COMP90042 Natural Language Processing Project: Evidence Retrieval and Claim Classification for Climate Change Statements

Tue9AM_Group2

Abstract

This report presents the methodology and results of the Neutral Semantics Matching Network(NSMN) model, applied to automatic fact-checking within the domain of Climate Science. It examines the performance of the model under various hyper-parameter settings for tasks involving information retrieval and classification. The report concludes with a discussion of potential future research directions in this area.

1 Introduction

Climate change poses a significant threat to humanity, gaining increased public attention in recent times. Despite robust scientific evidence supporting the realities of climate change, some skeptics continue to challenge the scientific consensus, undermining global efforts to address this critical issue. These skeptics often use social media platforms, which facilitate the rapid spread of opinions, to propagate counterfactual claims, confusing public discourse. In response to this challenge, the development of automated fact-checking technology is crucial. Such technology can be deployed on social media platforms to identify and label misinformation, helping to counteract the spread of false claims. Automated fact-checkers typically consist of two primary components:

Evidence Retrieval: This component searches a knowledge base to find scientific evidence most relevant to a given claim.

Classification: This component evaluates the retrieved evidence to determine the veracity of the claim, categorizing it into one of the following labels: "SUPPORTS", "REFUTES", "NOT ENOUGH INFO(NEI)", or "DISPUTED".

This report discusses the implementation and effectiveness of the NSMN model (Nie et al., 2019) in automating fact-checking for climate-related claims under limited resources, exploring different hyper-parameter settings and approaches to improve the performance.

2 Approach

2.1 Data Preprocessing

To emphasise the focus on keywords, the claim and evidence sentences undergo a simple three-step data preprocessing, which lemmatizes each token, and discards any stop words and punctuation.

2.2 Evidence Filtering (EF)

With over one million pieces of evidence, it is crucial to narrow down the potential evidence relevant to each claim. To achieve this, a Word2Vec model is pretrained with 128 dimensions and a window length of 30 on the training and development claims and all evidence. The skip-gram representations in Word2Vec model exhibit a linear structure that supports a meaningful combination of word embeddings via an element-wise addition (Mikolov et al., 2013). Therefore, dense representations for sentences are generated by summing the word vectors. Evidence embeddings are used to train a 3000-nearest-neighbour model(3000-NN) for the extraction of three thousand most similar evidences per claim based on cosine similarity. This procedure collects the most related evidences, improving the quality of negative sampling, and substantially reducing the training and testing size for the evidence retrieval model to a feasible volume.

2.3 Evidence Retrieval (ER)

We adopted the same architecture of NSMN across evidence retrieval and claim classification tasks and trained them separately as shown in Figure 1. For the sake of clarity, we refer to it as rNSMN. In rNSMN, the input and output are a pair of claim and evidence sequences of Word2Vec word embeddings and the probability that the evidence is related to the claim, respectively.

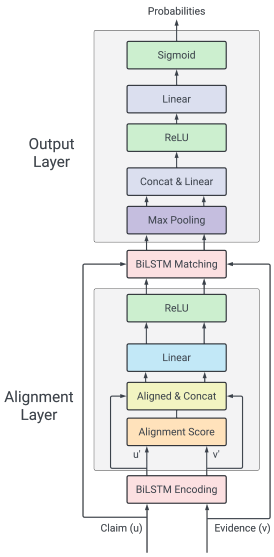


Figure 1: NSMN Model Architecture

2.3.1 NSMN

The NSMN consists of four main layers, namely:

Encoding Layer: A bidirectional LSTM to understand and encode the context of every input token sequentially, producing a layer of contextual representations on top of the Word2Vec embeddings. Here, $\mathbf{U} \in \mathbb{R}^{128\times m}$ represents the claim, and $\mathbf{V} \in \mathbb{R}^{128\times n}$ denotes the evidence, where m and n represent the max length of claim sequence and evidence sequence respectively. After this layer, we derive the contextual embedding below:

$$\mathbf{U}' = \text{BiLSTM}(\mathbf{U}) \in \mathbb{R}^{256\times m} \quad (1)$$

$$\mathbf{V}' = \text{BiLSTM}(\mathbf{V}) \in \mathbb{R}^{256\times n} \quad (2)$$

