. When we have found the properties for generating claims, it is easy to generate for other languages. Only three to five lines of code needed to be changed to generate in any language wanted, given there are Wikipedia articles in that language. There are mainly three things that need to change for the script to generate data for other languages. Firstly, we need to change the language of the result from the SPARQL-query. Then, we need to change the language in which we generate a claim from the properties in the SPARQL-result. This is done manually. Lastly, we change the end-point in the HTTP request to Wikipedia so that we get the abstract in the desired language.

As mentioned in earlier sections, the more used a language is, the more data we will generate. This is because there are more Wikipedia articles created in English, as opposed to Norwegian. As we get our evidence from the Wikipedia article abstract, this is the limiting factor in our case.

Chapter 6

Experimental Evaluation

In this chapter, we will present the results we achieved during this project and the data used to achieve these results. We decided to split the results into English and Norwegian because they are trained on different data and different models. English is trained on a pre-trained RoBERTa model, while the Norwegian model is trained on a BERT-multilingual model.

6.1 Experimental Setup and Data Set

All experiments were run on a Tesla V100-PCIE-32GB GPU, which made training fast. Thus we were able to train and test as many models as we did. Max sequence length was set to 100, and the learning rate was set to ”0.0002883”. For training, the parameter was simple transformer’s default values. These were the parameters we landed on after the parameter tuning.

6.1.1 English

As we wanted to see how our generated data would impact the models, we tried mainly on four combinations of datasets. One model was trained on the baseline dataset, which consists of around 500 000 claim-evidence pairs. As this was our baseline, we hope that this will be our lowest scoring model. Next, we have a trained model on only generated data, which consists of 750 000 claim-evidence pairs. We have a trained model on the baseline data and generated data combined, which makes about 1 250 000 claim-evidence pairs. Lastly, we have a model that is trained on the generated data that has been paraphrased. This makes about 1 900 000 claim-evidence pairs. As mentioned earlier, we have in total four models that we have fine-tuned to our task.

6.1.2 Norwegian

Experiments on Norwegian data are also done with multiple variations of datasets. Baseline data is 150 000 pairs of training instances. Then we trained a model with the data generated for Norwegian. This sums up to be around 100 000 claim-evidence pairs. Next, we trained a model with the baseline data and generated data combined. This consists of around 250 000 claim-evidence pairs. We found that a model trained with sentence-transformers showed promising results compared to other multilingual models in earlier projects. Therefore, we wanted to try it in this project as well, and this was the last model trained on Norwegian data.

6.1.3 Test Data

For each language, we have two datasets that we used to measure the performance of the models. The first is the test data that we generated during this project. The English test set contains 86 000 claim-evidence pairs that the model has never seen before. The Norwegian test set has 19 000 pairs. It includes some examples of some of the properties mentioned in earlier chapters, but also new properties. Most of the evidences in the training data are generated from the Wikipedia abstracts of humans. The Wikipedia abstract of humans is often written in a specific way, and they have all similarities. One example of this is that the person's birth date is almost always mentioned in the first sentence. The unique property for the test set is ”discovered or invented by”. In this case, the evidence is extracted from the Wikipedia abstract of what is being ”discovered or invented”. The reasoning behind this was to have some variety in the test set and have some different instances than what the model had trained on. The other dataset used for testing was the benchmark dataset.

6.1.4 Benchmark Data

Benchmark data consists of around 12 000 claim-evidence pairs, in both English and Norwegian. This is a dataset that we have used as a baseline at Factiverse for other tasks. It is interesting to see how models perform on other data than what we generated. This might be an indicator of how our model would perform on other stance-detection problems. The Norwegian dataset was created by translating the English dataset using Google Translate.

6.2 Hyperparameter Tuning

Simple transformers have a great way of performing hyperparameter tuning. A range of learning rate, number of epochs, and max sequence length, amongst other things, can be declared, and then simple transformers will randomly choose combinations of these and train models with these. The models will be evaluated, and their performance measures can be shown in a dashboard on a website, and we can easily compare their performance.

