UNIVERSITY OF
CAMBRIDGE
Department of Computer
Science and Technology

Explaining and Editing Language Models for Fact-Checking

Andrew Georgiou

King's College

June 2022

Submitted in partial fulfillment of the requirements for the
Master of Philosophy in Advanced Computer Science

I confirm that I have read the statement on plagiarism mentioned in www.admin.cam.ac.uk/univ/plagiarism/students/statement.html and that the work I have submitted for assessment is my own work, that I have not collaborated with anyone in its production and that it contains no unreferenced work of others.

I understand that the University reserves the right to use this piece of work for use in the detection of plagiarism.

Signed: Andrew Georgiou

Date: 06.06.2022

Total page count: 54

Main chapters (excluding front-matter, references and appendix): 39 pages (pp 8–46)

Main chapters word count: 10,218

Methodology used to generate that word count:

- Generated word count in OverLeaf online editor.
- Added \TC:ignore and \TC:endignore segments between pages < 9 and > 47.
- Ran 'Word Count' action in editor menu which uses TeXCount.

Abstract

Misinformation is a dangerous tool, causing confusion and distrust in order to spread panic and fear. Systems are being developed that can help disprove these false claims, these automated fact-checking systems test their performance through the 2018 FEVER shared-task. Research by Petroni et al. (2019) revealed that language models store implicit knowledge during pretraining that make them effective knowledge bases. Lee et al. (2020) developed an automated fact checking system that predicts a claim given no additional evidence by exploiting the implicit knowledge stored in BERT. Furthermore, we extended their work and built out a system that can be applied in a real-world setting through development of three components (i) We introduce RoBERTa to the task of isolated claim verification, improving accuracy on the task by 6%. (ii) We extended the hyper-network, *KnowledgeEditor* by De Cao et al. (2021) to learn the parameters of our claim verification model and attempt to correct obsolete knowledge whilst retaining accuracy. (iii) We developed a full system that enables our work to be used in a real-world context by generating explanations that allow us to evaluate the predictions made by our language model before trusting them. This system uses isolated evidence retrieval components to explain what information RoBERTa might have used from its pretraining. We applied LIME to understand which features of our input text were used in generating its prediction, and finally we treat our hyper-network as a probe that can learn the parameters used in generating a prediction. We exploited this to produce visualisations of our model that can be used to explain it's prediction.

Acknowledgements

This project certainly would not have been possible without the excellent guidance of my supervisors, Dr. Andreas Vlachos, Dr. Michael Schlichtkrull, and Dr. Zhijiang Guo.

I also want to thank my partner for her constant support during this incredibly stressful year and look forward to where the future may take us.

Contents

Chapter 1

Introduction

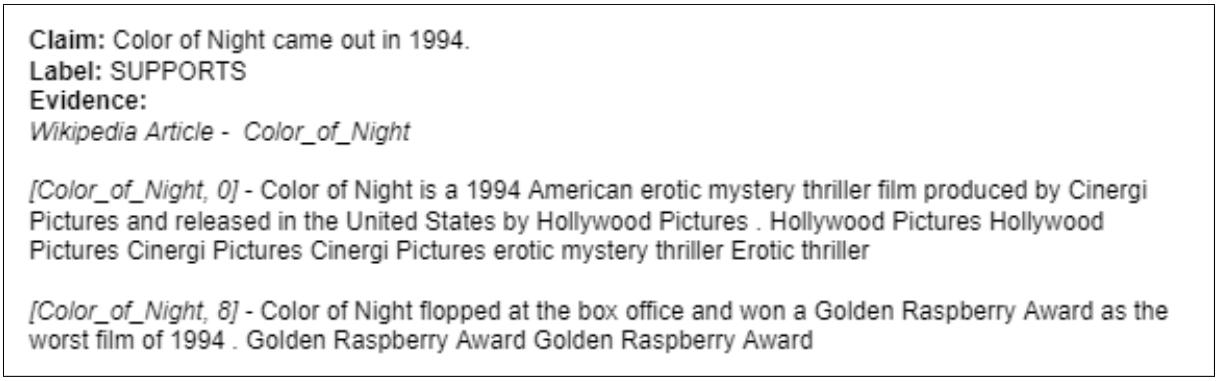

Figure 1.1: Example claim from the FEVER shared task - the claim requires two sentences from a single Wikipedia article as evidence.

Fact-Checking is a critical task in Natural Language Processing (Guo et al., 2022). Objectively validating claims made online can be used to prevent the spread of misinformation. The Fact Extraction and VERification (Thorne et al., 2018a), FEVER shared task challenged participants to a multiclass classification task to identify claims as either $SUPPORTED$, $REFUTED$ or $NOT\ ENOUGH\ INFO$. They were also challenged to retrieve supporting evidence from the 2017 Wikipedia dump (Wikipedia, 2017). An example claim with evidence is shown in Figure 1.1. The training set consists of 145,000+ claims, each with manually selected evidence. The final results saw participants break down the task into three separate components; *Document Retrieval*, *Sentence Selection* and *Claim Verification* (Thorne et al., 2018c).

The document retrieval component of the FEVER shared-task challenged participants to extract named entities from claims relating to a series of Wikipedia articles, such that they may be used as evidence to support or refute that given claim. Sentence selection tests the system's ability to identify up to five sentences within the extracted Wikipedia articles that can be used to verify a claim. The claim verification component combines the retrieved evidence with the claim to predict which of the three possible classes the claim belongs to.

The purpose of the sentence selection model in the original pipeline task is to feed evidence to the claim classification model; for our work we wish to use the output of the evidence retrieval components to explain the prediction of our claim.

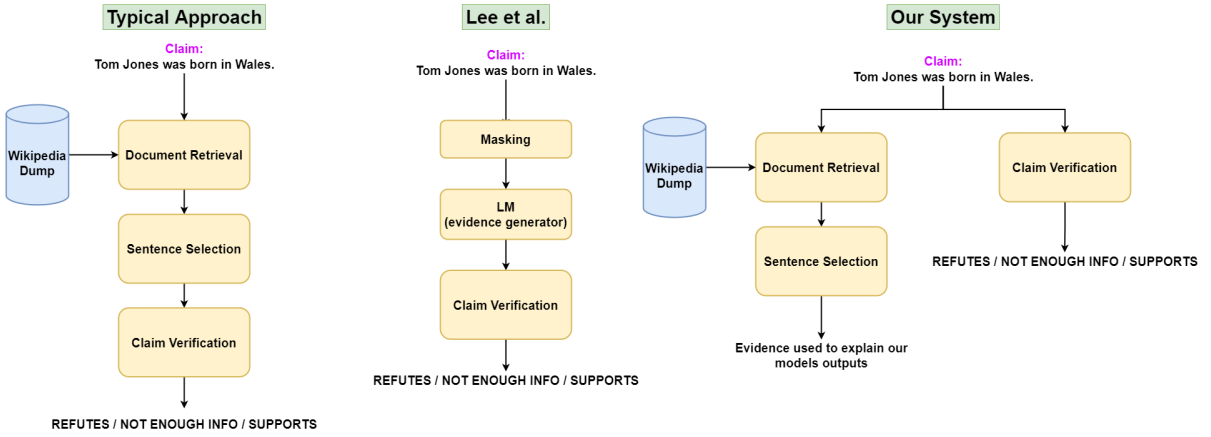

Figure 1.2: Typical pipeline approach (left) vs Lee et al. approach (center) vs our approach (right).

Typical claim verification systems attend to the shared-task by querying a knowledge base or an external source to verify claims (Ji, Grishman, 2011). This is not appropriate for *real-time* claim verification systems as they have to query and rank over 6.5 million Wikipedia articles per claim. Lee et al. (2020) has shown pre-trained language model BERT (Devlin et al. (2019), Vaswani et al. (2017)) can out perform the baseline fact-checking system implemented for the FEVER shared task (Thorne et al., 2018b). They did this without the requirements for explicit retrieval components. This represents a crucial milestone in building automated fact-checking systems, as it reduces the need for the multiple system pipeline as seen in the FEVER shared-task. Our project extends their work by also producing evidence alongside the claim verification task such that it can be used to explain and evaluate why the model generated its given output (Menick et al., 2022). Figure 1.2 shows how our approach expands upon their system by developing the retrieval components independent of the claim verification component such that it can be used to manually evaluate and explain our claim verification models prediction.

Over time the knowledge encoded in pre-trained language models can either not be encoded correctly or become out dated (Jang et al., 2021). It is infeasible to re-train the whole language model to update this knowledge. Re-training a model such as BERT required 64 GPUs training for over 4 days at a cost of $7000+ (Sharir et al., 2020). Fine-tuning the model on the updated knowledge is unstable and has the hazarding affect of not maintaining other encoded knowledges predictions and therefore causes a significant decrease in overall model performance (Mosbach et al., 2021). This problem prohibits language models from being a reasonable solution for the task of fact-checking, therefore obsolete data must be updated within the model whilst retaining previous knowledge that remains valid.

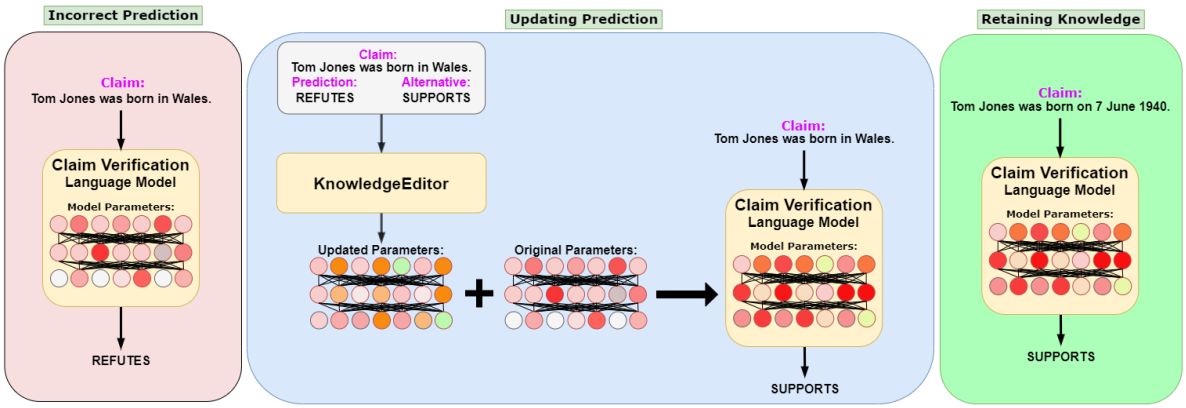

Figure 1.3: Our implementation of KnowledgeEditor fixing an incorrect prediction (left) and conditioning the model to an alternative label (center) whilst retaining knowledge (right)

Techniques exist that allow us to edit language models (Sinitsin et al. (2020), Mitchell et al. (2021)). KnowledgeEditor (Cao et al., 2021) uses a hyper-network (Krueger et al., 2018) with constrained optimization to probe and alter data without affecting other knowledge. This work was tested on a BERT language model for fact-checking and BART (Lewis et al., 2019) for closed-book question answering. The fact-checking system was only evaluated on the model's ability to randomly condition binary labels whilst retaining accuracy, and did not test the systems capacity to condition incorrectly predicted data for the full FEVER multi-classification task. Figure 1.3 shows what our system looks like for editing incorrect predictions whilst retaining other knowledge using our trained KnowledgeEditor hyper-network.

Cao et al. (2021) discovered the hyper-network can be regarded as a probe that reveals which parameters within the networks encode task specific knowledge. Through Local Interpretable Model-Agnostic Explanations (LIME) we can also generate visualisations for how our system uses words from our input in generating its predictions. Figure 1.4 shows a LIME visualisation of our sentence selection model predicting a claim, evidence pair. RoBERTa is one such model that we intend to edit to gain insight into its parameters, and evaluate its knowledge and performance on the task of claim verification. The approach we will take to this problem will be building our own hyper-network KnowledgeEditor which will extend beyond conditioning on binary logits to conditioning on multiple classes. We also wish to use KnowledgeEditor to experiment with fixing incorrect knowledge in our model whilst retaining our accuracy on the FEVER shared-task.

Text with highlighted words

Color of Night came out in 1994. Color of night flopped at the box office and won a golden raspberry award as the worst film of 1994.

Figure 1.4: LIME Visualisation of word weightings for sentence selection model trained on sentence similarity task.

Our projects final system consists of many parts, to determine if we have reached our goals we define a desiderata below:

1. Improve the claim verification model proposed by Lee et al. (2020) to increase label accuracy on isolated claim classification.
2. Extend KnowledgeEditor to work on our multiclass classification task, and evaluate the performance of editing incorrect knowledge.
3. Develop an explanation system that uses evidence retrieval components, LIME and our hyper-network to allow us to explain and evaluate our claim predictions.

We intended to achieve our desiderata by completing the following, We started by expanding upon the work of Lee et al. (2020); we did this by fine-tuning larger language models such as RoBERTa for claim verification. We further extended their work by implementing evidence retrieval components that can be used to help explain the predictions made by our language models. We then extended our model by continuing the work of (Cao et al., 2021), which investigated utilising a hyper-network to edit this implicit knowledge whilst retaining classification accuracy.

Our third desideratum was the main goal we intended to achieve for this project. We accomplished this by using the language model to select its own evidence, developing KnowledgeEditor to learn the parameters of our models and produce as output the parameters that are essential in editing predictions relating to our task, applying Local Interpretable Model-Agnostic Explanations (LIME) (Ribeiro et al., 2016) to explain individual predictions, and finally manually evaluating this information to help us reveal what components of our language models are important to the task of fact-checking and identifying if they use implicit knowledge to generate predictions (Petroni et al., 2019).