Alignment Layer: With the contextual representations of the two input sequences computed above, this layer aligns them using cross-product and produces a token-wise alignment score. Then, it applies softmax to normalise the alignment score for each token of a sequence and builds the aligned representation of one sequence using the other sequence by computing the relevant semantic content using the weighted sum of encoded tokens.

$$\mathbf{E} = \mathbf{U'}^{\top}\mathbf{V}' \in \mathbb{R}^{m\times n} \quad (3)$$

$$\widetilde{\mathbf{U}} = \mathbf{V}' \cdot \text{Softmax}_{\text{col}}(\mathbf{E}^{\top}) \in \mathbb{R}^{256\times m} \quad (4)$$

$$\widetilde{\mathbf{V}} = \mathbf{U}' \cdot \text{Softmax}_{\text{col}}(\mathbf{E}) \in \mathbb{R}^{256\times n} \quad (5)$$

The aligned and encoded representations are combined and processed by a fully connected linear layer with a rectifier activation function, where $\mathbf{S} \in \mathbb{R}^{256\times m}$ and $\mathbf{T} \in \mathbb{R}^{256\times n}$

$$\mathbf{S} = f([\mathbf{U}', \widetilde{\mathbf{U}}, \mathbf{U}' - \widetilde{\mathbf{U}}, \mathbf{U}' \circ \widetilde{\mathbf{U}}]) \quad (6)$$

$$\mathbf{T} = f([\mathbf{V}', \widetilde{\mathbf{V}}, \mathbf{V}' - \widetilde{\mathbf{V}}, \mathbf{V}' \circ \widetilde{\mathbf{V}}]) \quad (7)$$

Matching Layer: A second bidirectional LSTM to semantically match between the upstream aligned representation and original input sequence.

$$\mathbf{P} = \text{BiLSTM}([\mathbf{S}, \mathbf{U}]) \in \mathbb{R}^{512\times m} \quad (8)$$

$$\mathbf{Q} = \text{BiLSTM}([\mathbf{T}, \mathbf{V}]) \in \mathbb{R}^{512\times n} \quad (9)$$

Output Layer: The two matching sequences are summarised by max-pooling and concatenated with their absolute difference and element-wise multiplication. This combined vector is mapped to the final output of label probabilities by two fully connected linear layers with a rectifier activation function in between.

2.3.2 Training

Considering the significant imbalance of positive and negative evidences for each claim, the annealed sampling strategy was applied in training to gradually increase the proportion of positive evidences after every epoch in order to make the model more robust in selecting related evidences while discarding apparent negative evidences. Therefore, the model was trained with all positive evidences and the K negative evidences drawn from the 3000 nearest neighbours for each claim.

2.3.3 Baseline

The baseline for ER is a model that combines several randomly selected ground truth evidences and several randomly selected evidences from the knowledge source.

2.4 Claim Classification (CC)

The model architecture used for CC (cNSMN) mirrors that of the ER component. However, instead of processing individual claim-evidence pairs, the top four pieces of evidence returned by ER are concatenated. The output layer dimension is also adjusted to four, representing the probability of each classification category. Additionally, we have engineered three supplementary features as introduced by Nie et al. (2019).

Claim evidence indicator: [1,0] for claim tokens, and [0,1] for evidence tokens.

WordNet indicators: for each token in the claim sequence, the indicator is set to 1 if any of the 8 phenomenons listed in table 1 is found in the evidence sequence. The same applies vice versa for the evidence sequence. To better distinguish "REFUTES" and "DISPUTED" classes from "SUPPORTS", this

Category	Feature
Similarity	Exact same lemma
Similarity	Hyponym
Similarity	Hypernym
Similarity	Hyponym with 1 edge distance
Similarity	Hypernym with 1 edge distance
Similarity	Hyponym with 2 edge distance
Similarity	Hypernym with 2 edge distance
Opposite	Antonym

Table 1: WordNet features. The edge distance is in the context of the WordNet topological graph.

approach is modified to generate a two-element vector, where the first corresponds to the 7 events that signal similarity, and the second corresponds to antonym.

