COMP90042 Project 2023
Automated Fact Checking for Climate Science Claims

Student ID : 981067

1 Introduction

In the age of fake news, reliable fact-checking methods are essential. Manual fact-checking is useful but time-consuming and can't keep up with today's information flow. Thus, automatic fact-checking system is needed to combat disinformation epidemic. This complicated activity involves gathering relevant evidence and assessing a claim's authenticity. This report used a two-step approach to create an automated fact checking system. First, we construct an evidence retrieval model by experimenting with BM25, the Dense Retrieval model, and an ensemble of the two to balance keyword-based search with semantic comprehension. After that, we build a claim classification model and test conventional machine learning classifiers like Gaussian Naive Bayes, Logistic Regression, SVMs, and Random Forest, as well as DistilBERT, a transformer-based model that excels in NLP tasks. Our system struggled in evidence retrieval and claim categorisation. Automated fact-checking requires sophisticated models that can understand complicated language contexts and semantic linkages. This report analyses these difficulties and recommends ways to improve automated fact-checking. Our results provide the groundwork for future study and the creation of more comprehensive and effective fact-checking systems to counteract disinformation.

2 Methodologies

2.1 Data Preprocessing

The development of an automated fact-checking system required extensive text preprocessing. Our preprocessing pipeline involves converted all text to lowercase to ensure recognition of identical phrases regardless of case, thereby promoting model consistency[1]. Additionally, non-alphanumeric characters were replaced with spaces, simplifying text, and facilitating text analysis [2]. To prevent model misinterpretations, we reduced excess whitespace to a single space [3]. We do not remove stopwords to enhance phrase match precision in retrieval models such as BM25 and DistilBERT because context can be maintained [4][5]. It also upheld compatibility with pretrained models trained on datasets containing stopwords, such as DistilBERT[6].

2.2 Evidence retrieval model

2.2.1 BM25

We considered BM25 for evidence retrieval due to its simplicity, efficacy, and pervasive application in information retrieval [9]. BM25 is a well-known probabilistic information retrieval model that considers term frequency, inverse document frequency, and normalisation of document length. It can retrieve sections of relevant evidence [9]. The "rank-bm25" package made BM25 implementation straightforward and effective. Using the "word_tokenize" function from the "nltk" package, we created a BM25Okapi model with tokenized evidence. We tokenized claim language and generated passage scores based on the model's ability to identify evidence. The three highest-scoring evidence passages were selected based on the average number of evidence passages in the train and development datasets. For parameter optimisation, we used the default values for k1 and b (1.22 and 0.75, respectively) because they have been empirically demonstrated to be effective in a variety of contexts [9]. The F-score is used to measure the performance of our BM25 experiment.

2.2.2 Dense retrieval

Since it manages semantic similarity between queries and documents, Dense Retrieval is optimal for evidence retrieval [10]. Unlike TF-IDF or BM25 [5], Dense Retrieval uses complex language models to comprehend text semantics and retrieve relevant documents even without exact word matches. Using Hugging Face's transformers library, we fine-tuned a pre-trained DistilBERT [6] model for sequence classification. The model separated input claim-evidence pairs into two distinct categories:

1.'SUPPORTS' / 'REFUTES'.

2.'NOT_ENOUGH_INFO'/'DISPUTED',

represented by the binary labels '1' and '0', respectively. Since our objective is to obtain the most relevant evidence passages given the claim, we undertake binary classification to determine whether a passage is relevant. Accordingly, we categorise pairings of claims and evidence as "RELEVANT" with a value of "1" or "IRRELEVANT" with a value of "0." Popular deep learning framework PyTorch supported our implementation. With the supplied training parameters, the 'Trainer' class of Hugging Face's transformers library was used for training. We trained the model for ten epochs with a batch size of eight, weight decay for regularisation, and a warm-up phase to stabilise the learning rate. To prevent overfitting, an early halting trigger terminated training if the model's performance on development data set did not improve over three consecutive evaluation steps. After training, the refined model generated dense embeddings for the evidence and claim data in the training and development sets. This involved applying the fine-tuned model to each text data and extracting the mean of the last hidden states as the embedding to capture semantic content of the texts [5]. Finally, we developed a retrieval function that retrieves the 'top_k' most similar evidence passages for a claim based on the cosine similarity of their embeddings. Based on the average number of evidence passages in the train and development datasets, we set top_k to three [11]. Cosine similarity is useful for high-dimensional spaces such as our DistilBERT embeddings. Using this technique, all development set evidence passages were retrieved. By comparing the F1-score of these recovered passages to that of the passages in the development set, we determined the effectiveness of our Dense Retrieval method.

