Deep Learning-Based Claim Matching with Multiple Negatives Training

Anna Neumann[1], **Dorothea Kolossa**[2], **Robert M. Nickel**[3]
[1]Ruhr-Universität Bochum, Germany
[2]Technische Universität Berlin, Germany
[3]Bucknell University, Lewisburg, PA, USA
`anna.neumann1@uni-due.de`
`dorothea.kolossa@tu-berlin.de`
`robert.nickel@bucknell.edu`

Abstract

Numerous approaches for the implementation of automated fact-checking pipelines have been proposed and reviewed recently (Guo et al., 2022). A key part in these pipelines is a claim matching module that seeks to match new incoming claims with potentially existing, verified claims in a database of completed fact checks. To that end, we propose a modification of the two-stage deep learning-based approach for claim matching which won the CLEF CheckThat! 2022 Subtask 2A Challenge (Shliselberg and Dori-Hacohen, 2022). With our modification, we were able to reduce the error rate of the winning algorithm by more than 20%. This was accomplished by employing a loss function that fuses information from not only a single, but from multiple non-matching (i.e. *negative*) examples into the training process at each iteration.

1 Introduction

Fact-checking became an increasingly important step in journalistic work in response to the proliferation of fake news online. Misinformation on the internet spreads at a speed and scale that makes it more and more difficult for human fact-checkers to react in a timely manner. It is therefore a desirable goal to automate parts of the process. Claim matching is one portion of this process, in which an incoming claim is checked against a database of human-verified claims. The automation of claim matching has received a considerable amount of interest in recent years (Shaar et al., 2022a,b; Kazemi et al., 2021; Nakov et al., 2021).

For the CLEF CheckThat! 2022 Subtask 2A Challenge, Shliselberg and Dori-Hacohen (2022) proposed a two-step pipeline as the winning entry. First, a deep pre-trained language model based on BERT (Devlin et al., 2019) is fine-tuned to generate a selection of candidates of relevant claims from the database. Second, the candidates are re-ranked by fine-tuning a generative language model.

The contribution of this paper is an expansion of the winning method of Shliselberg and Dori-Hacohen (2022) by (1) including additionally mined negative examples into the training objective, (2) investigating a Ranked List Loss (RLL) as an alternative cost function, and (3) expanding the analysis of the proposed scheme by including Mean Average Recall (MAR).

2 Methods

We begin by introducing the employed dataset and the statistical benchmark used for the mining of negative examples. Then, we present the different training objectives for the two stages of the approach. Lastly, we discuss the evaluation metrics.

2.1 Data and Statistical Benchmark

For the dataset, we used the English portion (Subtask 2A) of the dataset provided for the CLEF CheckThat! 2022 Challenge (Nakov et al., 2022) based on verified claims from `Snopes.com`. The dataset consists of 13,835 verified claims, denoted with c for *claim*, and 1,400 input claims, denoted with t for *tweet*. The input claims are divided into 999 tweets for training, 199 tweets for development and 202 tweets for testing. For neural network training, the body of each fact-checked article is tokenized before it is fed into the respective networks.

For our statistical benchmark, we applied a standard BM25[1] ranking algorithm (Robertson and Zaragoza, 2009). The preprocessing for this step includes concatenating the title, subtitle and body of the claim, transforming everything into lowercase, followed by Porter stemming (Porter, 1980). BM25 provides a ranked list of claims for each input tweet. Each non-matching claim is defined as a 'negative' and the matching claim is defined as the 'positive'. The five highest-ranking negatives are mined for our experiments.

[1] *BM* is an abbreviation for *best matching*.

2.2 Candidate Selection

As suggested by Shliselberg and Dori-Hacohen (2022), we use Sentence-T5 (Ni et al., 2022) for the initial candidate selection. It is part of the family of sentence transformers (Reimers and Gurevych, 2019), i.e. deep neural language models based on self-attention mechanisms (Vaswani et al., 2017). It produces sentence embeddings projected on the Euclidean unit circle, which makes the angle between the embeddings a measure of contextual dissimilarity. We use the Multiple Negatives Ranking (*MNR*) Loss (Henderson et al., 2017), which minimizes the distance between the input and positive example and maximizes the distance to all other examples in the batch. Using batches $\mathcal{B} \in \mathcal{D}$ of sets $\mathcal{D} = \{(t_i, c_i^+, c_i^-)\}$ with a tweet t_i, a positive c_i^+ and a negative claim c_i^-, the dot product scoring $S_\theta(t_i, c_i)$ by the specific neural network θ, and a fixed temperature τ, the loss function becomes

$$\mathcal{L}_{MNR}(\mathcal{B},\theta) = -\frac{1}{|\mathcal{B}|}\sum_{i\in\mathcal{B}} \log \frac{\exp\left(S_\theta(t_i, c_i^+)/\tau\right)}{Q_{\theta,i}}$$

$$\text{with} \quad Q_{\theta,i} = \textstyle\sum_{j\in\mathcal{B}} \exp(S_\theta(t_i, c_j^+)/\tau) + \exp(S_\theta(t_i, c_j^-)/\tau). \qquad (1)$$