Numerical encoding: each numeric token is hashed to a randomly generated five-element encoding, set to zeros for non-numeric tokens.

2.4.1 Data Augmentation

With a combined training and development set containing only 1382 claims, data augmentation is needed to reduce underfitting, where the model consistently favours predicting the dominant class, "SUPPORTS". The CC model takes as input a concatenation of all four evidences extracted using the ER model, and rearranging the evidence order results in different input sequences while maintaining unchanged semantics. This strategy draws inspiration from the approach of randomly shuffling sentence order (Yan et al., 2019). Consequently, permutations of the evidences are generated for each claim, effectively expanding the training dataset to up to 24 times its original size.

A similar procedure is applied to the development and testing set. The model makes a prediction for each input pair, consisting of the claim and one of its concatenated permutations. Majority voting is then used to elect the final classification.

2.4.2 Baseline

The baseline used for analyzing CC model performance includes the original NSMN model (Nie et al., 2019), and majority voting which achieves 0.4416 accuracy on S_{dev} already.

3 Experiments

In order to optimise the performance of the models, we explored various model configurations during the development using the training (S_{train}) and development (S_{dev}) datasets.

Categories	Proportion	Average
Disputed	0.67	0.86
NEI	0.34	0.65
Refutes	0.70	0.82
Supports	0.70	0.81

Table 2: The performance of 3000-NN over different categories of claims

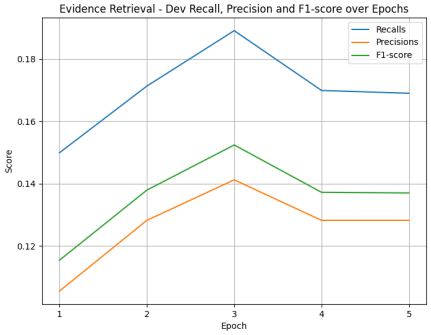


Figure 2: ER performance on S_{dev} over training epochs

3.1 Evidence Filtering

In this section, we explored increasing the size of the nearest neighbour (NN) group from 100 to 4000. Eventually, we settled on using 3000NN as it provided the best balance between training size and accuracy. As Table 2 illustrates, the "Proportion" column shows the percentage of claims for which all relevant evidence was found, while "Average" represents the pooled average proportion of evidence found across all claims. On average, this approach retains 80% of positive evidence.

3.2 Evidence Retrieval

In this section, we will employ not only the F1-score but also the recall and precision as the evaluation metrics to offer more detailed insights into the change in performance of our model and direct the design decisions.

As seen in Figure 2, after training the ER model for 3 epochs with the annealed sampling and a learning rate of 0.001, it exhibited the highest F1-score in the development set evaluation. It is worth noting that the improvement over the first 3 epochs was rather steep in the figure, suggesting that a lower learning rate could have been applied to precisely fine-tune the model.

Another important parameter in neural network

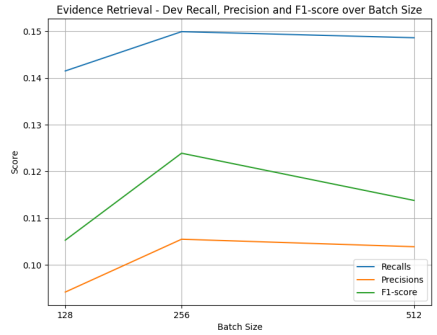


Figure 3: ER performance on S_{dev} over different training batch sizes

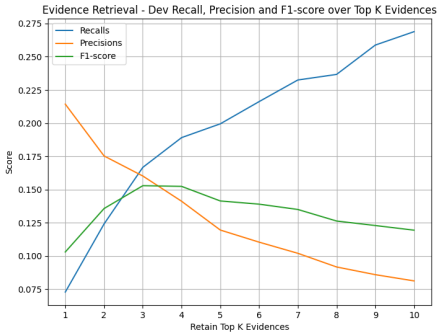


Figure 4: ER performance on S_{dev} over outputting top K evidences

training is the batch size, which determines the number of parameters being updated in each epoch and will affect the final prediction accuracy as well as its ability to generalise (Keskar et al., 2016). We examined training batch sizes 128, 256 and 512 for the first epoch, and the top-performing batch size was 256 from Figure 3.