Due to the complexity of the project, for the remainder of the report each section will be broken down into the different components, the first three being the pipeline components of the FEVER shared-task. Document Retrieval, Sentence Selection and Claim Verification. The final two components being the Hyper-Network and LIME for explainability.

Chapter 2

Background

This section covers the background knowledge required to understand the remainder of the report. Introducing key terms for techniques and models used in the development of this project. It assumes solid foundations in Computer Science but not the specific field of Natural Language Processing.

2.1 Word Embeddings

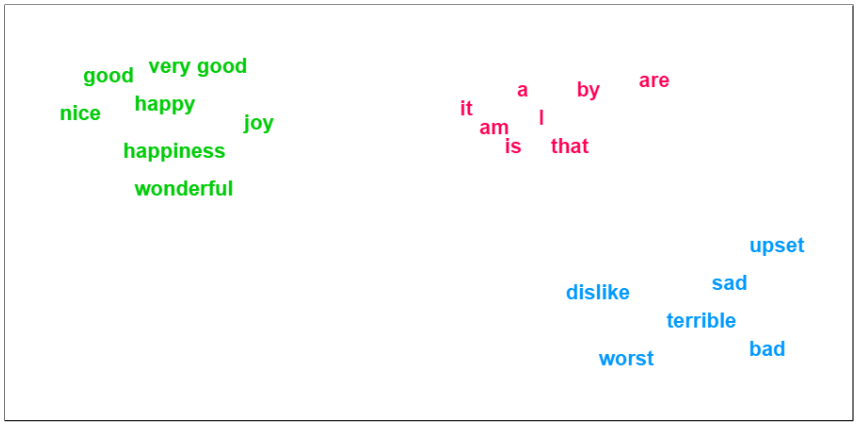

Figure 2.1: A two-dimensional projection of embeddings generated from some words and phrases. Originally trained for sentiment analysis.

Word embeddings are dense vectors that provide a powerful representation of words and how they relate to corresponding words within a vocabulary. Embeddings are an important topic in Natural Language Processing (Camacho-Collados, Pilehvar, 2020) as they provide a rich feature representation of words that can be used to calculate semantic similarity between articles of text and facilitate in classification tasks.

Plain text cannot be used as input to neural networks as they perform calculations on tensor values. We therefore need to build a way to represent our full list of possible words

in the dataset as a one-hot vector. Building a vocabulary is simply creating a set of words from our training set and giving each unique word an index i.

Skip-gram model is one of two models implemented in the popular Word2Vec architecture (Mikolov et al., 2013), it is a neural network model with one hidden layer that we train on pairs of words within a context window. The output of the model is a 300 dimension feature space. It then gives the model an artificial task to perform, given an input word from a sentence, select a random nearby word and output the probability of choosing that word given the entire models vocabulary.

Similarity measures compute the closeness of two words v, u through their vectors of length $|V|$. The most popular similarity metric used is cosine similarity, this computes the cosine of the angle between the two word vectors.
Most similarity measures are based on the dot product operation f which takes as arguments the two word vectors v, u:

$$f(u, v) = v \cdot \ u = \sum_{i=1}^{N} v_i u_i$$

The dot product is not normalised for vector length, therefore the dot product is higher for longer vectors. Greater frequency words have more co-occurence with other words which results in a higher dot product value for frequent words. This is a problem that is solved by normalisation. Dividing the dot product output by the lengths of both word vectors is equivalent to the cosine of the angle between both vectors. This is the cosine similarity operation:

$$\frac{v \cdot u}{|v||u|} = cos\ \theta$$

Embeddings and measurements of similarity are used quite often throughout this project, the skip-gram model is one of many different unsupervised learning techniques used to generate word embeddings. Most commonly associated with Word2Vec alongside continuous bag of words. GloVe (Pennington et al., 2014) and FastText (Joulin et al., 2016) are other unsupervised learning algorithms that will be mentioned throughout this report. GloVe intends to capture global and local statistics unlike Word2Vec which only captures local information. FastText uses the same algorithms found in Word2Vec with the addition of character level n-grams to enable vectorising words not found in the original vocabulary by breaking them down to their substructures.

2.2 Pre-trained Language Models

Pre-trained language models (PLM) (Vaswani et al., 2017) represent a large shift in the field of NLP. They have enabled easy adoption and quick development of state-of-the-art models on many downstream tasks. Pre-trained contextual encoders represent the second generation of models following on from the pre-trained embeddings discussed in section 2.1. The most successful context encoders are large transformer models, stacking many encoder-decoder layers with scaled dot-product attention throughout, instead of reading input left-to-right the transformer stack takes the full text sequence as input. It therefore encodes for true bi-directional context.

When trained with a large enough corpus, the transformer architecture enables these PLMs to learn universal language representations. These models can then be fine-tuned to adapt them for many downstream tasks. Recent work includes the application of PLMs such as Bi-directional Encoder Representation from Transformers **(BERT)** (Devlin et al., 2019) in document retrieval, sentence selection and claim verification, which produced state-of-the-art results (Soleimani et al., 2019).

PLMs differ slightly in their unsupervised learning task. The original BERT is a masked language model with next sentence prediction:

Masked Language Model The MLM task replaces 15% of the words in each sequence with a [MASK] token. It then attempts to predict the masked word given the context of the other words in the sequence.

Next Sentence Prediction NSP feeds a pair of sequences as input and tasks the model with predicting if the posterior sequence in the pair follows the the prior sequence. Combining both sequences involves concatenating sequences together and inserting a [SEP] token in the middle, denoting a separation of sequences.

A Robustly Optimized BERT Pretraining Approach **(RoBERTa)** (Liu et al., 2019b) is another pre-trained language model that we intend to use in our system. Trained on 160GB of data, RoBERTa removed BERT's next sentence prediction pre-training task and changed the static masking system to dynamic masking. This change in masking function avoiding reusing the same mask for a sequence at each epoch. As a result of these changes RoBERTa outperformed BERT in all individual tasks on the General Language Understanding Evaluation (GLUE) benchmark (Wang et al., 2018).

BART (Lewis et al., 2019) is a sequence-to-sequence model trained as a denoising autoencoder. They use a complete encoder-decoder transformer architecture, in training they produce noise in an input sequence and have an autoregressive decoder attempt to recreate the original input. BART takes a text sequence as input and outputs another sequence as output. This is useful in tasks such as machine translation, text summarisation (Widyassari et al., 2022) or sentence entailment. This means given two or more

sentences the model can evaluate if they are related to a given statement.

PLMs can be fine-tuned to a large range of downstream tasks, this involves tuning the weights and biases of all the layers in our language model. We then apply a linear classification layer to the end of our models architecture to produce the required number of classes as output.

2.3 Fact-Checking Pipeline

The fact-checking pipeline represents the three sub-components of the fact-checking task. These sub-components were commonly used by participants in the original FEVER shared-task and current SoTA fact-checking systems. These components are not necessarily always required as researchers can merge or isolate them as needed. This section provides a high level overview of each component with a graphical representation of the input and outputs to each system.

2.3.1 Document Retrieval

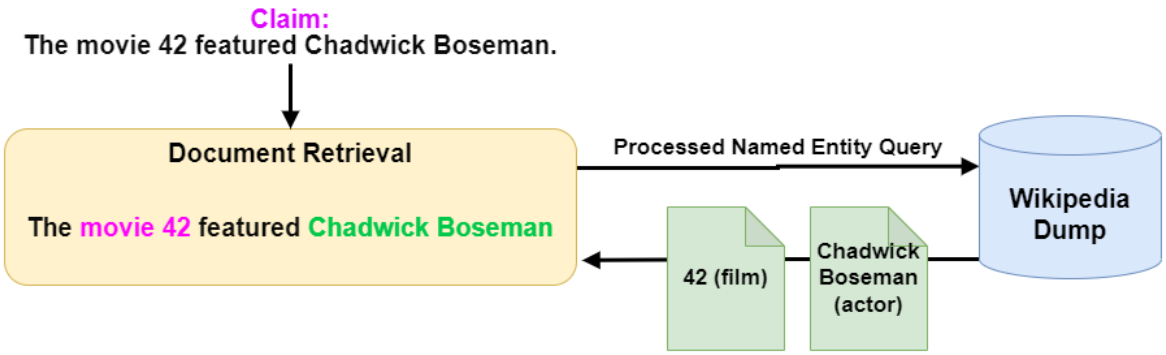

Figure 2.2: Example of document retrieval process, handles identifying named entities (left) and retrieving documents from wikipedia dump (right).

The document retrieval component of the FEVER shared-task consists of identifying the k-most relevant documents from the Wikipedia corpus for a given claim. Figure 2.2 shows a simple example of a document retrieval system. This system extracts named entities from the claim before processing them to a format that can be used to query the Wikipedia data dump to extract the relevant Wikipedia articles for our claim. Many claims can require multiple documents for evidence, therefore the document retrieval component must also rank document titles by relevancy.

2.3.2 Sentence Selection

Sentence selection aims to rank sentences from the extracted documents by their similarity to a claim. Figure 2.3 shows the input and outputs of this component. It takes as input a

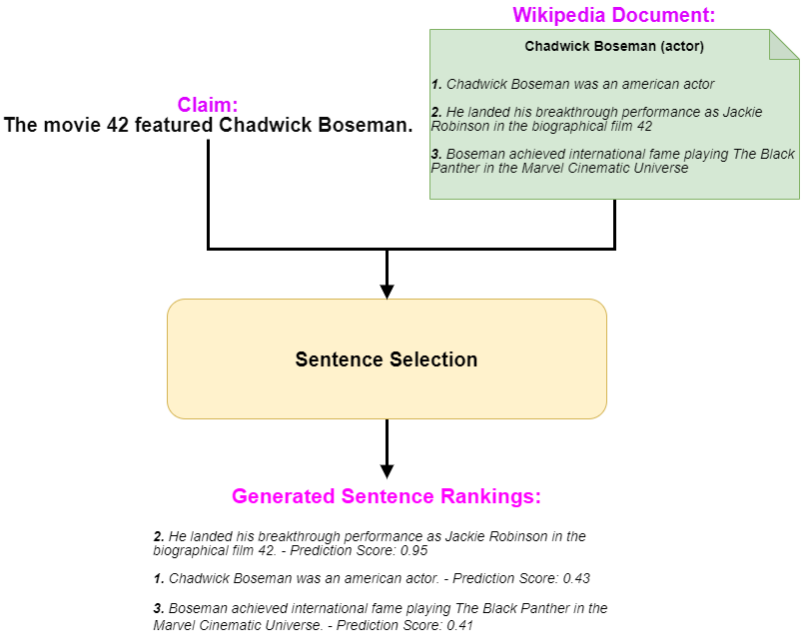

Figure 2.3: Example of sentence retrieval component, ranking sentences from a document based on similarity to a given claim.

claim and sentences extracted from the documents retrieved through document retrieval, it then calculates the similarity between claim and sentence, sorts the evidence by this ranking and outputs the final list. Note, only the top five sentences are used for scoring in the fever-scorer system.

There are two main approaches to generating sentence rankings; pairwise and pointwise sentence retrieval (Soleimani et al., 2020). Pointwise approaches predict a single tensor value for a ⟨claim, sentence⟩ pair. This can then be used to rank documents by prediction score. Pairwise approaches compare two sentences extracted for a claim and attempts to come up with an optimal ordering between pairs to identify the top-k rankings.

Only the document name and sentence indices are required for evaluation in the FEVER shared-task scoring system.

2.3.3 Claim Verification / Natural Language Inference

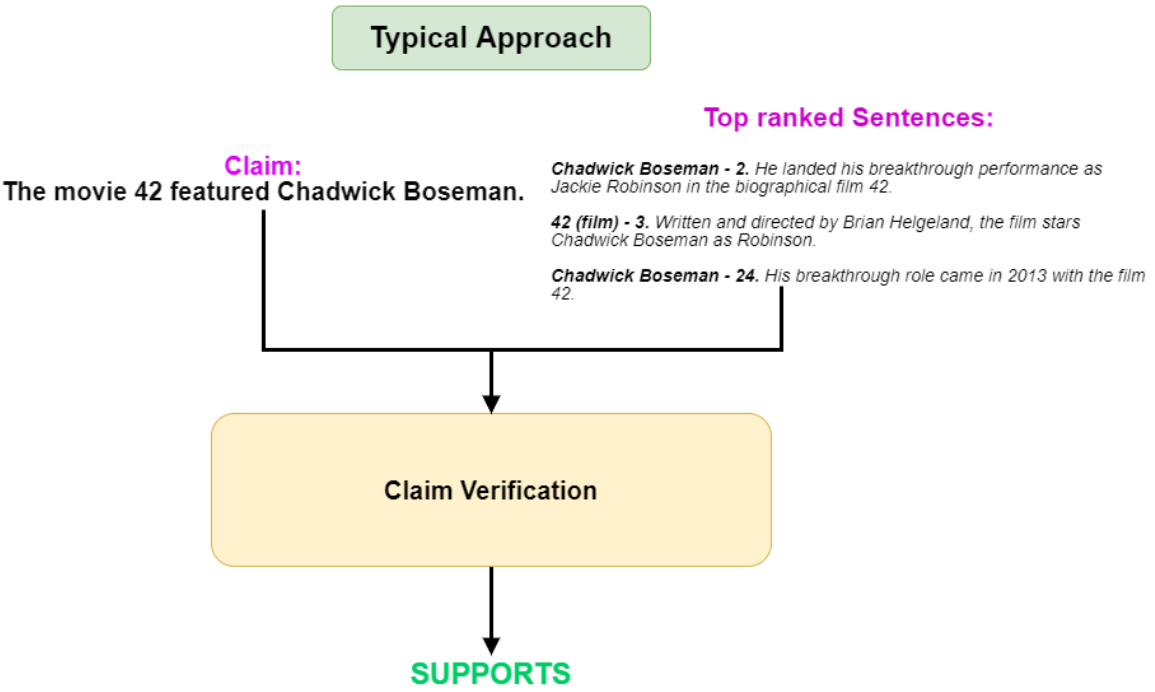

Figure 2.4: Example of typical claim verification system, utilises claim and k-top sentences to evaluate if SUPPORTS, REFUTES or NEI for claim.