As we will see in Section 3, the minimization of the MNR loss can lead to significant performance improvements of the overall system when it is expanded by mining not one but multiple negatives for every paired example. This gives the model potentially even more context, as the tweet is compared to every claim in the batch. We used up to five negatives that ranked highest in the BM25 run. For sets $\mathcal{D} = \{(t_i, c_i^+, c_{i,1}^-, c_{i,2}^-, c_{i,3}^-, c_{i,4}^-, c_{i,5}^-)\}$ the expanded loss is defined by

$$\mathcal{L}_{MNR}(\mathcal{B},\theta) = -\frac{1}{|\mathcal{B}|}\sum_{i\in\mathcal{B}} \log \frac{\exp(S_\theta(t_i, c_i^+)/\tau)}{\sum_{j\in\mathcal{B}} \mathcal{M}_{\theta,i,j}}$$

$$\text{with} \quad \mathcal{M}_{\theta,i,j} = \exp(S_\theta(t_i, c_j^+)/\tau) + \textstyle\sum_{k=1}^{5} \exp(S_\theta(t_i, c_{j,k}^-)/\tau). \qquad (2)$$

2.3 Generative Re-Ranking

For the second step, we, again, follow closely the setup proposed by Shliselberg and Dori-Hacohen (2022). The fine-tuned S-T5 network from the first step is used to generate ranked lists of the five most similar claims to each input tweet. The combination of the claim and each input tweet is then employed separately to fine-tune the generative deep language model GPTNeo[2] (Black et al., 2022). The generative model has the capacity to calculate the conditional probability $p(t|c)$ for tweet and claim pairs, which, in turn, can be utilized to re-rank the given list of five tweet/claim pairs provided by the preceding stage. The tweets and claims therefore have to be converted into prompts with beginning-of-sentence $< bos >$ and end-of-sentence $< eos >$ tokens: $< bos > c < eos >< bos > t < eos >$.

For the fine-tuning of GPTNeo, we considered a number of loss functions, primarily motivated by the loss types recommended in the commensurate literature. We expanded on the list of losses considered by Shliselberg and Dori-Hacohen (2022) by also including the standard ranked list loss (*RLL*) into our analysis. The RLL (Nogueira dos Santos et al., 2020) is defined by

$$\mathcal{L}_{RLL}(\mathcal{B},\theta) = \sum_{i\in\mathcal{B}} \max\{0, \lambda - \log p_\theta(t_i|c_i^+) + \log p_\theta(t_i|c_i^-)\}, \qquad (3)$$

in which the notation p_θ denotes the dependence of the probability estimate on the network parameters θ. The hinge margin λ is a hyperparameter of the training procedure. The also employed NL3U loss (Nogueira dos Santos et al., 2020) is based on the negative log-likelihood loss (Lesota et al., 2021). It incorporates the *unlikelihood probability*[3] of the negative claim:

$$\mathcal{L}_{NL3U}(\mathcal{B},\theta) = -\sum_{i\in\mathcal{B}} \log p_\theta(t_i|c_i^+) + \log(1 - p_\theta(t_i|c_i^-)). \qquad (4)$$

Shliselberg and Dori-Hacohen (2022) also introduced the mixed objective

$$\mathcal{L}_{Mix} = \mathcal{L}_{MI_1} + \mathcal{L}_{MI_2} + \mathcal{L}_{NL3U}, \qquad (5)$$

which utilizes a hinged prior mutual information loss ($\mathcal{L}_{MI_1}$) and a posterior-based hinged mutual information loss ($\mathcal{L}_{MI_2}$). The loss $\mathcal{L}_{MI_1}$ maximizes mutual information but reverts back to the maximum likelihood estimate above a threshold λ:

$$\mathcal{L}_{MI_1} = \begin{cases} \mathcal{K}_\theta^{\text{MI}} & \text{if } -\log\frac{p_\theta(t|c)}{p_\theta(t)} < \lambda \\ \mathcal{K}_\theta^{\text{MLE}} & \text{otherwise} \end{cases} \qquad (6)$$

with

$$\begin{aligned} \mathcal{K}_\theta^{\text{MI}} &= \mathrm{E}_{T,C}[-\log p_\theta(t|c) + \log p_\theta(t)] \\ \mathcal{K}_\theta^{\text{MLE}} &= \mathrm{E}_{T,C}[-\log p_\theta(t|c)] \end{aligned}$$

[2] GPTNeo contains on the order of 1.3 billion parameters.