2.2.3 Ensemble model

A hybrid Ensemble technique was created by combining the BM25 and Dense Retrieval models to improve the performance of evidence retrieval systems. We began with the probabilistic BM25 model, which calculates relevance scores using phrase frequency and document length. This model excels at resolving variations in text length and key phrase frequency, which could hinder evidence retrieval. BM25 fails to capture semantic similarity, an essential element in complex text relevance [9]. We implemented Dense Retrieval into Ensemble to address the issue. Deep learning-based Dense Retrieval converts textual data into dense vectors that represent semantic relationships[10]. Dense Retrieval and BM25 would enhance evidence retrieval by boosting term relevance and semantic understanding. Earlier implementations of the BM25 model and the DistilBERT model were utilised by the ensemble technique. Both BM25 scores and cosine similarity between claim and evidence embeddings were min-max scaled to ensure that both models contributed equally [12]. We adjusted the weights for BM25 and Dense Retrieval (bm25_weight and similarity_weight) as well as the number of 'top_k' evidence passages. We used the F-score to measure accuracy and recall in a grid search with different weightings [7]. The optimal number of evidence passages for a range of 'top_k' values was determined by a second grid search. Finally, the F-score evaluates our ensemble model using best 'top_k', 'bm25_weight', and 'similarity_weight'.

2.3 Classification model

2.3.1 Traditional machine learning classifiers

Our automated fact-checking system utilised a variety of classification techniques, such as Gaussian Naive Bayes (GNB) [13], Logistic Regression (LR) [14], Support Vector Machines (SVMs) [15], and Random Forest [16]. These methodologies were selected because they enabled a comprehensive comparative analysis. Despite its simplicity and feature independence, Gaussian Naive Bayes is effective for categorising text [17]. Logistic Regression is interpretable and effective [18]. SVMs can manage high-dimensional spaces, which makes them appropriate for text data [19]. Random Forest is an ensemble technique that effectively regulates feature interactions and resists overfitting [20]. We used claim-evidence pairings and horizontally stacked their embeddings for feature extraction [21] to capture semantic meanings. Synthetic Minority Oversampling Technique (SMOTE) [22] generated synthetic minority class instances to balance the dataset. For classifier compatibility, class labels were encoded to numbers. We implemented using classifiers from scikit-learn. Except for 'max_iter' in Logistic Regression to assure convergence and 'solver' set to 'lbfgs' for its effectiveness with large datasets [23], most parameters were left at their default settings. SVMs handled non-linearly distinct input by setting the "kernel" parameter to "rbf" [19] and using the Radial Basis Function. With the 'n_jobs' option, calculations were parallelized to accelerate training. To assure repeatability, the state of randomness was fixed. The default parameter settings for Logistic Regression and SVMs balanced bias and variance [19][23]. Random Forest's default parameters strike a balance between model performance and computational efficiency [20]. Accuracy on a development set was used to evaluate the efficacy of the model.

2.3.2 DistilBERT Sequence Classifier

We experiment with claim classification using a DistilBERT-based classifier, in addition to conventional machine learning classifiers. Claim status categorisation was facilitated by DistilBERT's capacity to recognise semantic connections between claim and evidence text. DistilBERT performs admirably on NLP tasks such as text categorisation and semantic comprehension[6]. DistilBERT is a transformer-based model trained on a vast corpus of literature. It employs the transformer architecture, which employs attention processes to comprehend the context of words within a phrase and the relationships between sentences, making it suitable for tasks such as comprehending the relationship between a claim and its supporting evidence. DistilBERT is chosen instead of BERT to balance model complexity and computing performance, allowing it to run on common hardware[6]. 'DistilBertTokenizer' and 'DistilBertForSequenceClassification' models were loaded using the transformers library. For deep learning and preprocessing and evaluation, respectively, PyTorch and sklearn were utilised. Using a custom 'ClaimsEvidenceDataset' class, training and development datasets were generated. This class encapsulates claim and evidence text using a tokenizer and truncates or extends them to the maximum length specified. Several crucial hyperparameters were chosen for this implementation. First ,the typical learning rate of 2e-5 is selected for transformer models [24]. It specifies the step size for each iteration as a loss function approaches its minimum value. Second, a batch size of 8 was determined to balance computational efficiency and model performance. Smaller batch sizes can accelerate convergence and improve generalisation, but more iterations are required to analyse the entire dataset [25]. Third, a gradient accumulation step of four was used to accumulate gradients across multiple steps to reduce memory consumption and permit larger effective batch sizes. [26]. The maximum patience for early halting is set to three to prevent overfitting by terminating training if validation loss does not improve after a predetermined number of epochs. Using SMOTE (Synthetic Minority Oversampling Technique), class imbalance in training data is addressed. This technique generates synthetic examples in the feature space to achieve a balanced distribution of classes [22]. In addition, the CrossEntropyLoss function is used to calculate loss using class weights to accommodate for class imbalance. The ‘compute_class_weight’ function of Scikit-learn gives less frequent classes greater weights. The AdamW optimizer, a technique for adaptive learning rate optimisation with weight decay regularisation [27], is used to fine-tune model parameters for transformer models. Scikit-learn's precision score assesses the model's accuracy.