Figure 2.4 represents a typical approach to the claim verification component. The goal of claim verification is to classify the claim to one of three classes $SUPPORTS$, $REFUTES$ or $NOT\ ENOUGH\ INFO$ (NEI). The most common approaches to this task are either pairwise classification that aggregates the result into a single score or evidence concatenation that combines all the evidence into a single sequence that is used to classify the claim.

Our approach differs from the typical approach in that we do not utilise evidence in making our prediction. We use the implicit knowledge that language models gain through pre-training to generate our prediction by only using the claim as input.

Chapter 3

Related work

In this section, we aim to build up the relative research in each component of our system that it positions our project in respect to more recent work. Capturing the need for a system that extends the work of Lee et al. (2020).

3.1 Language Models as Fact Checkers?

This project expands upon the work "Language Models as Fact Checkers?" (Lee et al., 2020). They utilised implicit knowledge in language models based on work by (Petroni et al., 2019), through exploiting the masked language modelling task in the pre-training of models such as BERT (Devlin et al., 2019). They produced a label accuracy of 0.49 in their masked language approach and 0.57 through fine-tuning all layers of BERT on the task, showing that language models perform better than a random baseline accuracy of 0.33. This demonstrates that language models must use some inference to generate accurate predictions.

The original paper did not attempt to explain what information the model must have used in generating its predictions (Aken van et al., 2019). Developing an understanding of why models generate their predictions is important (Gohel et al., 2021), as language models are seen as black box models. With their huge number of parameters these models are uninterpretable for humans, as we cannot track the basis of their decisions which could be full of bias, outdated, or wrong decisions that are missed in using the standard evaluation metrics (Bhardwaj et al., 2020).

3.2 Document Retrieval

The original FEVER baseline model by Thorne et al. (2018c) attended to this task by using a solution originally created by Chen et al. (2017a). They compared articles and claims as a term frequency-inverse document frequency (TF-IDF) weighted bag of word

vectors with bigram counts and optimised for speed through hashing bigrams to map to 2^{24} bins. When returning $k = 5$ documents, they reported a baseline oracle score of 70.20. The oracle score (OFEVER) is the strict FEVER system score if we assume perfect downstream components.

Teams UNC-NLP (Nie et al., 2018) and Athene UKP TU Darmstadt (Hanselowski et al., 2018a) came 1st and 3rd respectively in the original 2018 FEVER shared-task. Introducing novel approaches to document retrieval that influenced our approach to the problem:

UNC-NLP combined multiple techniques for their best performing document retrieval system, combining keyword matching with reranking based on pageview frequency before feeding the documents into a Neural Semantic Matching Network (NSMN). NSMN uses Bi-LSTM encoding and matching layers to identify the semantic similarity between two textual sequences. They reported an oracle fever score of 92.42 for k=5 documents.

UKP-Athene achieved an evidence recall of 85.19, the highest recall achieved in the shared task. They achieved this through an *entity linking* approach to the document retrieval sub task. *Mentioned extraction* uses an AllenNLP (Gardner et al., 2018) constituency parser to extract noun phrases to identify potential entities. They combined this with a *candidate article search* that utilises the MediaWiki API to identify Wikipedia articles with matches to potential entity mentions found within a given claim. Finally filtering out potential candidate articles longer than the entity mention if it does not overlap with the rest of the claim.

These approaches achieve high document recall but are context independent methods that can miss identifying named entities belonging to different classes such as the difference between a book or movie of the same name. This signifies a need for an approach that exploits transformer architecture and the encoder mechanisms ability to generate context-dependent embeddings.

The Generative ENtity REtrieval system **(GENRE)** (De Cao et al., 2020) is built using sequence-to-sequence architecture (Sutskever et al., 2014) to generate entity names autoregressively conditioned on context. GENRE uses the pre-trained transformer-based model BART (Lewis et al., 2019) that is fine-tuned to generate entity names. The largest benefit of GENRE is its ability to utilise a constrained decoding strategy. This constrains generated names to exist within a predefined candidate set. GENRE has only been experimented on document retrieval accuracy on the FEVER set taken from the KILT benchmark set (Petroni et al., 2021) achieving 83.6% R-Precision. R-Precision is defined as $\frac{r}{R}$ where r is the number of relevant documents selected correctly from the list of possible relevant documents in the collection R. The KILT task differs from the FEVER shared-task as it removes the 'NEI' class and uses one article per claim prediction.

The current state-of-the-art systems in the shared task all use the same techniques used

by UKP-Athene (Hanselowski et al. (2018a), Soleimani et al. (2019), Zhou et al. (2019), Wang et al. (2019)), DREAM (Chernyavskiy, Ilvovsky, 2019) used a slight variation on Hanselowski et al. (2018b). Post-extraction their component determines document relevancy through calculating cosine similarity between TF-IDF embeddings of the claim and sentence within the document. There has been little improvement of this component since the shared-task in 2018, this could open the space for new development of techniques that can further push the full FEVER benchmark.

3.3 Sentence Selection

The three most common approaches are: *keyword matching*, *supervised classification* and *sentence similarity scoring*. Both UNC-NLP and UKP-Athene used a supervised binary classification approach to this task.

UKP-Athene attended to the task by modifying the Enhanced Sequential Inference Model (ESIM) originally developed by Chen et al. (2017b). to generate a ranking score between two input sequences instead of predicting the entailment relationship. ESIM encodes premise and hypothesis through a Bidirectional Long Short-Term Memory (Bi-LSTM) (Schuster, Paliwal, 1997) layer, aligning the encoded sequences using a bidirectional attention mechanism. Finally applying a softmax output that classifies the max and mean pooled representations of the final Bi-LSTM.

UNC-NLP re-uses the NSMN from their document retrieval component, during this stage they restrict the search space. This limited space is transferred to the sentence selection component to generate the probability of sentence_i being used as evidence for the claim.

The majority of current state-of-the-art systems in the FEVER full task (Liu et al. (2019c), Chernyavskiy, Ilvovsky (2019), Zhong et al. (2019), Soleimani et al. (2019)) all make use of pre-trained language models for pointwise and pairwise sentence retrieval. These approaches to sentence selection have proven to be very successful in achieving over 88.0% recall on the FEVER development set. Soleimani et al. (2019) showed using the base-BERT model with proper parameter tuning can perform better than RoBERTa in both pairwise and pointwise retrieval systems.

3.4 Claim Verification

Claim Verification in the original FEVER shared-task was solely approached as a supervised learning task that combines evidence sentences with the claim (Thorne et al., 2018a). UNC-NLP use a neural semantic matching network that expands the natural language inference model by using GloVe and ELMo embeddings (Peters et al., 2018) concatenated with WordNet (Fellbaum, 1998) one-hot vectors for signaling for specific hypernym, antonym or edge distance based phenomena and a sentence relatedness score

taken from the sentence selection module. This gave the UNC-NLP team a really powerful set of features for claim verification although expensive in their generation.

ESIM has seen popularity in this task as many current SoTA systems still use it (Yoneda et al. (2018), Hanselowski et al. (2018b), Hidey, Diab (2018), Nie et al. (2018)), UNC-NLP's NSMN model being a modification of ESIM with added shortcut connections from input to matching layer with adjustments to the alignment layer to only use max-pool. The Papelo team (Malon, 2018) applied pre-trained language models to this task achieving a label accuracy of 0.6074, representing the shift to current state-of-the-art models that focus on the transformer network architecture.

(Stammbach, Neumann (2019), Nie et al. (2018), Chernyavskiy, Ilvovsky (2019), Portelli et al. (2020), Jobanputra (2019), Ye et al. (2020), Lee et al. (2020)) all exploited the use of transformer models for the task of claim verification. In the current post shared-task challenge hosted on CodaLab, the highest performing team in claim verification *ProoFVer* (Krishna et al., 2021) achieved a label accuracy of 0.7947 using a seq2seq BART model for proof generation with majority voting.

3.5 Editing Language Models

Sinitsin et al. (2020) introduced the method of editable training to correct model mistakes without influencing behaviour of other samples. This work was motivated by a desire to correct mistakes in training that could produce mistaken results if not fixed. They showed they could edit 100 samples on ResNet-18 to reduce test error rate by over 0.2%. This proved language model editing to be viable research area.

Zhu et al. (2020) approaches this problem by means of constrained optimisation through re-fine tuning on their task with altered data. Their method uses a norm based constraint between the original model and the updated one. This ignores understanding how the parameters may affect the output. Sinitsin et al. (2020) and Zhu et al. (2020) both ignored evaluation of *accuracy retained* by the model upon editing a piece of information. This is critical to assess when we are interested in fixing knowledge to improve our model, not retaining accuracy per update would result in a net negative loss in our final models accuracy.

3.5.1 KnowledgeEditor

Cao et al. (2021) proposed KnowledgeEditor, a hyper-network approach for editing knowledge in language models. Their hyper-network attend to this task with a goal of maximising retained accuracy for each edit performed. They train a hyper-network with constrained optimisation which is tasked with predicting the required weight updates to the original models state dictionary to condition a given input to a new label. They

perform editing on a BERT model applied to FEVER for fact-checking and a sequence-to-sequence BART model for question answering (Lewis et al., 2019).

Cao et al. (2021) does not evaluate the performance of editing BERT on the FEVER task. Instead they evaluate the ability to randomly condition a claim to a binary label output of either *SUPPORTS* or *REFUTES*. The FEVER shared-task is a multi-classification task and as such we cannot truly evaluate the performance of editing language models without applying this approach to the full task. This includes the *NEI* label prediction. In their evaluation they also do not perform multiple edits as the hyper-network only learns the current parameters of the model.

In their original paper, the author noted how KnowledgeEditor can be regarded as a probe which reveals what components need to be changed in order to alter factual knowledge and how subsequent updates show how updates are concentrated to a small set of components. Probing our model can help us in understanding how our model generates its predictions and explaining its output.

3.6 Explainability

The black box nature of artificial intelligence means predictions made by models are often blindly trusted. In the fields of health, law and fact-checking the decisions made by these models must be explainable or interpretable. Explainability is beginning to become a legal requirement for artificial intelligence with the European Commission in February 2020 publishing The White Paper - On Artificial Intelligence (Commission, 2020) which highlighted the need for transparency in order to have trustworthy AI. Trust is the most essential component of any fact-checking system that is to be applied to the real world.

Glass et al. (2008) first introduced the idea of trust and understanding for complex agents, Jacovi, Goldberg (2020) defines faithfulness as reflecting the true reasoning process of the model when making a decision. Their exist two main approaches to explainability; interpretable models and post-hoc explanations (Madsen et al., 2022). Approaches that looked into BERT as an interpretable model through analysing individual nodes discovered an 'interpretability illusion' that can arise where seemingly consistent patterns in the models output turn out to be contingent only upon the test set itself and are not generalisable beyond that data (Bolukbasi et al., 2021). This prompts the use of post-hoc explanations, Locally Interpretable Model-agnostic Explanations (LIME) (Ribeiro et al., 2016) and Shapley Values (SHAP) (Lundberg, Lee, 2017) are well known post-hoc explanation methods that we explored to develop a better understanding of what information our model used to generate its outputs.

LIME attempts to understand decisions locally by permuting over the input data and identifying which permutations had the largest affect on the output.

SHAP is an inhereted solution from cooperative game theory. It treats each feature as a player in a game where the prediction is the payout. Cooperation between players (features) receives a certain profit, these cooperations are called a coalition. The Shapley value is the average marginal contribution of a feature value across all possible coalitions.

Both models behave similar in that they output the feature importance of text for single predictions, the benefit of LIME over SHAP is speed. SHAP computes all possible permutations to provide global and local consistency whilst LIME builds sparse linear models around a single prediction.

Chapter 4

Methodology

The methodology chapter is sorted in implementation order, the normal approach is to develop each stage of the pipeline in order of data flow. We are following the methology of Lee et al. (2020) such that our classification base models were designed and trained first, we then developed our retrieval components after in order to supply explanations for our classification outputs. The final components are LIME for further explainability and KnowledgeEditor to evaluate future research.

4.1 Claim Verification - Fact-Checking Model

Model Architecture:
BERT-base is a 12 layer model with a 768 dimension hidden size, 12 self-attention heads and around 110 million trainable parameters. BERT has a vocabulary size of 30,000 tokens. It's pre-training task utilised the masked word prediction task and next sentence prediction.
RoBERTa-base is also a 12 layer model with 768 dimension heads as the model is based on the BERT-base model it therefore also has the same 121 self-attention heads. It begins to differ from BERT in its 125 million parameters and 50,000 token vocabulary size. RoBERTa varies in pre-training as it implements dynamic word masking and removes the next sentence prediction task.