[3] The *unlikelihood probability* is defined as one minus the probability.

To also model the posterior, the input order is flipped and $\mathcal{L}_{MI_2}$ is defined as:

$$\mathcal{L}_{MI_2} = \mathrm{E}_{C,T}[max(0, \lambda - \log p_\theta(c|t) + \log p_\theta(c))] \tag{7}$$

We utilize $\mathcal{L}_{RLL}$, $\mathcal{L}_{NL3U}$, $\mathcal{L}_{Mix}$ and a sum of both mutual information losses, i.e.

$$\mathcal{L}_{MutInf} = \mathcal{L}_{MI_1} + \mathcal{L}_{MI_2}, \tag{8}$$

as training objectives to fine-tune the generative model.

2.4 Evaluation Metrics

The CheckThat! Challenge uses Mean Average Precision (*MAP*) for evaluation. In addition to MAP scores, we also considered the Mean Average Recall (*MAR*) in our analysis. Average Recall at length k is defined as

$$\mathrm{AR@}k\{R, g\} = \sum_{i=1}^{k} \mathbb{1}[R[i] == g] \tag{9}$$

for a ranked list R with one gold label g, using $\mathbb{1}$ as an indicator function. MAR is the mean over all ranked lists and label pairs denoted as Ω:

$$\mathrm{MAR@}k\{\Omega\} = \frac{1}{|\Omega|} \sum_{R,g \in \Omega} \mathrm{AR@}k\{R, g\} \tag{10}$$

We are using the standard definition of MAP@k as the mean value over the average precision AP@$k\{R, g\} = \sum_{i=1}^{k} \mathbb{1}[R[i] == g]\frac{1}{i}$. MAP and MAR values are bounded between zero and one. MAP@1 equals MAR@1 by definition. We therefore only report MAP@1 scores.

3 Experiments

In our experiments we generated five S-T5 models through fine-tuning: One with an MNR loss with one negative, one with an MNR loss with two negatives, and so forth, up to one with an MNR loss with five negatives. Each S-T5 model is then paired with four GPTNeo models, one for each of the four loss functions described in Section 2.3, to create a total number of 20 systems. The ranked lists generated by the S-T5 models are used to fine-tune the respectively paired GPTNeo models. All training was performed on a server with two NVIDIA RTX A6000 GPUs with 48 GB memory each. We report MAP@1, MAP@5, and MAR@5 scores in all our cases. All evaluations are performed on the test set defined by the CheckThat! Challenge. Finally, we compare top performing methods to the BM25 benchmark.

3.1 Candidate Selection

S-T5 is fine-tuned using a batch size of 3, the AdamW optimizer with a constant learning rate of 5e-6, an MNR loss temperature τ of 0.1 and a maximum of 128 tokens for each input for a single epoch. Results are presented in Table 1. The respective highest values in each column are highlighted in bold face.

#Neg	MAP@1	MAP@5	MAR@5
1	0.896	0.932	0.975
2	0.896	0.933	**0.980**
3	**0.901**	**0.936**	**0.980**
4	0.896	0.934	**0.980**
5	0.891	0.931	**0.980**

Table 1: Evaluation of the S-T5 MNR Fine-Tuning.

Both, MAP@1 and MAP@5, peak for a training with three negatives and then both decline again for a training with four and five negatives. The improvement that a training with three negatives affords over a training with one negative is on the order of 0.5 percentage points in both cases. This finding supports our hypothesis that training with more negatives provides better context for the model. Yet, training with too many negatives appears to put too much weight on the rejection of negatives and too little weight on the support of positives. The MAR@5 values increase slightly for two negatives and then stay constant at 0.980.