Figure 6.1 and figure 6.2 shows how the performance measures are displayed in the dashboard. If we look at the x-axis of both figures, about every 30 "Step" is equal to one epoch. Each line in the plots is a model train with random parameters chosen from the pre-defined ranges. Figure 6.1 shows that the models start to learn quickly but that the training loss also flattens quickly. The main reason that it flattens may be that it rapidly reaches a training loss that is almost impossible to beat. However, this might be a good indication that there is no need for lots and lots of epochs, as a large number of epochs does not improve the performance.

In figure 6.2 the evaluation loss is displayed. The evaluation loss is calculated on the generated test set after every epoch. The figure shows that the evaluation does not decrease but rather a slight increase. This is a typical sign that the models are overfitting the training data. Thus, we trained a model that trained on fewer epochs to compare the results and see if overfitting was a problem.

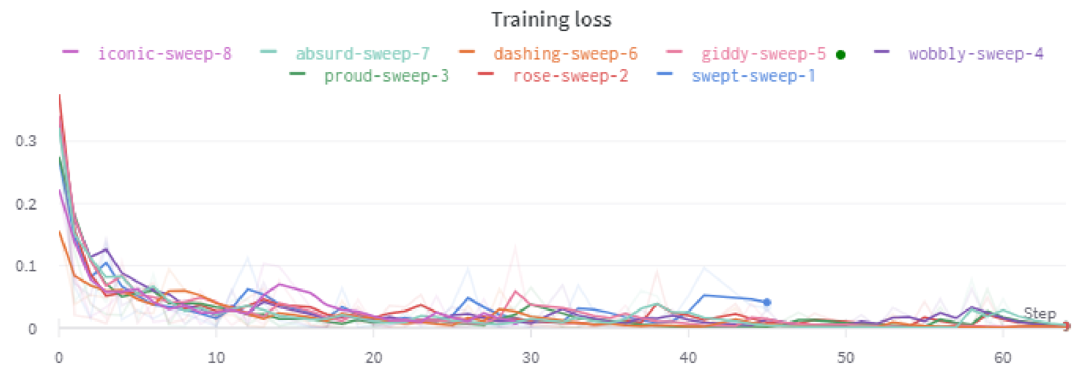

Figure 6.1: Showing training loss during parameter tuning.

The parameter tuning also provides some helpful insights into what values that might be the most promising and impactful when training the models. Table 6.1 showcases how the learning rate can alter the performance of the model. "Run 2" in the table shows almost double the "eval_loss" with almost four times as big "learning_rate". We can learn from this and use a relatively small learning rate when training our models.

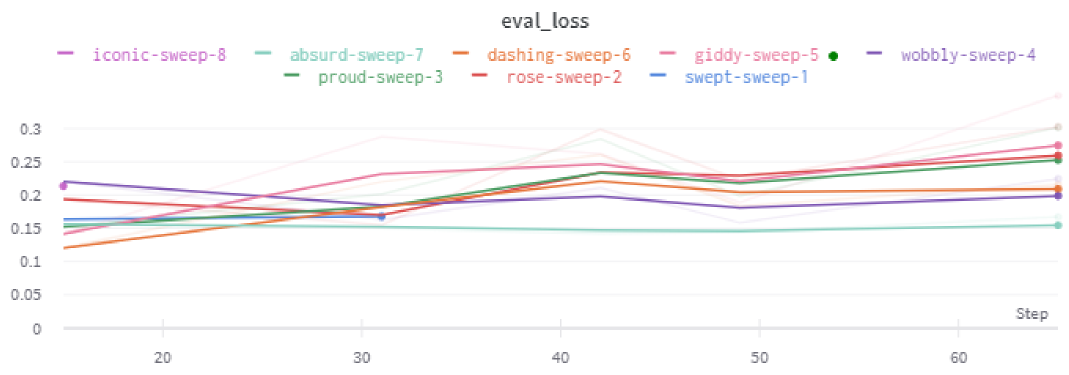

Figure 6.2: Showcasing evaluation loss during parameter tuning.

name	learning_rate	max_seq_length	eval_loss
Run 1	0.00012	52	**0.167**
Run 2	0.00040	56	0.303

Table 6.1: Some values and scores from parameter tuning.