Task
We want to produce a model that when presented with a claim as input can output the probability that the claim belongs to one of three classes $SUPPORTS$, $REFUTES$ or $NOT\ ENOUGH\ INFO$. We require the model to not take as input any evidence for the input claim and still perform better than a random baseline accuracy of 0.33 for an even class distribution. We therefore seek to optimise the models parameters for the task whilst retaining the models implicit knowledge.

Definition

Formally we have a fact-checking model $c \mapsto f(c)$, and a dataset $\langle c, y \rangle \in D$ of claim, label pairs where c is the input to our model and y is an index i that maps to a string label in our dictionary $\{0 : REFUTES;\ \ 1 : NEI;\ \ 2 : SUPPORTS\}$. The output of our model is the logits p of length 3 which is used in our loss function $\mathcal{L}$.

We applied cross entropy loss as our loss function $\mathcal{L}$ with mean entropy smoothing to our model. Precisely we define the loss $\mathcal{L}$ as:

$$\left[-\sum_{b=1}^{\mathcal{N}} \sum_{c=1}^{M} y_{\phi f(\mathcal{B}_b),c} \cdot log(\sigma(f(\mathcal{B}_b))_c) \right] - s \cdot \left[-\frac{\sum_{b=1}^{\mathcal{N}} \sum_{c=1}^{M} \sigma(f(\mathcal{B}_b))_c \cdot log(\sigma(f(\mathcal{B}_b))_c)}{\mathcal{N}} \right]$$

Where $\mathcal{B}$ of length $\mathcal{N}$ is the batched output tensors generated through a forward pass of our claims through the model $c \mapsto f(c)$ for our current training step, M, the number of classes for our task (which we set at 3). ϕ denoting an argmax function and $y_{\phi f(\mathcal{B}_b),c}$ being a binary indicator (0 or 1) if our label y is equal to the argmax of our logits at index b. σ is our sofmtax function which $\sigma(f(\mathcal{B}_b))_c$ represents the probability of class c for index b in our batch once fed through our model. s is our smoothing hyperparameter which we default as 0.1, this smoothing parameter is multiplied by the mean entropy categorical distribution.

4.2 Document Retrieval - GENRE

Assuming a collection of Wikipedia articles $\mathcal{E}$, given a claim c. GENRE is tasked with returning the most relevant articles from $\mathcal{E}$ with respect to c. Each article $e \in \mathcal{E}$ is assumed to have an article name.

GENRE uses the sequence-to-sequence model BART to generate entity names. GENRE does this by using an autoregressive formulation to rank each article $e \in \mathcal{E}$. The formulation; $score(e|c) = P_z(y|c) = \prod_{i=1}^{N} P_z(y_i|y_{<i}, c)$ where y is a set of tokens of length N used to identify e, and z is the parameters of the BART model.

GENRE is trained using an objective commonly used for neural machine translation, that is to say maximising $log\ P_z(y|c)$ with respect to the models parameters z.

Computing the score for every element in the set $\mathcal{E}$ is very expensive to compute when the Wikipedia Dataset exceeds 5M+ article titles. Therefore Cao et al. (2021) exploits Beam Search to efficiently explore the search space. We find the top k entities in $\mathcal{E}$ decoding from the model through performing beam search with k beams. Traditional Beam Search allows for generation of any token from the vocabulary at each decoding step. This can result in generation of documents outside the set $\mathcal{E}$. Therefore Constrained Beam Search is required, this limits the next possible token only to that which exists in the set $\mathcal{E}$. They create this constraint through a trie $\mathcal{T}$.

The trie structure $\mathcal{T}$ has nodes that are annotated with the tokens t generated by the vocabulary. The root node of the trie structure $\mathcal{R}$ defines the starting node for each search. Each node $n \in \mathcal{T}$ has children that are the allowed paths within the set $\mathcal{E}$, each leaf node and some children nodes represent the full tokenised articles e from the set $\mathcal{E}$. Algorithm 1 defined below demonstrates the necessary function to generate the trie object given the set of tokenised documents $\mathcal{E}$.

Algorithm 1
Generating Documents Trie

Require: Tokenised Documents $\mathcal{E}$
  $\mathcal{T} \leftarrow \{\}$
  $\mathcal{R} \leftarrow \mathcal{T}$
  for all $e_i \in \mathcal{E}$ **do**
    $\mathcal{T} \leftarrow \mathcal{R}$
    for all $t_j \in e$ **do**
      if t_j not in $\mathcal{T}$ **then**
        $\mathcal{T}[t_j] \leftarrow \{\}$
      end if
      $\mathcal{T} \leftarrow \mathcal{T}[t_j]$
    end for
  end for
  return $\mathcal{R}$

4.3 Sentence Selection

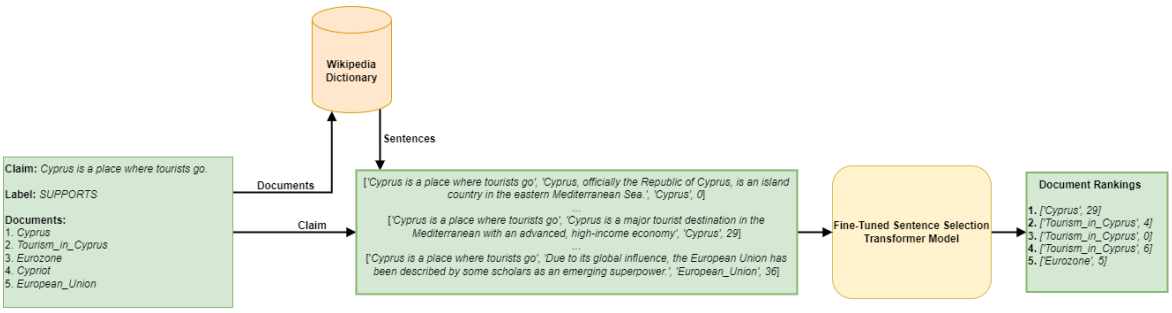

Figure 4.1: End-to-end Evidence Selection Pipeline. Extracting sentences using Wikipedia dictionary, building new dataset E to generate document rankings for final retrieval output.

Sentence selection was the last developed component of our system, it was developed to explain our baseline claim verification model therefore we use the same architecture, with a few optimisations to training for this pointwise task. We demonstrate the structure our sentence selection system in Figure 4.1.

Task

We want to fine-tune a model $f = (c, s) \mapsto \hat{y}$ where we have $\langle c, s, y \rangle$ values in our dataset D, where $c \in D$ is our claim, $s \in D$ an evidential sentence and $y \in D$ a binary label

signifying if the sentence is or is not used as evidence. .Our output $\hat{y}_i$ is a binary logit output. We therefore seek to fine-tune our model on a binary task that attempts to predict the binary label $y \in D$ where our output is converted to binary through the sigmoid activation function $sigma$ with threshold $\sigma(\hat{y}_i) > 0.5$.

Definition
Formally we have our sentence selection model $f = (c, s) \mapsto \hat{y}_i$, and defined dataset D which is split into batches B of size N. We applied binary cross entropy loss as our loss function $\mathcal{L}$ to our model. Precisely we define the loss $\mathcal{L}$ as:

$$-\frac{1}{\mathcal{N}} \sum_{i=1}^{\mathcal{N}} y_i \cdot log(\sigma(\hat{y}_i)) + (1 - y_i) \cdot (1 - \sigma(\hat{y}_i))$$

Where $\hat{y}_i$ is the i-th tensor for the output of applying claim, sentence pair $\langle c_i, s_i \rangle$ to model function $f = (c_i, s_i) \mapsto \hat{y}_i$. y_i is the corresponding binary target label.

Selecting Top-K sentences
We formulate selecting the top-k sentences for a single row i in our dataset D of length $\mathcal{N}$ as follows:

$$Top_{-K}\ Sentences = \arg \max_{i \in \mathcal{N}; J \leftarrow min(K, \mathcal{N})} (\sigma(\hat{y}_i))$$

Where we generate a new set of elements E of size J by iterating over all documents in our document set D_i and appending $\langle c, s, k, t \rangle$ to the new set E, where k is the index of the current sentence $s_k \in d_j \in D_i$ and t is the document title of $d_j \in D_i$.

Each tokenised pair $\langle s; c \rangle$ is fed through our model $f = (c, s) \mapsto \hat{y}_i$, the argmax function returns J values where $J \leftarrow min(K, \mathcal{N})$ of the indices q in set E.

4.4 Local Interpretable Model-Agnostic Explanations

Task
We want to find a model that can locally approximate our claim verification model $c \mapsto f(c)$ for our selected claim c. We define an explanation $g \in G$ where G is a class of simple, interpretable models such as a decision tree or linear model. To find our explanation we must minimise our loss function $\mathcal{L}$ with an applied model g complexity penalty. The loss function $\mathcal{L}$ uses locally weighted square loss, which is defined further in this report.

Generating Explanations Definition
Formally we define this loss function as $\mathcal{L}(\hat{f}_d, g, v_c(z))$ where f is our classification model to be explained, $\hat{f}_d(c)$ defines the binary probability of $f(c)$ output belonging to class d. $v_c(z)$ is our proximity measure between instance z and claim c. Our loss measures

how unfaithful model g is at approximating our model f in the locality defined v_c. We introduce their measure of model complexity $\Omega(g)$ which is a factor dependent upon the complexity of the model $g \in G$. This can be used as a complexity penalty in computing our explanation. Formally computing an explanation is obtained through the following:

$$\hat{g}_d = \arg\min_{g \in G} \Big(\mathcal{L}\{\hat{f}_d, g, v_c(z)\} + \Omega(g) \Big)$$

Where $\hat{g}_d$ is the resultant explanation for class d in our multi-class claim classification task for our claim c.

Loss Function Definition
Our loss function $\mathcal{L}$ is the locally weighted square loss, first we must define our proximity measure $v_c(z) = exp(\frac{-cos(c,z)^2}{\sigma^2})$ which is an exponential kernel defined by the cosine distance. The cosine distance uses the trained embeddings of each of the models $g \in G$, we also let G be a class of linear models, such that $g(z') = w_g \cdot z'$. We therefore define the loss function:

$$\mathcal{L}(\hat{f}_d, g, v_c(z)) = \sum_{z,z' \in \mathcal{Z}} v_c(z)(f(z) - g(z'))^2$$

@misccommission$_2$020, $title = WhitePaperonArtificialIntelligence - EuropeanCommission, url = https://ec.europa.eu/info/sites/default/files/commission - white - paper - artificial - intellige EuropeanCommission, author = Commission, European, year = 2020$

4.5 KnowledgeEditor

The task of the KnowledgeEditor is to learn the parameters z of our base model $x \rightarrow f(c; z)$ such that we can fix our model architecture and for a set of data $\langle c, y, a \rangle \in D$ where c is a given claim, y the prediction output from our model $f(c; z)$ and a is the alternative prediction we seek to condition our model to predict instead of y. We use the KnowledgeEditor hyper-network to learn the parameters of our model z such that it finds alternative parameters z' for our model x, so that $f(c; z')$ predicts the alternative condition a instead of y. The most important aspect of retaining knowledge in our model is that for all other input, prediction pairs $\langle c', y' \rangle$ remain unchanged using the updated model $f(c'; z')$. We also wish that all semantically equivalent claims $c*$ for a given claim c are also predicted the alternative value a.

4.5.1 Semantically Equivalent Claims

For conditioned claims our model x must also predict their alternative a for any semantically equivalent claim $c*$, we generate our semantically equivalent statements not during

testing as this is computationally inefficient. We therefore pre-generate them using our dataset D, for each claim c we include a list of semantically equivalent statements $c*$. We utilise the *MarianMTModel* (Junczys-Dowmunt et al., 2018) from the *HuggingFace* (Wolf et al., 2019) library to perform translations of our claim c. The *opus-mt-en-de* and *opus-mt-de-en* pre-trained translation models perform translation from English to German and from German to English respectively without the need for fine-tuning. By performing many translations and backtranslations between German and English it enables us to generate a new claim semantically and structurally similar to our original claim c. Performing many backtranslations creates a list of semantically equivalent statements to our original claim, we then feed each of these new statements through our base model $f(c*, z)$ to filter out claims that do not result in the same prediction as our original claim c as we only intend to condition from one class to another. This forms the augmented dataset containing $\langle c, c*, y, a\rangle \in D*$ used in training our model.

4.5.2 Hyper-Network Architecture

We seek to alter the hyper-network to perform edits and conditioning for multi-classification, we plan to explore and evaluate the performance of performing multiple edits that fix incorrectly predicted claims. Further, we seek to explain what knowledge the updated model uses to generate these predictions through Local Interpretable Model-agnostic Explanations (LIME).

The Hyper-Network is a model that predicts our updated parameters z' to modify the output of our model x for a claim c with prediction y given an alternative condition a. Since we wish to edit the parameters for a given claim and retain the predictions of all other claims this is seen as a constrained optimisation task. This task is defined as minimising the loss $\mathcal{L}(z'; c; a)$ for alternative target a. We wish to preserve the remainder of the knowledge constraining the updated parameters z' such that our model predictions remain the same for $c' \in \mathcal{O}^c$.