3.2 Generative Re-Ranking

Fine-tuning the re-rankers for one epoch includes a batch size of 1, a maximum of 256 input tokens for padded prompts, a hinge margin $\lambda = 2$ and the AdamW optimizer with a learning rate of 2e-5. Our system with one-negative-training is essentially the system proposed by Shliselberg and Dori-Hacohen (2022). The MAP@5 score we obtained for the one-negative/mixed case matches the result reported by them closely. We attribute slight deviations to differences in random initialization, differing batch sizes due to computational limits and batch shuffling. We omitted the MAR@5 results, since these did not change and stayed constant at 0.980 for every number of negatives above one. The MAR@5 results for one negative all came out to 0.975. The best-performing loss for each base model in each column is highlighted in bold. It is apparent that the mixed approach performed best for all candidate selection models and that the RLL approach

#Neg	Loss	MAP@1	MAP@5
1	Mixed	**0.921**	**0.947**
	NL3U	0.911	0.941
	MutInf	0.896	0.935
	RLL	0.658	0.778
2	Mixed	**0.936**	**0.958**
	NL3U	0.901	0.938
	MutInf	0.891	0.933
	RLL	0.757	0.850
3	Mixed	**0.926**	**0.953**
	NL3U	0.921	0.949
	MutInf	0.891	0.933
	RLL	0.223	0.475
4	Mixed	**0.926**	**0.953**
	NL3U	0.921	0.947
	MutInf	0.886	0.931
	RLL	0.871	0.925
5	Mixed	**0.926**	**0.953**
	NL3U	0.916	0.945
	MutInf	0.906	0.942
	RLL	0.183	0.421

Table 2: Evaluations over the test set of the CLEF CheckThat! 2022 Subtask 2A Challenge with GPTNeo re-ranking for four training objectives.

performed the worst. The NL3U loss alone performs significantly better than RLL and achieves a peak performance in MAP@1 and MAP@5 for three negatives. The mutual information loss consistently performs a bit worse than NL3U alone and achieves its best performance for five negatives. The mixed loss consistently outperforms other losses and peaks for two negatives with a MAP@1 value of 0.936 and a MAP@5 value of 0.958. When compared to the MAP@5 value of 0.947 for the training with a single negative, it can be seen that the error rate, when defined as one minus MAP@5, is reduced by over 20% with the proposed multiple negatives training. We attribute the superior performance of the mixed loss training to the fact that it incorporates different aspects of text similarity, measuring mutual information on the one side and contrasting it with information about negative examples on the other side.

3.3 Discussion

We present the BM25 evaluation, the best-performing candidate selection model and the best-performing generative re-ranking model in Table 3. The BM25 method already provided comparatively high scores with a MAP@5 value of 0.852 and a MAR@5 value of 0.936. The best-performing S-T5 network, fine-tuned with a three-negatives MNR loss, outperforms the BM25 baseline by more than eight percentage points with a MAP@5 of 0.936. The MAP@1 value increases by over ten points and the MAR@5 value by over four points. The fine-tuned GPTNeo system based on a two-negatives S-T5 model with a mixed objective yields the best performance with a MAP@5 value of 0.958. It outperforms the reference model with mixed loss and one-negative-training by about one percentage point and the best candidate selection model by over two percentage points.

Model	MAP@1	MAP@5	MAR@5
BM25	0.797	0.852	0.936
S-T5$_{3N}$	0.901	0.936	**0.980**
GPT$_{Mix,2N}$	**0.936**	**0.958**	**0.980**

Table 3: Performance summary for experiments over the test set from the CLEF CheckThat! 2022 Subtask 2A Challenge.

4 Conclusions and Future Work

The winning algorithm of the CLEF CheckThat! 2022 Challenge for claim matching (Subtask 2A) consists of a two step process: (1) candidate selection and (2) generative re-ranking. Both steps are achieved with fine-tuned deep neutral networks. We generalized the loss function used in the fine-tuning of the candidate selection network by including not just one negative example but multiple negative examples. Through experimentation with various configurations and loss functions we were able to create an overall system that improves the MAP@1 and MAP@5 scores by over one percentage points each, leading to an effective reduction in error rate of around 20%. Future work may include an incorporation of other, more powerful large language models (*LLMs*) in lieu of GPTNeo.

Acknowledgements

The work on this paper was performed as part of a master's thesis at the Ruhr-University Bochum in support of the project "noFake", funded by the German Federal Ministry of Research and Education (BMBF) under Award Identifier F16KIS1519. Our implementations are based on the code provided by Shliselberg and Dori-Hacohen (2022) at https://github.com/RIET-lab/GenerativeClaimMatchingPipeline.

References

Sidney Black, Stella Biderman, Eric Hallahan, Quentin Anthony, Leo Gao, Laurence Golding, Horace He, Connor Leahy, Kyle McDonell, Jason Phang, Michael Pieler, Usvsn Sai Prashanth, Shivanshu Purohit, Laria Reynolds, Jonathan Tow, Ben Wang, and Samuel Weinbach. 2022. GPT-NeoX-20B: An Open-Source Autoregressive Language Model. In *Proceedings of BigScience Episode #5 – Workshop on Challenges & Perspectives in Creating Large Language Models*, pages 95–136, Dublin, Ireland. Association for Computational Linguistics.