6.3 Experimental Results

As both the test set and the benchmark data have a class imbalance, the scores shown in the result tables are all f1-score, as this will best measure performance in our case. Both of the following subsections will provide an explanation of the acronyms used in the result tables.

6.3.1 English Results

The reason we trained a model for only two epochs was to test what we found in section 6.2. The performance of this model is poor, so there might be other parameters that had to be tuned for it to perform better. The model trained on generated data and baseline data outperforms every other model. This adds more variety to the training data, and it clearly shows that this had a positive impact on the model. Scores on the benchmark data are bad for every model, and it has a hard time scoring above 0.5 f1-score. In the next chapter we will present a discussion on why this might be.

These are the acronyms for the datasets that the model has been trained on. This way, we can keep the tables clean while still providing the information needed to understand the tables. Unless specified otherwise, the models were trained on five epochs.

- BD stands for Baseline Data (557 467)
- GD stands for Generated Data (742 952)

- PD stands for Paraphrased Data (1 965 267)

Model Dataset		
	Test data	Benchmark data
RoBERTa + BD	0.793	0.470
RoBERTa + GD	0.820	0.490
RoBERTa + GD + BD	**0.920**	**0.542**
RoBERTa + PD	0.796	0.492
RoBERTa + GD, 2 epochs	0.817	0.471

Table 6.2: Final test results for models trained on English data.

6.3.2 Norwegian Results

Table 6.3 showcases the result for the multilingual models. We can see that almost every model performs better than the baseline data. Since BERT-multilingual is a multilingual model, we wanted to train in on English data and then test it on Norwegian data. As shown, this model has poor performance, so the generation of Norwegian data was worth it to increase the performance.

The model based on sentence-transformers is the best performing model, by far, on the benchmark data. However, we suspect that the scores are faulty. It has identical accuracy, recall, and precision. This is not normal on this dataset. Therefore we look at the results with a grain of salt.

- BD stands for Baseline Data (148 933)
- GD stands for Generated Data (98 787)

Model Dataset		
	Test data	Benchmark data
BERT-multilingual + BD	0.824	0.395
BERT-multilingual + GD	0.825	**0.500**
BERT-multilingual + GD + BD	**0.906**	0.438
BERT-multilingual + GD(English)	0.805	0.456
sentence-transformers + GD, 1 epoch	0.78	0.64

Table 6.3: Final test results for models trained on Norwegian data

Chapter 7

Discussion

This chapter aims to discuss topics related to the thesis but mainly to discuss the solution approach and results. The solution part discusses how the choices we made impacted the quality of the generated datasets and why we decided on these choices, including advantages and disadvantages. We will also discuss some other alternative decisions and how this might have impacted the solution.

7.1 Solution

In Chapter 5 we mentioned how and why we chose the properties that we did. The critical point was that the need of pre- and post-processing was low. Using these properties, we could generate hundreds of thousands of training instances in a mere few hours. The amount of data generated allowed us to experiment with the amount of training data needed to have the models perform to our needs. Therefore, looking at the results, we were confident that a too low amount of data did not cause the low scoring performance.

We decided that for every true claim, five false claims were generated. In the false claims, it was often either a single word or a date changed from the original claim. We decided to five false claims for every true claim because we wanted the model to be aware of these small changes. One word can change the correct label, and the models should notice these subtle changes. The downside of this is, of course, that the dataset becomes imbalanced in terms of labels. The model has seen more false examples than true examples and may then be biased towards predicting false. However, there are label imbalance in both the generated test set and the benchmark test set, so it might not impact the performance scores as much as first expected.