Our model intends to predict the shift $\mathcal{Z}$ defined as $z' - z$ such that $z' = z + \mathcal{Z}$. We use a bi-LSTM model which takes as input an encoded set $\langle c, y, a\rangle$. The Bi-LSTM model's hidden states are then fed into an AllenNLP FeedForward module. The final output of the FeedForward module is a single vector *condition* variable. This condition variable is used to predict the shift for our model parameters, for each parameter from our base model we generate a parameter to conditioner mapping that means as our base model scales the Hyper-Network scales linearly.

Cao et al. (2021) also utilised a margin to constrain the amount we update our model to avoid exploding parameter values, we chose to retain the margin hyperparameter and the *margin annealing* technique to aid convergence during training. This process selects some initial value and anneals it throughout training; they base the annealing amount on the validation performance. When the model manages to alter 90%+ of the predictions

we multiply the margin by 0.8. This procedure was used to prevent diverging whilst tightening the constraint.

Chapter 5

Experimental Results

The main aim of our project was to further the system developed by Lee et al. (2020) by introducing components to explain the output on the classification task. We will therefore first evaluate our claim verification models against theirs before evaluating our explanation components.

We aim to evaluate the claim verification and retrieval systems by their performance on the FEVER shared-task, we further evaluate our retrieval systems, KnowledgeEditor and LIME in their ability to generate explanations for the output of our claim verification system. We finish our evaluation by critiquing KnowledgeEditor's ability to condition alternative predictions whilst retaining previous knowledge, in order to evaluate its ability to improve our claim verification systems performance. We begin this chapter by describing the dataset used in our experiments.

5.1 Dataset

Split	SUPPORTED	REFUTED	NOT ENOUGH INFO
Training	80,035	29,775	35,639
Development	3,333	3,333	3,333
Test	3,333	3,333	3,333

Table 5.1: Dataset split and sizes for each class in the FEVER shared-task.

The FEVER shared-task is split into training, development and test sets, each sets class occurrence is listed in Table 5.1. Figure 1.1 shows an example claim from the labelled test set, it includes the evidential sentences of the full shared-task.

The Wikipedia dump contains over 5.4 million articles, each article contains a set of sentences, the Wikipedia article contains 27,121 sentences (a table that converted poorly to sentences), the Wikipedia article set contains 8 sentences on average. Searching through a set of 5 million articles for each piece of evidence is computationally inefficient, we therefore processed the Wikipedia set into a dictionary object to enable O(1) search

Split	No. Instances	Evidence Count
Training	145,449	311,431
Dev	9,999	18,999
Test	9,999	18,567

Table 5.2: Number of instances and evidence for FEVER shared-task split into separate datasets.

efficiency, where the key is our articles title and the value is a list of sentences to use as evidence. We use the library unidecode which maps non-ASCII characters to a safely decoded ASCII string, this fixes issues caused where the encoding does not match between the Wikipedia dump and FEVER dataset for foreign entity names.

5.2 FEVER shared-task

5.2.1 Evaluation Metrics

For evaluating on the FEVER shared-task we use the shared metric across all FEVER systems which use the FEVER scorer metric. This metric combines the performance of the retrieval systems with the output of the claim predictions. We make use of the library "fever-scorer" produced by Thorne et al. (2018a) to calculate a *FEVER score* for our entire systems performance. The FEVER score is the *strict score/number of instances*, where the strict score is the number of instances that have a correct label prediction and correctly predicts the full set of evidence.

5.2.2 Claim Verification

Baselines

Our system is an extensions of work by Lee et al. (2020) which did not implement retrieval systems, we will therefore compare our two claim verification models to the performance of his three original baseline systems, *i)* BERT fine-tuned on all model layers, *ii)* BERT fine-tuned only on top classification layer, *iii)* BERT as knowledge base, their model that exploits the BERT masked language modelling task to create "evidence" that is used by their textual entailment model before being fed into a multi-layer perceptron. Details of model *iii* are found in figure 5.1.

Hyperparameters:

We train the base model in batches of 32 with the Adam optimiser (Kingma, Ba, 2014), setting the learning rate at 2e-5 with 1e-5 weight decay. We perform the training sequence over 15 epochs and use the checkpoint functionality within PyTorch (Paszke et al., 2019) to save the model with the highest validation across the training cycle. The validation accuracy is calculated after each epoch. We use 500 warm up steps and set the number of workers to 16 which was the most our system could handle in memory. We calculate

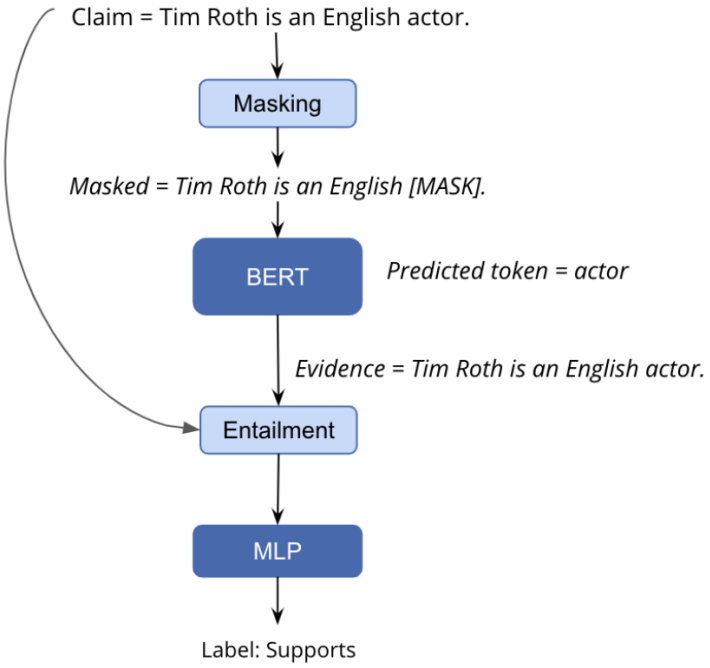

Figure 5.1: End-to-end entailment model (Lee et al., 2020)

the loss using *binary cross entropy loss with logits* minimised by an entropy smoothing factor. We use these settings for both BERT and RoBERTa models.

Results:

Model	Label Accuracy
BERT *Freeze*	0.38
BERT *Fine-Tuned*	0.57
BERT as *Knowledge Base*	0.49
BERT *Fine-Tuned* (Our System)	<u>0.58</u>
RoBERTa *Fine-Tuned* (Our System)	**0.63**

Table 5.3: Performance comparison between our models and work by (Lee et al., 2020) using label accuracy. **Bold** indicates the highest performing system, <u>underline</u> indicates second highest performer.

The results of our models are reported in Table 5.3. We observe that both of our proposed models outperform the work of Lee et al. (2020) on all their final models. These results suggest that larger language models are a better approach for the task of fact-checking as they have more capacity to encode knowledge. We can agree with our results as we see our model *BERT Fine-Tuned* performs very similarly to their system, suggesting the 1% improvement could be as a result of our altered loss function that penalises randomness in our system.

We found the performance increase in RoBERTa was a larger increase than expected over the performance of BERT, this further prompts the need to explore the possible results that can be achieved through even larger language models such as RoBERTa-large, XLNet (Yang et al., 2019) or T5 (Raffel et al., 2019) to find where we reach the

limits of their performance. T5 and XLNet would be particularly interesting as both differ in their model architecture to BERT and RoBERTa's encoder only stack. T5 is an encoder-decoder model which can have up to 11 billion parameters and XLNet a decoder only model with up to 340 million parameters.

5.2.3 Document Retrieval

Baselines

Our document retrieval component uses GENRE, a new approach that uses BART for autoregressive entity-linking. We compare our results to the SoTA on the original shared-task by UKP-Athene as it remains relevant since many current SoTA systems such as KGAT (Liu et al., 2019c) and ProoVer (Krishna et al., 2021) still use their retrieval system. We also compare our results to UNC-NLP as their approach similarly uses a neural network (NSMN) to rank the documents against the claim.

Implementation Details:

We implemented GENRE for document retrieval, implementing the library from GitHub requires creating a Python 3.7 environment and installing the correct dependencies. Some issues were found with FairSeq (Ott et al., 2019) installation due to compatibility with setuptools that can be corrected through downgrading the current setuptools version to <= 50.0.0.. We created a class named RetrieveDocument to handle pre-processing, model sampling and post-processing of the output data.

Pre-processing in previous versions of RetrieveDocument generated the wikipedia dictionary of document entity names to article sentences. This proved to be inefficient as GENRE can output characters outside of utf-8 encoding which would result in similar text not being encoded equivalently. One method to avoid this is utilising the unidecode library which standardises the wikipedia text with the GENRE document outputs. It does this by converting non-ASCII characters to ASCII equivalent encodings.

Model Sampling in the un-tuned RetrieveDocument class only used a beam of 5 documents for extraction and used a pre-trained Fairseq wikipage retrieval model and the pre-generated Wikipedia trie that was built to constrain the models output to a finite set of documents. They make use of both a normal trie and a MARISA-trie which is much more memory efficient than the standard dictionary lookup trie. We experimented with both trie objects and found the former to perform the constraint function much faster.

One issue discovered with the beam size was that GENRE's output favours entity identification over relevancy which could affect the performance of downstream tasks. As a result larger claims that contain multiple entities produce less useful entity rankings over more semantically similar documents. An example of this is the claim "The actor Johnny Depp starred in the movie Pirates of the Caribbean". GENRE's output maximises the number of entities recognised and not the prediction score generated by the model. The

generated output would be {[Actor]: 0.3, [Johnny Depp]: 0.9, [Pirates of the Caribbean (film series)]: 0.95, [Movies]: 0.3, [Pirates]: 0.5}, this maximises the number of entities recognised within the claim but not the ranking of all entities combined, by expanding our beam we find entities such as {[Pirates of the Caribbean]: 0.9} which contain much more richly related information and have a higher ranking score than *Actor* and *Movies*. This is why we expanded the beam-size of the model to 7 to maximise the amount of input for our sentence selection model.

Post-processing was required as the FEVER shared-task opts to replace punctuation with string literals i.e *"(Hello):(World)"* is replaced with *"-LRB-Hello-RRB–COLON–LRB-World-RRB-"*. This is handled in post-processing converting GENRE output to FEVER formatting as well as replacing whitespace with udnerscore characters as used in the FEVER dataset.

Custom Trie Constraints Using the base GENRE system for document retrieval resulted in low recall in comparison to other entity-linking systems. The major factor affecting performance is the model outputting documents not found in the Wikipedia dataset supplied for the FEVER shared-task. This is due to their trie included in the system being created on an updated version of the Wikipedia dump which contains over 6.5 million documents. We therefore deemed it necessary to experiment with generating our own trie object from the 5.2 million article Wikipedia dump used in the original task.

It was initially difficult to build the trie as there exists no documentation within their repository, this was a problem as using the Trie class within GENRE will not result in a working object. The original author utilises a token *[2]* to indicate the root of the trie to the Beam Search constraint. The process of generating the trie was both memory and computationally expensive on our personal machine, taking over 24 hours to generate the object. Once generated we used the library pickle to convert the trie to a importable *.pkl* file.

Results

Model (k=5)	Oracle Accuracy
UNC-NLP (KM + Pageview + dNSMN)	92.42
UKP-Athene (Entity Linking)	**93.30**
GENRE w/o Constraints	90.97
GENRE /w Constraints	<u>92.90</u>

Table 5.4: Oracle Accuracy scores for range of Document Retrieval models on Development Set where number of documents, k=5.

Our results of our document selection model are shown in Table 5.4, we can see GENRE outperforms UNC-NLP's Neural Semantic Matching Network on the task but underperforms when compared to the approach taken by UKP-Athene. Through evaluating our incorrect instances we viewed that the model struggled with instances that use parenthet-

ical disambiguation where article titles overlap so they use additional context to describe the entity. One such incorrect prediction was for the claim *"Scandal refuses to play rock music."* which the model predicted as evidence *"Scandal (TV Series)"* instead of *"Scandal (American band)"*. The results also show the increase in performance when applying custom trie constraints to our system as it removed generating documents outside the Wikipedia dump used in the FEVER shared-task.

Error Analysis

Claim: Arjit Singh is Indian.
Correct Evidence: [Arijit_Singh]
Predicted Evidence: [Arjan_Singh, Arjit_Gupta, Indian_people, Arjan, Arjen]

Claim: 57 elevators reside in the Burj Kahlifa.
Correct Evidence: [Burj_Khalifa]
Predicted Evidence: [Dubai_Mixed-Use_Towers, Almas_Tower, Dubai_Metro, Burj_Al_Arab, Burj_Al_Alam]

Claim: Gaius Julius Caesar was the son of Gaius Julius Caesar.
Correct Evidence: [Gaius_Julius_Caesar (proconsul)]
Predicted Evidence: [Gaius_Julius_Caesar, Gaius_Julius_Caesar (Rome_character), Julius_Caesar, Gaius_Caesar, Roman_emperor]

Figure 5.2: Errors in Sentence Selection model GENRE where model failed to produce single correct document.

Figure 5.2 shows some example errors in our GENRE output for document retrieval. The first two errors are quite unique as they show spelling errors in the entity names, UKP-Athene's system correctly predicts these entities where as GENRE struggled. This is most likely due to the techniques used in GENRE to constrain the search space resulting in incorrect spellings to be ignored as not existing by the trie object. This results in the second case where the model attempts to predict objects with 'Burj' in the title as it treats that as a seperate entity to the whole 'Burj Kahlifa' since they do not calculate character level similarity for entity names.