3 Results and evaluations

3.1 Evidence retrieval model

Method	F-score
Ensemble	0.1244
BM25	0.1122
Dense Retrieval	0.0535

Table 3: Evidence retrieval results

Table 3 shows that the ensemble technique combining BM25, and Dense Retrieval outperforms all other methods on development set. However, a low F-score across all models, including the ensemble, suggests that the models are struggling to find the most relevant evidence passages for a given claim, indicating a large potential for improvement in precision and recall. The BM25 model, while marginally inferior to the ensemble technique, struggles to understand semantic meanings and context, essential for complicated tasks like fact-checking. The Dense Retrieval model performs the worst, suggesting that the sentence embeddings or ranking method may not capture the semantic information needed for this job. We suggest fine-tuning phrase embeddings on a task-specific corpus to increase their semantic relevance to improve the evidence retrieval model, notably Dense Retrieval. Different ranking techniques may enhance results.

3.2 Classification model

Classifier	Accuracy score
DistilBERT	0.6558
Random Forest	0.4868
SVMs	0.4664
GNB	0.4297
LR	0.4094

Table 4: Classifier’s accuracy scores

DistilBERT outperforms all classifiers with an accuracy of 0.6558 when evaluating using development set, demonstrating its superiority in NLP tasks like text categorisation. This BERT-based paradigm, incorporating DistilBERT, can recognise word and phrase context, making it easier to relate a claim to its proof. Traditional machine learning models like Random Forest, Support Vector Machines (SVMs), Gaussian Naive Bayes (GNB), and Logistic Regression (LR) have lower accuracy scores between 0.4868 and 0.4094, suggesting they struggle to match transformer-based models like DistilBERT in context comprehension. Despite DistilBERT's higher performance, its accuracy is not yet ideal, suggesting that linguistic context mastery may not be enough for this complex job. The approach may neglect important factors like semantic connection between claim and evidence. Alternative BERT-based designs like RoBERTa or XLNet may improve language comprehension and claim classification model efficacy. The Siamese network or attention processes might also be used to better represent claim-evidence relationships.

3.3 Automated fact checking system

Both the development set and the test set have low F-scores for evidence retrieval (0.1244 and 0.1278, respectively), indicating the system's difficulty in recognising the most relevant evidence passages for a claim. This low F-score indicates that the algorithm fails to recover enough relevant evidence passages and that the retrieved passages are typically irrelevant.

Metrics	Dev	Test
Evidence Retrieval F-score (F)	0.1244	0.1278
Claim Classification Accuracy (A)	0.4416	0.4342
Harmonic Mean of F and A	0.1941	0.1975

Table 5: Integrated system performance results

The claim classifier has a low accuracy (0.4416 and 0.4342 respectively) on both the development and test sets, suggesting that it is difficult to categorise the claim based on the evidence into one of the four classes: SUPPORTS, REFUTES, NOT_ENOUGH_INFO, and DISPUTED. The Harmonic Mean of F and A, a single metric meant to balance evidence retrieval and claim classification tasks, is disappointingly low (0.1941 on development and 0.1975 on test), indicating poor system efficacy. The evidence retrieval model's inability to find relevant evidence passages and the claim classification model's difficulty classifying claims contribute to this low performance. Refining phrase embeddings and the ranking mechanism for the evidence retrieval model and researching other architectures or attention processes for the claim classification model may improve the system's performance. End-to-end models may reduce mistakes from evidence retrieval to claim classification. Such models might map claims directly to their truth, eliminating the evidence collection stage, but they require a big, annotated dataset and a lot of computer power.

4 Conclusion

In conclusion, even though our automated fact-checking system did not perform as expected, the results provide essential insights into the challenges associated with automated fact-checking. They form the basis for future research in this field. Given the increasing sophistication of their models and methodologies, we are optimistic about the ability of automated fact-checking systems to effectively combat disinformation.

References.

[1] *Natural Language Processing with Python*. Accessed: May 12, 2023. [Online]. Available: https://learning.oreilly.com/library/view/natural-language-processing/9780596803346/

[2] "Introduction to Information Retrieval." https://nlp.stanford.edu/IR-book/information-retrieval-book.html (accessed May 12, 2023).

[3] "Speech and Language Processing." https://web.stanford.edu/~jurafsky/slp3/ (accessed May 12, 2023).

[4] S. Robertson, H. Zaragoza, and M. Taylor, "Simple BM25 extension to multiple weighted fields," in *Proceedings of the thirteenth ACM international conference on Information and knowledge management*, Washington D.C. USA: ACM, Nov. 2004, pp. 42–49. doi: 10.1145/1031171.1031181.