As mentioned, the false examples were generally generated by swapping a word, name, or date. A downside of this is that the model may become too sensitive to specific phrases. If a true birth date claim is always swapped with, e.g., ”24.05.1987”, the models could be trained to be too sensitive to ”24.05.1987”. It may appear to the model that a claim containing ”24.05.1987” will always be false. Of course, the data was generated by swapping random dates and names, but it may be possibility for it. A workaround could be to have more variety in the generated false claims by swapping more words or paraphrasing the claims.

We managed to reach our desired training data size with few combinations of properties. Having more variety of properties and thus more variety in the training data is, in most cases, a good idea. Nevertheless, this was easier said than done, and we had a hard time finding more of these properties that could generate lots of training instances. Therefore we settled for a lower variety in training data.

The amount of Norwegian data generated was quite a lot smaller compared the English generated data. The properties we used were the same in both languages. It may be worth trying to find more properties for Norwegian to increase the amount of generated data. However, it was even harder to find properties in Norwegian, as more analysis needed to be done. As the Norwegian Wikipedia is a lot smaller than English, there is also a limitation to how much data we can generate using this method.

Paraphrasing may not have been the best choice, as it did not improve the model. The reasoning for this might have been that the paraphrased claim-evidence pairs were too similar to the original. As mentioned earlier, if we did too much paraphrasing, the sentences will not keep their meaning and turn into useless data.

7.2 Training and Results

When training English models, we only tried with RoBERTa-base. For Norwegian, we only tried BERT-base and sentence-transformers. We chose these models based on experience from earlier projects. These models seemed to be the best performing pre-trained models for these tasks. However, it could be worth trying more pre-trained models to see if they fit this task better.

As seen in section 6.3 the score on the benchmark is bad. It struggles to get above 0.5 in f1-score. There may be two reasons for this. One, the benchmark was more dissimilar to our training than we first thought, and therefore not fitted to measure how these models perform. Or two, that the quality of our data does not meet the quality needed to perform well on the benchmark test set. However, if we see the scores, all the models outperform the baseline data. Even though the improvements are marginal, it may be a good place to start. Most of the models outperform the baseline data in the generated test set. These test data and benchmark data scores indicate that the data generation did improve the models.

Chapter 8

Conclusion and Future Directions

This thesis has looked into generating data for fact-checking using stance detection. Multiple datasets have been created for two languages. The most promising finding during this project is how sufficient it is to generate training data for a multitude of languages. As long as there are Wikipedia articles in the wanted language, training data can be generated.

All the models perform well on the generated test data, which consists of some properties the model has never seen before. In general, it seems that the more variety the training data have, the better it performs. We know this because three out of the four best scores are from a model trained on baseline data + generated data. This goes to show that variety in training data is good. It may be worth finding other methods to generate data so that we get even more variety. However, the models perform less than satisfactory on the benchmark data. Most models struggled to get above 0.5 in f1-score. This proves that either the generated data is too dissimilar to the benchmark data or the quality of the generated data is too poor.

All the models, however, that were trained on generated data outperform the baseline. Even though the results are bad, there is a slight increase in performance.

The most important part of the future directions would be to look into why the models perform so badly for the benchmark data and improve this.

Generating more data with different properties than we chose would also be a part of future directions. We can determine that the model performs better with more variety in the training data from our experimental results.

Lastly, it would be worth looking more into sentence-transformers, as this was by far the best performing model on the benchmark data. Although we can not trust the score

100% as we suspect it might have some flaws. If these bugs were fixed and the model performs well, it looks like the most promising model that we trained.

Hopefully, this thesis has presented a new and inspiring way of generating training data, especially for low resource languages like the Nordic languages. We think that this shows a feasible strategy to generate data, as the models are improving with our new data.