The last error is the result of the model predicting multiple entity names within a single entity such as *Julius_Caesar* and *Gaius_Caesar*. UKP-Athene uses a candidate article search that prefers wikipedia articles that match closest to the entire entity name once parenthesis have been stemmed which prefers longer entity names removing sub-name entities.

These issues could be attended to in GENRE by adjusting the sequence to sequence model training to predict incorrectly spelt sequences through augmenting the input by randomly shifting or removing characters. This would be useful in a real-world setting such as on social media where spelling errors are most common in the informal setting.

The model could also be re-trained to prefer longer sequences that do not exceed the length of the entity (once parenthesis are stemmed) and apply a semantic similarity measure between the parenthetical text and the claim to better identify these ambiguities.

5.2.4 Sentence Selection

Baselines

We treated the sentence selection task as a supervised classification problem similar to other SoTA models on the original FEVER shared-task. We therefore evaluate our models against the work produced by UNC-NLP and UKP-Athene.

We also intended to evaluate the performance increase of treating this task as a binary supervised classification task over a sentence similarity task by evaluating our own embedding baseline systems that ranks cosine similarity between claim and evidence:

GloVe-wiki-gigaword-300
Custom Trained FastText Model
Combined Embeddings

Our custom FastText model is pre-trained on the Wikipedia dump for the FEVER shared-task.

Thresholding

Method	Precision	Recall	F1	Maximal Accuracy
Without Thresholding				
Custom Trained FastText	0.13	0.55	0.21	0.62
GLoVe-300	0.13	0.55	0.21	0.62
Combined Embeddings	0.15	0.61	0.24	0.68
BERT	0.15	0.64	0.24	0.72
RoBERTa	**0.16**	**0.70**	**0.26**	**0.78**
With Thresholding				
Custom Trained FastText	**0.25**	0.71	**0.37**	0.79
GLoVe-300	0.19	0.73	0.31	0.82
Combined Embeddings	0.20	0.74	0.31	0.83
BERT	0.21	0.75	0.33	0.84
RoBERTa	0.22	**0.83**	0.34	**0.93**

Table 5.5: Full system scores on the FEVER shared-task test set.

We experimented with thresholding both sentence and document probabilities, thresholding sentence probabilities limits the amount of evidence for scoring which has a greater effect on the precision and F1 score of the final output. Document thresholding reduces the rank of ⟨ claim, sentence ⟩ pair for less relevant documents. We performed a beam search to identify which document based weights had the largest increase in validation performance on our overall results. These results are reported in Table 5.6.

When analysing the top 5 documents selected for our model we found that the last 2 GENRE extracted documents for each claim only accounted for 0.2% of the possible evidential sentences. Thresholding enables us to reduce the probability of sentences from these documents being selected over ones of more relevancy.

Results

Model	Precision	Recall	F1	Sentence Recall
UNC-NLP	**0.42**	0.71	**0.53**	0.7682
Athene UKP Tu Darmstadt	0.24	**0.85**	0.37	**0.9086**
Custom Trained FastText	0.25	0.71	0.37	0.7643
GLoVe-300	0.19	0.73	0.31	0.7858
Combined Embeddings	0.20	0.74	0.31	0.7966
BERT	0.21	0.75	0.33	0.8073
RoBERTa	0.22	0.83	0.34	0.8934

Table 5.6: Full retrieval system scores on the FEVER shared task, **Bold** indicates the highest performing model of that metric, underline indicates the second highest performing model.

Table 5.6 shows the final output of our retrieval systems compared against our baseline systems. The most important metric we wish to focus on is recall, as we want our system to perform well in retrieving all possible evidence for our claims, it is also preferred as the strict score metric used by the FEVER scorer measures only fully complete evidence as strictly correct. We can see our RoBERTa and BERT models perform second and third to UKP-Athene's ESIM model. The outputs in table 5.6 reflect the precision, recall and F1 of the models given the documents retrieved in the previous stage. We extract the sentence recall by calculating the full retrieval components recall divided by the document retrieval components accuracy.

What we can see from our results is that our RoBERTa sentence retrieval model performed comparably to UKP-Athene's ESIM model, we notice a similar performance increase from the BERT to RoBERTa model that prompts the need for further evaluation into larger language models for sentence selection. These results show that our retrieval systems are comparable to SoTA in the original FEVER shared-task which should allow us to produce evidence with over 80% recall to explain the output of our claim verification model.

5.3 KnowledgeEditor

Baseline

We will compare our multi-label KnowledgeEditor implementation to the original binary implementation by Cao et al. (2021), we also compare our system against fine-tuning all layers of our classification model to fix incorrect predictions. To do this we follow the work of Sinitsin et al. (2020) who employ RMSProp, we fine-tune until our output changes to our desired label or we reach 100 gradient steps with a learning rate of 1e-5.

Results

From the results in Table 5.7 we can see that KnowledgeEditor succeeds in altering our claims to their alternative predictions whilst performing greater at retaining accuracy and altering equivalent claims than fine-tuning our models. These results demonstrates

Method	Equivalent Acc. ↑	Retained Acc. ↑	Success Rate ↑
BERT - Fine-Tune (all-layers)	95.70	86.00	100.0
RoBERTa - Fine-Tune (all-layers)	88.30	74.80	100.0
BERT - KnowledgeEditor	96.90	92.60	100.0
RoBERTa - KnowledgeEditor	89.70	97.60	100.0

Table 5.7: Oracle Accuracy scores for range of Document Retrieval models on Development Set where number of documents, k=5.

the performance of the hyper-network in altering our claims such that it has learnt the parameters of our model and will allow us to express the layers that require the largest shifts to change the encoded knowledge which can give us an insight into how our language models function.

We also evaluated the viability of using KnowledgeEditor to perform multiple concurrent edits to fix incorrect predictions in our development set to evaluate how it affects our claim verification performance on the test set.

Method	FEVER - label accuracy
K=1	
BERT + KnowledgeEditor	0.54
RoBERTa + KnowledgeEditor	0.61
K=10	
BERT - KnowledgeEditor	0.45
RoBERTa - KnowledgeEditor	0.57

Table 5.8: Oracle Accuracy scores for range of Document Retrieval models on Development Set where number of documents, k=5.

Hyper-Network Parameters Evaluation

The hyper-network enables us to generate as output the shift required in our models parameters in order to fix our model to an alternative prediction, we measure these shifts on incorrect model predictions and plot their normalised magnitude:

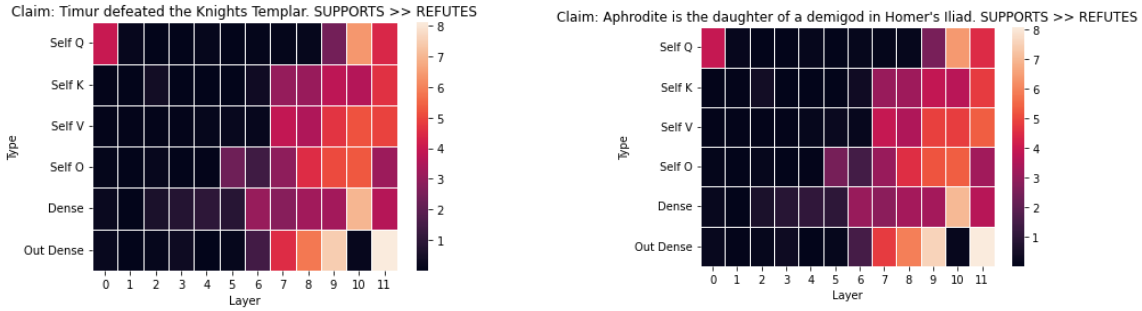

Figure 5.3: plots of BERT parameter shift from class SUPPORTS to REFUTES

Figure 5.3 and Figure 5.4 allow us to see the difference in shifting from classes *SUPPORTS* to *REFUTES* and *NEI* to *REFUTES*, we found for each unique pair of classes differ in the area of the model architecture they affect, this is interesting in how it affects the early layers of the model.

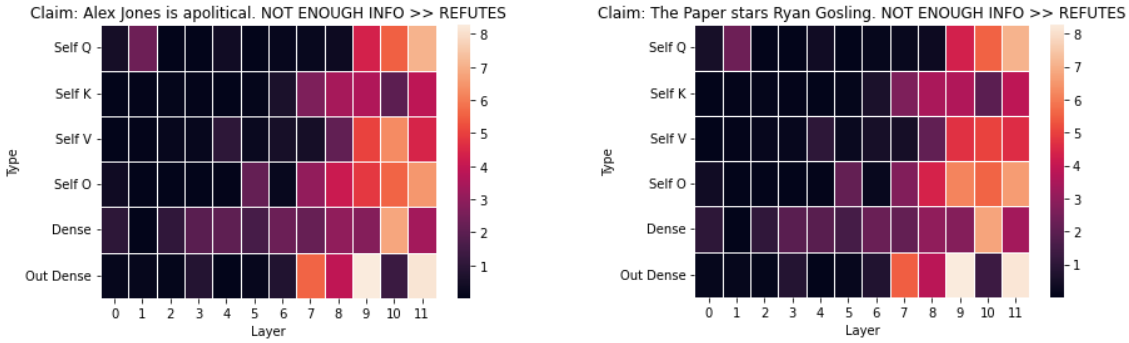

Figure 5.4: plots of BERT parameter shift from class NEI to REFUTES

Across all models the highest layers experience the most shifts, this is expected as the final layers of BERT are the most task-specific as noted by Liu et al. (2019a) and explains why in fine-tuning we also find the final layers adjust the most (Kovaleva et al., 2019). Clark et al. (2019) identified that attention heads often attend to "special tokens", early heads attending to the classification token, middle heads attending [SEP] tokens and deep heads attending to periods and commas. Large parameter changes took place for $SUPPORTS$ in layer-0 for the query mechanism of the self-attention heads and for $REFUTES$ it solely makes changes to layer-1 of the query mechanism in our early heads. Classes that use NEI make no changes to either but instead have a greater affect on earlier layers in our intermediate dense layer.

Geva et al. (2020) explored the role of the feed-forward layers in transformer architecture and discovered that they operate as key-value memories where each key correlates with textual patterns in the training examples, where lower layers capture shallow patterns whilst upper layers learn more semantic patterns.

Shallow patterns relate to the basic syntactic relationship between words and n-grams, for our work it more likely relates to specific entity names and their relations such as what "Alex Jones" means to "Politican".

Semantic patterns relates to some topic, either describing a specific category such as "TV shows" or a semantic language pattern such as describing "part-of" relations in text. When we normalise the values to produce our heatmap it no longer describes if a change is positive or negative to the weights, when we analysed the weight shift from NEI to $SUPPORTS$ or $REFUTES$ relative to the original parameters we found that in earlier layers it tended the parameters away from 0 whilst altering the parameters of our models to NEI it tends towards 0. From this we begin to understand how the model uses the intermediate layer to encode information for named entities in the early layers, when learning new knowledge it focuses on increasing the weights of these layers to encode patterns relating to these entities and decreases the weights when removing knowledge.

One interesting thing we noted was that the BERT hyper-network does not use the final fully-connected layer at layer 10, Alammar (2020) researched the best contextualised embeddings from BERT based upon the output of each encoder layer, they found that

the best performing layer to use to generate contextualised encoding information for a single token is from the penultimate layer, this exceeded performance of summing all 12 encoder layers together. They identified that as we get deeper and deeper into our model we encode more and more context information but the last layer word embeddings begin to get BERT information about task specific training. We can therefore assume as KnowledgeEditor attempts to retain information, adjusting rich contextual features would impact many alternative but similar sequences we do not wish to alter the prediction for. Fine-tuning our model changed all layers evenly and therefore resulted in greater loss of previously encoded rich context for our word embeddings.

This opens up an interesting area of future research that could explore freezing specific areas of our language models that relate to knowledge encoding in order to retain as much information whilst taking advantage of fine-tuning's ability to encode many batches of new information quickly.

5.4 Explaining Model Predictions

In this section we will finally combine all our components to complete our system. We will blind analyse a randomly selected predicted claim from the FEVER shared-task test data using a combination of our evidence retrieval systems, LIME system and our hyper-network trained on our claim verification model. We intend to use these systems to explain and evaluate if we can trust the prediction given to us from the claim verification model.

5.4.1 Claim Analysis

Claim: 'Birmingham is a large town.'
Prediction: 'SUPPORTS'

Evidence Retrieved:
Birmingham
- Birmingham is a city and metropolitan borough in the West Midlands, England. West Midlands (county) Birmingham Airport city city status in the United Kingdom metropolitan borough metropolitan borough England

History_of_Birmingham
- The last 200 years have seen Birmingham rise from market town into the fastest-growing city of the 19th century, spurred on by a combination of civic investment, scientific achievement, commercial innovation and by a steady influx of migrant workers into its suburbs.

History_of_Birmingham
- The history of Birmingham in England spans 1400 years of growth, during which time it has evolved from a small 7th century Anglo Saxon hamlet on the edge of the Forest of Arden at the fringe of early Mercia to become a major city through a combination of immigration, innovation and civic pride that helped to bring about major social and economic reforms and to create the Industrial Revolution , inspiring the growth of similar cities across the world.