In addition, we also tuned the best number of evidences to retain in the output to balance between the recall and precision. Figure 4 illustrates that the model could reach the highest F1-score when K was between 3 and 4. This is understandable because the average and maximum number of related evidences for a claim is 3.34 and 5 respectively from inspecting the dataset. However, we realised that outputting the best 4 evidences for each claim in ER can retain more information and improve the performance of the downstream task. Hence, the best K value was decided to be 4.

3.3 Claim Classification

Evaluation metrics used for the CC task include accuracy and per class F1-score. Models displaying similar accuracy are favoured if they demonstrate a more balanced distribution of accurate predictions across all classes, better addressing the dataset's inherent class imbalance issue.

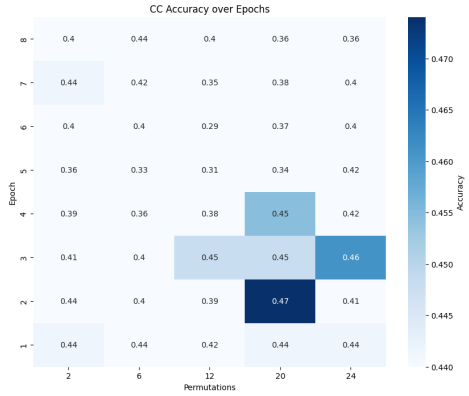


Figure 5: CC performance on S_{dev} over number of permutations sampled and training epochs.

Model	Dataset	F1-score
rNSMN	train	0.1970
rNSMN	development	0.1524
rNSMN	test	0.1576
rNSMN	final evaluation	0.1160
Baseline	development	0.3378

Table 3: The performance comparison of ER.

The model can easily overfit to the small S_{train} as the number of epochs increases, especially in conjunction with the effect of data augmentation that repeats the same claim several times. The two hyper-parameters tuned are thus the number of epochs, and the number of evidence permutations per claim. A grid search is conducted with 0.001 learning rate and 128 batch size, and model accuracies across different hyper-parameters combinations are displayed in figure 5. The minimum of the colour scale is set to the accuracy of majority voting (0.44). The optimal performance is achieved with 20 permutations over 2 epochs, striking a balance between learning from S_{train} and avoiding overfitting.

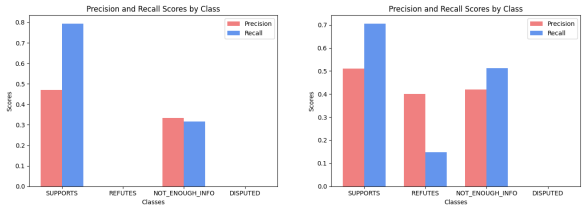
4 Results

4.1 Evidence Retrieval

From Table 3, our rNSMN achieved 0.15 F1-scores in the development and test set evaluation. It is worth noting that the train F1-score was only 0.197, suggesting model underfitting given the complexity of our model. In the final evaluation on Codalab, the retrieval F1-score dropped to 0.12 possibly due

Model	Dataset	Accuracy
cNSMN	train	0.5871
cNSMN	development	0.4740
cNSMN[0]	test	0.4342
cNSMN[0]	final evaluation	0.4545
Majority	development	0.4416

Table 4: The performance comparison of CC.



(a) n = 2 (b) n = 20

Figure 6: Per class CC precision and recall of S_{dev} over varying permutations (n) after 2 epochs.

to the inadequate size of the secret test set, introducing bias to the evaluation. Compared to the baseline, our rNSMN was significantly worse in the development set evaluation. This unsatisfactory performance illustrates the difficulty of retrieving related evidences from an enormous knowledge source given limited training data.