[5] N. Reimers and I. Gurevych, "Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks," in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, Hong Kong, China: Association for Computational Linguistics, Nov. 2019, pp. 3982–3992. doi: 10.18653/v1/D19-1410.

[6] V. Sanh, L. Debut, J. Chaumond, and T. Wolf, "DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter." arXiv, Feb. 29, 2020. doi: 10.48550/arXiv.1910.01108.

[7] M. Sokolova and G. Lapalme, "A systematic analysis of performance measures for classification tasks," *Inf. Process. Manag.*, vol. 45, no. 4, pp. 427–437, Jul. 2009, doi: 10.1016/j.ipm.2009.03.002.

[8] D. M. W. Powers, "Evaluation: from precision, recall and F-measure to ROC, informedness, markedness and correlation." arXiv, Oct. 10, 2020. doi: 10.48550/arXiv.2010.16061.

[9] S. Robertson and H. Zaragoza, "The Probabilistic Relevance Framework: BM25 and Beyond," *Found. Trends Inf. Retr.*, vol. 3, pp. 333–389, Jan. 2009, doi: 10.1561/1500000019.

[10] V. Karpukhin *et al.*, "Dense Passage Retrieval for Open-Domain Question Answering." arXiv, Sep. 30, 2020. doi: 10.48550/arXiv.2004.04906.

[11] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed Representations of Words and Phrases and their Compositionality," in *Advances in Neural Information Processing Systems*, Curran Associates, Inc., 2013. Accessed: May 12, 2023. [Online]. Available: https://proceedings.neurips.cc/paper/2013/hash/9aa42b31882ec039965f3c4923ce901b-Abstract.html

[12] S. G. K. Patro and K. K. Sahu, "Normalization: A Preprocessing Stage," *IARJSET*, pp. 20–22, Mar. 2015, doi: 10.17148/IARJSET.2015.2305.

[13] K. P. Murphy, *Machine learning: a probabilistic perspective*. in Adaptive computation and machine learning series. Cambridge, MA: MIT Press, 2012.

[14] A. Cucchiara, "Applied Logistic Regression," *Technometrics*, vol. 34, pp. 358–359, Mar. 2012, doi: 10.1080/00401706.1992.10485291.

[15] C. Cortes and V. Vapnik, "Support-vector networks," *Mach. Learn.*, vol. 20, no. 3, pp. 273–297, Sep. 1995, doi: 10.1007/BF00994018.

[16] L. Breiman, "Random Forests," *Mach. Learn.*, vol. 45, no. 1, pp. 5–32, Oct. 2001, doi: 10.1023/A:1010933404324.

[17] A. McCallum and K. Nigam, "A Comparison of Event Models for Naive Bayes Text Classification".

[18] S. Le Cessie and J. C. Van Houwelingen, "Ridge Estimators in Logistic Regression," *J. R. Stat. Soc. Ser. C Appl. Stat.*, vol. 41, no. 1, pp. 191–201, 1992, doi: 10.2307/2347628.

[19] M. A. Hearst, S. T. Dumais, E. Osuna, J. Platt, and B. Scholkopf, "Support vector machines," *IEEE Intell. Syst. Their Appl.*, vol. 13, no. 4, pp. 18–28, Jul. 1998, doi: 10.1109/5254.708428.

[20] A. Liaw and M. Wiener, "Classification and Regression by randomForest," vol. 2, 2002.

[21] X. Li, W. Wang, J. Fang, L. Jin, H. Kang, and C. Liu, "PEINet: Joint Prompt and Evidence Inference Network via Language Family Policy for Zero-Shot Multilingual Fact Checking," *Appl. Sci.*, vol. 12, no. 19, Art. no. 19, Jan. 2022, doi: 10.3390/app12199688.

[22] "SMOTE: Synthetic Minority Over-sampling Technique | Journal of Artificial Intelligence Research." https://www.jair.org/index.php/jair/article/view/10302 (accessed May 12, 2023).

[23] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin, "LIBLINEAR: A Library for Large Linear Classification".

[24] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding." arXiv, May 24, 2019. doi: 10.48550/arXiv.1810.04805.

[25] N. S. Keskar, D. Mudigere, J. Nocedal, M. Smelyanskiy, and P. T. P. Tang, "On Large-Batch Training for Deep Learning: Generalization Gap and Sharp Minima." arXiv, Feb. 09, 2017. doi: 10.48550/arXiv.1609.04836.

[26] M. Ott, S. Edunov, D. Grangier, and M. Auli, "Scaling Neural Machine Translation." arXiv, Sep. 04, 2018. doi: 10.48550/arXiv.1806.00187.

[27] I. Loshchilov and F. Hutter, “Decoupled Weight Decay Regularization.” arXiv, Jan. 04, 2019. doi: 10.48550/arXiv.1711.05101.