Birmingham
- A medium-sized market town in the medieval period, Birmingham grew to international prominence in the 18th century at the heart of the Midlands Enlightenment and subsequent Industrial Revolution, which saw the town at the forefront of worldwide advances in science, technology, and economic development, producing a series of innovations that laid many of the foundations of modern industrial society.

List_of_cities_and_town_in_Alabama
- The largest municipality by population is Birmingham with 212,237 residents while the smallest by population is McMullen with 10 people. Birmingham, Alabama

Table 5.9: System outputs for claim: 'Birmingham is a large town.'

Table 5.9 shows the prediction of our claim verification model and evidence retrieved through our retrieval components for the claim "Birmingham is a large town.". Without the label and evidence from the labelled test set we aim to visualise if we can trust this prediction and can we understand how our model came to this conclusion manually.

The higher up our evidence retrieval list an item is the more weight it has in relation to our claim. All extracted evidence items do not explicitly state that Birmingham is a large town but instead note the growth of Birmingham from a town into a city which by definition is 'a large town'. We can see that Birmingham has been a rapidly growing town into a city within the United Kingdom, this added information helps us trust the output

of our claim verification system as we can manually evaluate the prediction through our evidence retrieval system. We can also see that the evidence extends beyond the original claim's evidence selection to include Birmingham, Alabama and demonstrate that it is the largest municipality by population demonstrating further evidence to support the prediction output.

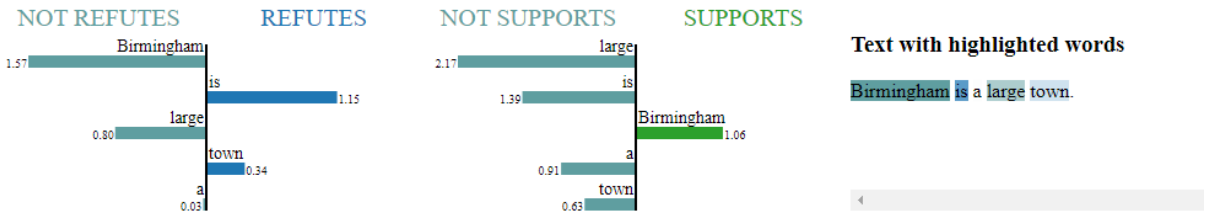

Figure 5.5: LIME output of sentence pair claim: "Birmingham is a large town."

Figure 5.5 visualises the importance of each token in our input to our claim verification model. It highlights that the model used the information 'Birmingham' and 'Large' to reject the 'REFUTES' class as a possible class which enforced the prediction 'SUPPORTS'. We can evaluate that our claim verification model is using the entity names in making its predictions. Whilst stop words such as 'is' are still being used by the model in making its prediction, they are not affecting the output. When this visualisation of our input sequence is combined with the information retrieved from the evidence retrieval components, it can help build our confidence that the claim verification model is using the correct features in generating its predictions.

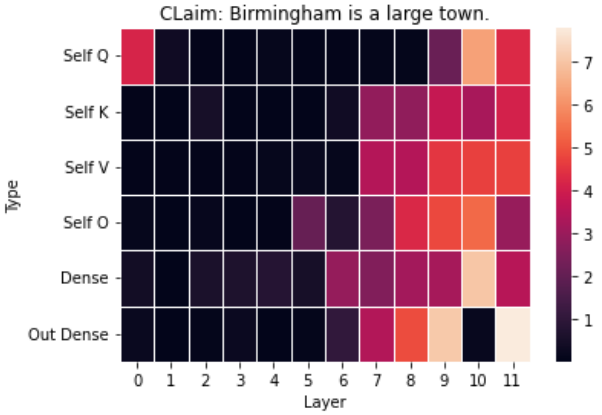

Figure 5.6: Claim Verification model parameter update for claim conditioned on its prediction class: *SUPPORTS*.

Figure 5.6 demonstrates that the claim verification model predicted its output for our claim using mainly its task specific parameters in the later layers of our model and the penultimate layers which contains the encodings for the rich contextual embeddings of the model. We prefer the model to use this information in generating predictions as it shows the model is not using a single entity or feature in generating its output but placing more value on the context of the feature, which LIME showed it places high

weighting on the entity 'Birmingham', this enables further trust in the output of our claim verification model as we can demonstrate from these systems that it focused on our named entities with context in generating its output, and we can further trust the output as we can manually verify the evidence through our isolate document retrieval and evidence selection components.

Chapter 6

Conclusions and Future Work

6.1 Conclusions

When introducing our project we defined a list of desiderata that we can use to consider if our project is a success. We can do this by evaluating our systems results against each desideratum.

1. We intended to improve the claim verification model by Lee et al. (2020). We have achieved this aim, through improving the goal of our loss function $\mathcal{L}$ and experimenting with larger language models such as RoBERTa with better pretraining, we improved the claim verification models accuracy by 6%. Prompting research into the performance of even larger language models on this task to identify if knowledge is linearly correlated to pretraining and parameter size.

2. Our system extends KnowledgeEditor to learn the parameters of multiclass classification models whilst retaining greater equivalent accuracy and retained accuracy than fine-tuning the same models. We evaluated the viability of KnowledgeEditor as a method to fix correct knowledge and found after multiple edits the performance of our model begins to deteriorate with each contiguous edit.

3. In section 5.4 we combined all our systems for explanation to evaluate a single claim from our model. We demonstrated how our evidence retrieval components, LIME and our hyper-network gave us detailed insight into what information was used by our model to generate its prediction. It allowed us to manually explain and trust the language models output.

We conclude that our project furthered the work of Lee et al. (2020), improving performance and understanding of the model, transitioning their work to being implementable in real-world settings for fact-checking. It also prompted many areas of future research in isolated claim verification systems, editing language models and explaining their output.

6.2 Future Work

In this section we will discuss future work we would extend our components to experiment with if given more time,

Through our work we have seen the huge performance gains that are possible from transitioning from pretrained BERT to RoBERTa for claim verification. We highlighted the need for future research into applying models such as T5 and XLNet to the same task to identify if the performance increase is due to the optimises in data and pretraining tasks or if the number of parameters directly relates to the amount of knowledge that can be encoded.

GENRE for document retrieval performed close to current SoTA techniques with little tuning required. Future work would look into improving areas where it suffers such as parenthetical disambiguation by stemming titles and using a neural network to rank similarity between the claim and the parenthetical categories.

Current editing systems for language models can only condition one claim to another at a time, this results in an inability to batch edits and results in contiguous edits losing retained knowledge. Extending this project, we would look into performing contiguous class shifts on large batches of incorrect knowledge to avoid this effect.

Generating explanations using LIME can suffer as it is susceptible to *hindsight bias*, as the pruning of words in generating its output can be overly-aggressive and introduce noise to our output. Diffmask learns to mask subsets of the input while maintaining differentiability, it allows for producing outputs similar to LIME and heatmaps similar to those we produced through our hyper-network. In future work we would implement Diffmask and evaluate its performance against our two explanation components.

Any of these areas of research should produce interesting results and help further our work towards explainable automated fact-checking systems.

Bibliography

Aken Betty van, Winter Benjamin, Löser Alexander, Gers Felix A. How Does BERT Answer Questions? // Proceedings of the 28th ACM International Conference on Information and Knowledge Management. nov 2019.

Alammar Jay. The illustrated transformer. 2020.

Bhardwaj Rishabh, Majumder Navonil, Poria Soujanya. Investigating Gender Bias in BERT. 2020.

Bolukbasi Tolga, Pearce Adam, Yuan Ann, Coenen Andy, Reif Emily, Viégas Fernanda, Wattenberg Martin. An Interpretability Illusion for BERT. 2021.

Camacho-Collados Jose, Pilehvar Mohammad Taher. Embeddings in Natural Language Processing // Proceedings of the 28th International Conference on Computational Linguistics: Tutorial Abstracts. Barcelona, Spain (Online): International Committee for Computational Linguistics, XII 2020. 10–15.

Cao Nicola De, Aziz Wilker, Titov Ivan. Editing Factual Knowledge in Language Models. 2021.

Chen Danqi, Fisch Adam, Weston Jason, Bordes Antoine. Reading Wikipedia to Answer Open-Domain Questions // Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). Vancouver, Canada: Association for Computational Linguistics, VII 2017a. 1870–1879.

Chen Qian, Zhu Xiaodan, Ling Zhen-Hua, Wei Si, Jiang Hui, Inkpen Diana. Enhanced LSTM for Natural Language Inference // Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). 2017b.

Extract and Aggregate: A Novel Domain-Independent Approach to Factual Data Verification. // . 01 2019. 69–78.

Clark Kevin, Khandelwal Urvashi, Levy Omer, Manning Christopher D. What Does BERT Look At? An Analysis of BERT's Attention. 2019.

Commission European. White Paper on Artificial Intelligence - European Commission. 2020.

De Cao Nicola, Izacard Gautier, Riedel Sebastian, Petroni Fabio. Autoregressive Entity Retrieval. 2020.

Devlin Jacob, Chang Ming-Wei, Lee Kenton, Toutanova Kristina. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding // Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers). Minneapolis, Minnesota: Association for Computational Linguistics, VI 2019. 4171–4186.

Fellbaum Christiane. WordNet: An Electronic Lexical Database. 1998.

Gardner Matt, Grus Joel, Neumann Mark, Tafjord Oyvind, Dasigi Pradeep, Liu Nelson, Peters Matthew, Schmitz Michael, Zettlemoyer Luke. AllenNLP: A Deep Semantic Natural Language Processing Platform. 2018.

Geva Mor, Schuster Roei, Berant Jonathan, Levy Omer. Transformer Feed-Forward Layers Are Key-Value Memories. 2020.

Glass Alyssa, McGuinness Deborah L., Wolverton Michael. Toward Establishing Trust in Adaptive Agents // Proceedings of the 13th International Conference on Intelligent User Interfaces. New York, NY, USA: Association for Computing Machinery, 2008. 227–236. (IUI '08).

Gohel Prashant, Singh Priyanka, Mohanty Manoranjan. Explainable AI: current status and future directions. 2021.

Guo Zhijiang, Schlichtkrull Michael, Vlachos Andreas. A Survey on Automated Fact-Checking // Transactions of the Association for Computational Linguistics. 02 2022. 10. 178–206.

Hanselowski Andreas, Zhang Hao, Li Zile, Sorokin Daniil, Schiller Benjamin, Schulz Claudia, Gurevych Iryna. UKP-Athene: Multi-Sentence Textual Entailment for Claim Verification // Proceedings of the First Workshop on Fact Extraction and VERification (FEVER). Brussels, Belgium: Association for Computational Linguistics, XI 2018a. 103–108.

Hanselowski Andreas, Zhang Hao, Li Zile, Sorokin Daniil, Schiller Benjamin, Schulz Claudia, Gurevych Iryna. UKP-Athene: Multi-Sentence Textual Entailment for Claim Verification. 2018b.

Team SWEEPer: Joint Sentence Extraction and Fact Checking with Pointer Networks. // . 01 2018. 150–155.

Jacovi Alon, Goldberg Yoav. Towards Faithfully Interpretable NLP Systems: How Should We Define and Evaluate Faithfulness? // Proceedings of the 58th Annual Meeting of

the Association for Computational Linguistics. Online: Association for Computational Linguistics, VII 2020. 4198–4205.

Jang Joel, Ye Seonghyeon, Yang Sohee, Shin Joongbo, Han Janghoon, Kim Gyeonghun, Choi Stanley Jungkyu, Seo Minjoon. Towards Continual Knowledge Learning of Language Models. 2021.

Ji Heng, Grishman Ralph. Knowledge Base Population: Successful Approaches and Challenges // Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies. Portland, Oregon, USA: Association for Computational Linguistics, VI 2011. 1148–1158.

Jobanputra Mayank. Unsupervised Question Answering for Fact-Checking. 2019.

Joulin Armand, Grave Edouard, Bojanowski Piotr, Mikolov Tomas. Bag of Tricks for Efficient Text Classification. 2016.

Junczys-Dowmunt Marcin, Grundkiewicz Roman, Dwojak Tomasz, Hoang Hieu, Heafield Kenneth, Neckermann Tom, Seide Frank, Germann Ulrich, Aji Alham Fikri, Bogoychev Nikolay, Martins André F. T., Birch Alexandra. Marian: Fast Neural Machine Translation in C++ // Proceedings of ACL 2018, System Demonstrations. Melbourne, Australia: Association for Computational Linguistics, VII 2018. 116–121.

Kingma Diederik P., Ba Jimmy. Adam: A Method for Stochastic Optimization. 2014.

Kovaleva Olga, Romanov Alexey, Rogers Anna, Rumshisky Anna. Revealing the Dark Secrets of BERT // Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP). Hong Kong, China: Association for Computational Linguistics, XI 2019. 4365–4374.

Krishna Amrith, Riedel Sebastian, Vlachos Andreas. ProoFVer: Natural Logic Theorem Proving for Fact Verification. 2021.

Krueger David, Huang Chin-Wei, Islam Riashat, Turner Ryan, Lacoste Alexandre, Courville Aaron. Bayesian Hypernetworks. 2018.

Lee Nayeon, Li Belinda Z., Wang Sinong, Yih Wen-tau, Ma Hao, Khabsa Madian. Language Models as Fact Checkers? 2020.

Lewis Mike, Liu Yinhan, Goyal Naman, Ghazvininejad Marjan, Mohamed Abdelrahman, Levy Omer, Stoyanov Ves, Zettlemoyer Luke. BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension. 2019.