4.2 Claim Classification

From table 4, the model achieved 0.474 accuracy on S_{dev}. The test set performance are of a previous model without data augmentation, predicting all "SUPPORTS" using top 3 evidences. From comparing S_{dev} performances in Figure 6, data augmentation facilitates more accurate predictions distributed over the classes. Yet the result is lower than expected, with 0.696 accuracy achieved in the original paper (Nie et al., 2019). The primary reasons could be the small training dataset, and the approach is unable to distinguish the additional class introduced ('DISPUTED').

4.3 Combined Pipeline

The two-component system is evaluated using the harmonic mean of ER F1-score and CC accuracy. Trade-off exists when different numbers of evidence (K) are retrieved: F1-score is highest when K=3, while more evidence improves the CC performance. As discussed in Section 3.2, K is set to 4 to enhance the combined pipeline performance.

[0] Models without data augmentation and ran for 1 epoch only. The competition ended upon model adjustment.

5 Discussion

In our exploration of methods for evidence retrieval and claim verification, we have identified several key factors influencing the effectiveness of these processes. Our model's strength lies in its use of a neural semantic matching network, which enables a nuanced understanding of the semantic relationships between claims and evidence. This approach allows for more accurate and context-aware matching, which is crucial for effective verification.

However, we also observed several weaknesses in our model. One significant limitation is the comparatively smaller dataset we used for training, which restricted the model's ability to generalize across diverse scenarios. The original paper (Nie et al., 2019) achieved higher performance due to access to larger datasets and advanced embeddings like ELMo and GloVe, which enriched their model with more comprehensive linguistic knowledge. Additionally, the original study benefited from the availability of significantly more data, enhancing the robustness and accuracy of their results.

This large performance difference underscores the importance of data quantity and quality in training effective neural models. While our neural semantic matching network demonstrated promising results, further improvements could be realized by manually augmenting more extensive datasets such as using a 2D matrix (Lin et al., 2017) and advanced embeddings. Future work should focus on acquiring and utilizing such resources if possible to bridge the performance gap and enhance the overall effectiveness of our model.

6 Conclusion

Our research explored the application of a Neural Semantic Matching Network (NSMN) for evidence retrieval and claim verification within the domain of climate change statements. Our model demonstrated promising results in understanding semantic relationships between claims and evidence. However, limitations such as the smaller dataset and lack of advanced embeddings impacted the overall performance. Despite these challenges, our findings highlight the potential of neural networks in fact-checking tasks. Future work should focus on leveraging larger datasets and incorporating advanced embeddings to further enhance the model's performance and reliability in verifying climate-related claims.

7 Team contributions

In this project, all team members contributed evenly to the various stages of research and development. Each member participated in finding and attempting methods for data preprocessing, model implementation, and hyper-parameter tuning. Everyone took part in the experiments, analysis of results, and discussion of findings. The drafting and revision of the report were also collaborative efforts, ensuring that the final submission accurately reflected the collective work and insights of the entire team. This even distribution of tasks and responsibilities underscores our commitment to teamwork and shared learning.

References

Keskar, N. S., Mudigere, D., Nocedal, J., Smelyanskiy, M., & Tang, P. T. P. (2016). On large-batch training for deep learning: Generalization gap and sharp minima. *arXiv preprint arXiv:1609.04836*.

Lin, Z., Feng, M., Santos, C. N. d., Yu, M., Xiang, B., Zhou, B., & Bengio, Y. (2017). A structured self-attentive sentence embedding. *arXiv preprint arXiv:1703.03130*.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G., & Dean, J. (2013). Distributed representations of words and phrases and their compositionality. *Advances in neural information processing systems*, *26*, 3111–3119.

Nie, Y., Chen, H., & Bansal, M. (2019). Combining fact extraction and verification with neural semantic matching networks. *Proceedings of the AAAI conference on artificial intelligence*, *33*(01), 6859–6866.

Yan, G., Li, Y., Zhang, S., & Chen, Z. (2019). Data augmentation for deep learning of judgment documents. In Z. Cui, J.-S. Pan, S. Zhang, L. Xiao, & J. Yang (Eds.), *Intelligence science and big data engineering. big data and machine learning - 9th international conference, iscide 2019* (pp. 232–242). https://doi.org/10.1007/978-3-030-36204-1_19