List of Figures

List of Tables

Bibliography

[1] Esquire. News article on the pizzagate conspiracy theory, 2020. URL https://www.esquire.com/news-politics/news/a51268/what-is-pizzagate/.

[2] Sinan Aral Soroush Vosoughi, Deb Roy. The spread of true and false news online, 2018. URL https://science.sciencemag.org/content/359/6380/1146.

[3] Amazon. Amazon mechanical turk. URL https://www.mturk.com/.

[4] LTG Oslo. Language technology group, uio.

[5] web64. Norwegian nlp resources.

[6] factcheck.org. Online fact-checking, 2021. URL https://www.factcheck.org/.

[7] snopes.com. Online fact-checking, 2021. URL https://www.snopes.com/.

[8] Mohnish Dubey, Debayan Banerjee, Abdelrahman Abdelkawi, and Jens Lehmann. Lc-quad 2.0: A large dataset for complex question answering over wikidata and dbpedia. In *Proceedings of the 18th International Semantic Web Conference (ISWC)*. Springer, 2019.

[9] Ann-Kathrin Hartmann, Edgard Marx, and Tommaso Soru. Generating a large dataset for neural question answering over the DBpedia knowledge base. 2018. URL https://www.researchgate.net/publication/324482598_Generating_a_Large_Dataset_for_Neural_Question_Answering_over_the_DBpedia_Knowledge_Base.

[10] Gottschalk Simon Souza, Tarcisio and Elena Demidova. Event-qa: A dataset for event-centric question answering over knowledge graphs. 2019.

[11] Wikidata.org. Wikidata main page, 2021. URL https://www.wikidata.org/wiki/Wikidata:Main_Page.

[12] w3.org. Sparql documentation, 2021. URL https://www.w3.org/TR/rdf-sparql-query/.

[13] MediaWiki. Documentation for wikitext, 2021. URL mediawiki.org/wiki/Wikitext.

[14] Hugging Face. Hugging face home page, 2021. URL https://huggingface.co/.

[15] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online, October 2020. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/2020.emnlp-demos.6.

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

[17] Jay Alammar. A visual guide to using bert for the first time. URL http://jalammar.github.io/a-visual-guide-to-using-bert-for-the-first-time/.

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

[19] Simple Transformers. Simple transformers home page, 2021. URL https://simpletransformers.ai/.

[20] Nils Reimers and Iryna Gurevych. Making monolingual sentence embeddings multilingual using knowledge distillation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 11 2020. URL https://arxiv.org/abs/2004.09813.

[21] James Thorne, Andreas Vlachos, Christos Christodoulopoulos, and Arpit Mittal. FEVER: a large-scale dataset for fact extraction and verification. *CoRR*, abs/1803.05355, 2018. URL http://arxiv.org/abs/1803.05355.

[22] Christopher Potts Samuel R. Bowman, Gabor Angeli and Christopher D. Manning. A large annotated corpus for learning natural language inference, 2015. URL https://nlp.stanford.edu/projects/snli/.

[23] Adina Williams, Nikita Nangia, and Samuel Bowman. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational*

Linguistics: Human Language Technologies, Volume 1 (Long Papers), pages 1112–1122. Association for Computational Linguistics, 2018. URL http://aclweb.org/anthology/N18-1101.

[24] Demian Gholipour Ghalandari, Chris Hokamp, Nghia The Pham, John Glover, and Georgiana Ifrim. A large-scale multi-document summarization dataset from the Wikipedia current events portal. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1302–1308, Online, July 2020. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/2020.acl-main.120.

[25] DocTranslator. Free online document translator, 2020. URL https://onlinedoctranslator.com/en/.

[26] Google. Document translation by google translate, 2020. URL https://translate.google.com/?sl=en&tl=no&op=docs.

[27] Jason Wei and Kai Zou. EDA: Easy data augmentation techniques for boosting performance on text classification tasks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 6383–6389, Hong Kong, China, November 2019. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/D19-1670.