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

Zhijiang Guo, Michael Schlichtkrull, and Andreas Vlachos. 2022. A survey on automated fact-checking. *Transactions of the Association for Computational Linguistics*, 10:178–206.

Matthew L. Henderson, Rami Al-Rfou, Brian Strope, Yun-Hsuan Sung, László Lukács, Ruiqi Guo, Sanjiv Kumar, Balint Miklos, and Ray Kurzweil. 2017. Efficient Natural Language Response Suggestion for Smart Reply. Computing Research Repository, arXiv: 1705.00652.

Ashkan Kazemi, Kiran Garimella, Devin Gaffney, and Scott Hale. 2021. Claim Matching Beyond English to Scale Global Fact-Checking. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 4504–4517, Online. Association for Computational Linguistics.

Oleg Lesota, Navid Rekabsaz, Daniel Cohen, Klaus Antonius Grasserbauer, Carsten Eickhoff, and Markus Schedl. 2021. A modern perspective on query likelihood with deep generative retrieval models. In *Proceedings of the 2021 ACM SIGIR International Conference on Theory of Information Retrieval*, pages 185–195, New York, USA. Association for Computing Machinery.

Preslav Nakov, Alberto Barrón-Cedeño, Giovanni da San Martino, Firoj Alam, Julia Maria Struß, Thomas Mandl, Rubén Míguez, Tommaso Caselli, Mucahid Kutlu, Wajdi Zaghouani, Chengkai Li, Shaden Shaar, Gautam Kishore Shahi, Hamdy Mubarak, Alex Nikolov, Nikolay Babulkov, Yavuz Selim Kartal, Michael Wiegand, Melanie Siegel, and Juliane Köhler. 2022. Overview of the clef–2022 checkthat! lab on fighting the covid-19 infodemic and fake news detection. In *Proceedings of the 13th International Conference of the CLEF Association*, pages 495–520, Bologna, Italy.

Preslav Nakov, David Corney, Maram Hasanain, Firoj Alam, Tamer Elsayed, Alberto Barrón-Cedeño, Paolo Papotti, Shaden Shaar, and Giovanni Da San Martino. 2021. Automated Fact-Checking for Assisting Human Fact-Checkers. In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence*, pages 4551–4558, Montreal, Canada. International Joint Conferences on Artificial Intelligence Organization.

Jianmo Ni, Gustavo Hernandez Abrego, Noah Constant, Ji Ma, Keith Hall, Daniel Cer, and Yinfei Yang. 2022. Sentence-T5: Scalable Sentence Encoders from Pretrained Text-to-Text Models. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 1864–1874, Dublin, Ireland. Association for Computational Linguistics.

Cicero Nogueira dos Santos, Xiaofei Ma, Ramesh Nallapati, Zhiheng Huang, and Bing Xiang. 2020. Beyond [CLS] through Ranking by Generation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1722–1727, Online. Association for Computational Linguistics.

Martin F. Porter. 1980. An algorithm for suffix stripping. *Program*, 14(3):130–137.

Nils Reimers and Iryna Gurevych. 2019. 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)*, pages 3982–3992, Hong Kong, China. Association for Computational Linguistics.

Stephen Robertson and Hugo Zaragoza. 2009. The Probabilistic Relevance Framework: BM25 and Beyond. *Foundations and Trends in Information Retrieval*, 3(4):333–389.

Shaden Shaar, Firoj Alam, Giovanni Da San Martino, and Preslav Nakov. 2022a. The Role of Context in Detecting Previously Fact-Checked Claims. In *Findings of the Association for Computational Linguistics: NAACL 2022*, pages 1619–1631, Seattle, USA. Association for Computational Linguistics.

Shaden Shaar, Nikola Georgiev, Firoj Alam, Giovanni Da San Martino, Aisha Mohamed, and Preslav Nakov. 2022b. Assisting the Human Fact-Checkers: Detecting All Previously Fact-Checked Claims in a Document. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 2069–2080, Abu Dhabi, UAE. Association for Computational Linguistics.

Michael Shliselberg and Shiri Dori-Hacohen. 2022. RIET Lab at CheckThat! 2022: Improving decoder based re-ranking for claim matching. *Proceedings of the Working Notes of CLEF 2022 - Conference and Labs of the Evaluation Forum*, 3180:671–678.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 6000–6010, Long Beach, USA.