Liu Nelson F., Gardner Matt, Belinkov Yonatan, Peters Matthew E., Smith Noah A. Linguistic Knowledge and Transferability of Contextual Representations. 2019a.

Liu Yinhan, Ott Myle, Goyal Naman, Du Jingfei, Joshi Mandar, Chen Danqi, Levy Omer, Lewis Mike, Zettlemoyer Luke, Stoyanov Veselin. RoBERTa: A Robustly Optimized BERT Pretraining Approach. 2019b.

Liu Zhenghao, Xiong Chenyan, Sun Maosong, Liu Zhiyuan. Fine-grained Fact Verification with Kernel Graph Attention Network. 2019c.

Lundberg Scott, Lee Su-In. A Unified Approach to Interpreting Model Predictions. 2017.

Madsen Andreas, Reddy Siva, Chandar Sarath. Post-hoc Interpretability for Neural NLP: A Survey. 2022.

Malon Christopher. Team Papelo: Transformer Networks at FEVER // Proceedings of the First Workshop on Fact Extraction and VERification (FEVER). Brussels, Belgium: Association for Computational Linguistics, XI 2018. 109–113.

Menick Jacob, Trebacz Maja, Mikulik Vladimir, Aslanides John, Song Francis, Chadwick Martin, Glaese Mia, Young Susannah, Campbell-Gillingham Lucy, Irving Geoffrey, McAleese Nat. Teaching language models to support answers with verified quotes. 2022.

Mikolov Tomas, Chen Kai, Corrado Greg, Dean Jeffrey. Efficient Estimation of Word Representations in Vector Space. 2013.

Mitchell Eric, Lin Charles, Bosselut Antoine, Finn Chelsea, Manning Christopher D. Fast Model Editing at Scale. 2021.

Mosbach Marius, Andriushchenko Maksym, Klakow Dietrich. On the Stability of Fine-tuning BERT: Misconceptions, Explanations, and Strong Baselines. 2021.

Nie Yixin, Chen Haonan, Bansal Mohit. Combining Fact Extraction and Verification with Neural Semantic Matching Networks. 2018.

Ott Myle, Edunov Sergey, Baevski Alexei, Fan Angela, Gross Sam, Ng Nathan, Grangier David, Auli Michael. fairseq: A Fast, Extensible Toolkit for Sequence Modeling. 2019.

Paszke Adam, Gross Sam, Massa Francisco, Lerer Adam, Bradbury James, Chanan Gregory, Killeen Trevor, Lin Zeming, Gimelshein Natalia, Antiga Luca, Desmaison Alban, Kopf Andreas, Yang Edward, DeVito Zachary, Raison Martin, Tejani Alykhan, Chilamkurthy Sasank, Steiner Benoit, Fang Lu, Bai Junjie, Chintala Soumith. PyTorch: An Imperative Style, High-Performance Deep Learning Library // Advances in Neural Information Processing Systems 32. 2019. 8024–8035.

Pennington Jeffrey, Socher Richard, Manning Christopher. GloVe: Global Vectors for Word Representation // Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP). Doha, Qatar: Association for Computational Linguistics, X 2014. 1532–1543.

Peters Matthew E., Neumann Mark, Iyyer Mohit, Gardner Matt, Clark Christopher, Lee Kenton, Zettlemoyer Luke. Deep contextualized word representations. 2018.

Petroni Fabio, Piktus Aleksandra, Fan Angela, Lewis Patrick, Yazdani Majid, De Cao Nicola, Thorne James, Jernite Yacine, Karpukhin Vladimir, Maillard Jean, Plachouras Vassilis, Rocktäschel Tim, Riedel Sebastian. KILT: a Benchmark for Knowledge Intensive Language Tasks // Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies. Online: Association for Computational Linguistics, VI 2021. 2523–2544.

Petroni Fabio, Rocktäschel Tim, Lewis Patrick, Bakhtin Anton, Wu Yuxiang, Miller Alexander H., Riedel Sebastian. Language Models as Knowledge Bases? 2019.

Portelli Beatrice, Zhao Jason, Schuster Tal, Serra Giuseppe, Santus Enrico. Distilling the Evidence to Augment Fact Verification Models // Proceedings of the Third Workshop on Fact Extraction and VERification (FEVER). Online: Association for Computational Linguistics, VII 2020. 47–51.

Raffel Colin, Shazeer Noam, Roberts Adam, Lee Katherine, Narang Sharan, Matena Michael, Zhou Yanqi, Li Wei, Liu Peter J. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. 2019.

Ribeiro Marco Tulio, Singh Sameer, Guestrin Carlos. "Why Should I Trust You?": Explaining the Predictions of Any Classifier // Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016. 2016. 1135–1144.

Schuster Mike, Paliwal Kuldip. Bidirectional recurrent neural networks // Signal Processing, IEEE Transactions on. 12 1997. 45. 2673 – 2681.

Sharir Or, Peleg Barak, Shoham Yoav. The Cost of Training NLP Models: A Concise Overview. 2020.

Sinitsin Anton, Plokhotnyuk Vsevolod, Pyrkin Dmitriy, Popov Sergei, Babenko Artem. Editable Neural Networks. 2020.

Soleimani Amir, Monz Christof, Worring Marcel. BERT for Evidence Retrieval and Claim Verification. 2019.

Soleimani Amir, Monz Christof, Worring Marcel. BERT for Evidence Retrieval and Claim Verification // Advances in Information Retrieval. Cham: Springer International Publishing, 2020. 359–366.

Stammbach Dominik, Neumann Guenter. Team DOMLIN: Exploiting Evidence Enhancement for the FEVER Shared Task // Proceedings of the Second Workshop on Fact

Extraction and VERification (FEVER). Hong Kong, China: Association for Computational Linguistics, XI 2019. 105–109.

Sutskever Ilya, Vinyals Oriol, Le Quoc V. Sequence to Sequence Learning with Neural Networks. 2014.

Thorne James, Vlachos Andreas, Christodoulopoulos Christos, Mittal Arpit. FEVER: a Large-scale Dataset for Fact Extraction and VERification // Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers). New Orleans, Louisiana: Association for Computational Linguistics, VI 2018a. 809–819.

FEVER: a large-scale dataset for Fact Extraction and VERification (baseline system introduction). // . 03 2018b.

Thorne James, Vlachos Andreas, Cocarascu Oana, Christodoulopoulos Christos, Mittal Arpit. The Fact Extraction and VERification (FEVER) Shared Task // Proceedings of the First Workshop on Fact Extraction and VERification (FEVER). Brussels, Belgium: Association for Computational Linguistics, XI 2018c. 1–9.

Vaswani Ashish, Shazeer Noam, Parmar Niki, Uszkoreit Jakob, Jones Llion, Gomez Aidan N., Kaiser Lukasz, Polosukhin Illia. Attention Is All You Need. 2017.

Wang Alex, Singh Amanpreet, Michael Julian, Hill Felix, Levy Omer, Bowman Samuel R. GLUE: A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding. 2018.

Wang Xiang, He Xiangnan, Cao Yixin, Liu Meng, Chua Tat-Seng. KGAT // Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery &amp Data Mining. jul 2019.

Widyassari Adhika Pramita, Rustad Supriadi, Shidik Guruh Fajar, Noersasongko Edi, Syukur Abdul, Affandy Affandy, Setiadi De Rosal Ignatius Moses. Review of automatic text summarization techniques methods // Journal of King Saud University - Computer and Information Sciences. 2022. 34, 4. 1029–1046.

Wikipedia . Wikipedia Dump September 2017. 2017.

Wolf Thomas, Debut Lysandre, Sanh Victor, Chaumond Julien, Delangue Clement, Moi Anthony, Cistac Pierric, Rault Tim, Louf Rémi, Funtowicz Morgan, Davison Joe, Shleifer Sam, Platen Patrick von, Ma Clara, Jernite Yacine, Plu Julien, Xu Canwen, Scao Teven Le, Gugger Sylvain, Drame Mariama, Lhoest Quentin, Rush Alexander M. HuggingFace's Transformers: State-of-the-art Natural Language Processing. 2019.

Yang Zhilin, Dai Zihang, Yang Yiming, Carbonell Jaime, Salakhutdinov Ruslan, Le Quoc V. XLNet: Generalized Autoregressive Pretraining for Language Understanding. 2019.

Ye Deming, Lin Yankai, Du Jiaju, Liu Zhenghao, Li Peng, Sun Maosong, Liu Zhiyuan. Coreferential Reasoning Learning for Language Representation. 2020.

Yoneda Takuma, Mitchell Jeff, Welbl Johannes, Stenetorp Pontus, Riedel Sebastian. UCL Machine Reading Group: Four Factor Framework For Fact Finding (HexaF) // Proceedings of the First Workshop on Fact Extraction and VERification (FEVER). Brussels, Belgium: Association for Computational Linguistics, XI 2018. 97–102.

Zhong Wanjun, Xu Jingjing, Tang Duyu, Xu Zenan, Duan Nan, Zhou Ming, Wang Jiahai, Yin Jian. Reasoning Over Semantic-Level Graph for Fact Checking. 2019.

Zhou Jie, Han Xu, Yang Cheng, Liu Zhiyuan, Wang Lifeng, Li Changcheng, Sun Maosong. GEAR: Graph-based Evidence Aggregating and Reasoning for Fact Verification. 2019.

Zhu Chen, Rawat Ankit Singh, Zaheer Manzil, Bhojanapalli Srinadh, Li Daliang, Yu Felix, Kumar Sanjiv. Modifying Memories in Transformer Models. 2020.

Appendix A

Additional Figures

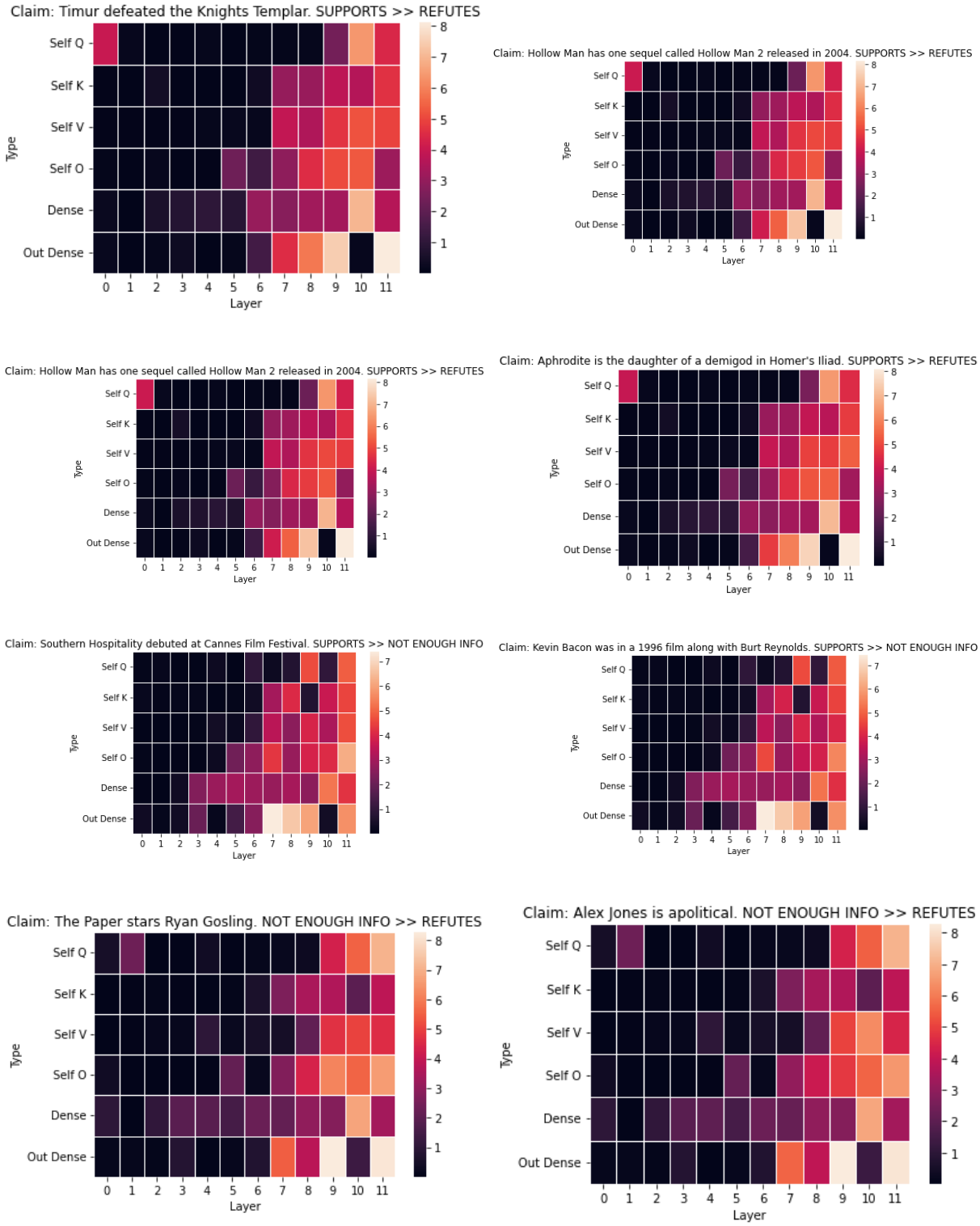

Figure A.1: plots of normalised parameter shifts from KnowledgeEditor on BERT architecture, Self = Attention Layers, Dense = Fully Connected layers