BUT-FIT at SemEval-2019 Task 7: Determining the Rumour Stance with Pre-Trained Deep Bidirectional Transformers

Martin Fajcik, Lukas Burget, Pavel Smrz
Brno University of Technology, Faculty of Information Technology
612 66 Brno, Czech Republic
{ifajcik,burget,smrz}@fit.vutbr.cz

Abstract

This paper describes our system submitted to SemEval 2019 Task 7: RumourEval 2019: Determining Rumour Veracity and Support for Rumours, Subtask A (Gorrell et al., 2019). The challenge focused on classifying whether posts from Twitter and Reddit *support, deny, query,* or *comment* a hidden rumour, truthfulness of which is the topic of an underlying discussion thread. We formulate the problem as a stance classification, determining the rumour stance of a post with respect to the previous thread post and the source thread post. The recent BERT architecture was employed to build an end-to-end system which has reached the F1 score of 61.67 % on the provided test data. Without any hand-crafted feature, the system finished at the 2nd place in the competition, only 0.2 % behind the winner.

1 Introduction

Fighting false rumours at the internet is a tedious task. Sometimes, even understanding what an actual rumour is about may prove challenging. And only then one can actually judge its veracity with an appropriate evidence. The works of Ferreira and Vlachos (2016) and Enayet and El-Beltagy (2017) focused on predictions of rumour veracity in thread discussions. These works indicated that the veracity is correlated with discussion participants' stances towards the rumour. Following this, the SubTask A SemEval-2019 Task 7 consisted in classifying whether the stance of each post in a given Twitter or Reddit thread *supports*, *denies*, *queries* or *comments* a hidden rumour.

Potential applications of such a function are wide, ranging from an analysis of popular events (political discussions, academy awards, etc.) to quickly disproving fake news during disasters.

Stance classification (SC), in its traditional form, is concerned with determining the attitude of a source text towards a target text (Mohammad et al., 2016). It has been studied thoroughly for discussion threads (Walker et al., 2012; Hasan and Ng, 2013; Chuang and Hsieh, 2015). However, the objective of SubTask A SemEval-2019 Task 7 is to determine the stance to a hidden rumour which is not explicitly given (it can be often inferred from the source post of the discussion – the root of the tree-shaped discussion thread – as demonstrated in Figure 1). The competitors were asked to classify the stance of the source post itself too.

.@AP I demand you retract the lie that *people in #Ferguson were shouting "kill the police"*, local reporting has refuted your ugly racism

Figure 1: An example of discussion's source post denying the actual rumour which is present in the source post – annotated with red cursive

The provided dataset was collected from Twitter and Reddit tree-shaped discussions. Stance labels were obtained via crowdsourcing. The discussions deal with 9 recently popular topics – Sydney siege, Germanwings crash etc.

The approach followed in our work builds on recent advances in language representation models. We fine-tune a pre-trained end-to-end BERT (Bidirectional Encoder Representations from Transformers) model (Devlin et al., 2018), while using discussion's source post, target's previous post and the target post itself as inputs to determine the rumour stance of the target post. Our implementation is available online.[1]

2 Related Work

Previous SemEval competitions: In recent years, there were two SemEval competitions targeting the stance classification. The first one focused on the setting in which the actual rumour was provided (Mohammad et al., 2016). Organizers of

[1] www.github.com/MFajcik/RumourEval2019

SemEval-2016 Task 6 prepared a benchmarking system based on SVM using hand-made features and word embeddings from their previous system for sentiment analysis (Mohammad et al., 2013), outperforming all the challenge participants.

The second competition was the previous RumourEval won by a system based on word vectors, handcrafted features[2] and an LSTM (Hochreiter and Schmidhuber, 1997) summarizing information of the discussion's branches (Kochkina et al., 2017). Other submissions were either based on similar handcrafted features (Singh et al., 2017; Wang et al., 2017; Enayet and El-Beltagy, 2017), features based on sets of words for determining language cues such as Belief or Denial (Bahuleyan and Vechtomova, 2017), post-processing via rule-based heuristics after the feature-based classification (Srivastava et al., 2017), Convolutional Neural Networks (CNNs) with rules (Lozano et al., 2017), or CNNs that jointly learnt word embeddings (Chen et al., 2017).

End-to-end approaches: Augenstein et al. (2016) encode the target text by means of a bidirectional LSTM (BiLSTM), conditioned on the source text. The paper empirically shows that the conditioning on the source text really matters. Du et al. (2017) propose target augmented embeddings – embeddings concatenated with an average of source text embeddings – and apply them to compute an attention based on the weighted sum of target embeddings, previously transformed via a BiLSTM. Mohtarami et al. (2018) propose an architecture that encodes the source and the target text via an LSTM and a CNN separately and then uses a memory network together with a similarity matrix to capture the similarity between the source and the target text, and infers a fixed-size vector suitable for the stance prediction.

3 BUT-FIT's System Description

3.1 Pre-processing

We replace URLs and mentions with special tokens $\$URL\$$ and $\$mention\$$ using tweet-processor[3]. We use spaCy[4] to split each post into sentences and add the $[EOS]$ token to indicate termination of each sentence. We employ the tokenizer that comes with the Hugging Face PyTorch re-implementation of BERT[5]. The tokenizer lowercases the input and applies the WordPiece encoding (Wu et al., 2016) to split input words into most frequent n-grams present in the pre-training corpus, effectively representing text at the sub-word level while keeping a 30,000-token vocabulary only.

	S	**D**	**Q**	**C**	**Total**
train	925	378	395	3519	5217
in %	18	7	8	67	
dev	102	82	120	1181	1485
in %	7	6	8	80	
test	157	101	93	1476	1827
in %	9	6	5	81	

Table 1: Distribution of examples across classes in the training/development/test data set. The examples belong to 327/38/81 training/development/test tree-structured discussions.

3.2 Model

Following the recent trend in transfer learning from language models (LM), we employ the pre-trained BERT model. The model is first trained on the concatenation of BooksCorpus (800M words) (Zhu et al., 2015) and English Wikipedia (2,500M words) using the multi-task objective consisting of LM and machine comprehension (MC) sub-objectives. The LM objective aims at predicting the identity of 15% randomly masked tokens present in the input[6]. Given two sentences from the corpus, the MC objective is to classify whether the second sentence follows the first sentence in the corpus. The sentence is replaced randomly in half of the cases. During pre-training, the input consists of two documents, each represented by a sequence of tokens divided by the special $[SEP]$ token and preceded by the $[CLS]$ token used by the MC objective, i. e., $[CLS] document_1 [SEP] document_2 [SEP]$. Input tokens are represented by jointly learned token embeddings E_t, segment embeddings E_s, capturing whether the word belongs into $document_1$ or $document_2$, and positional embeddings E_p.

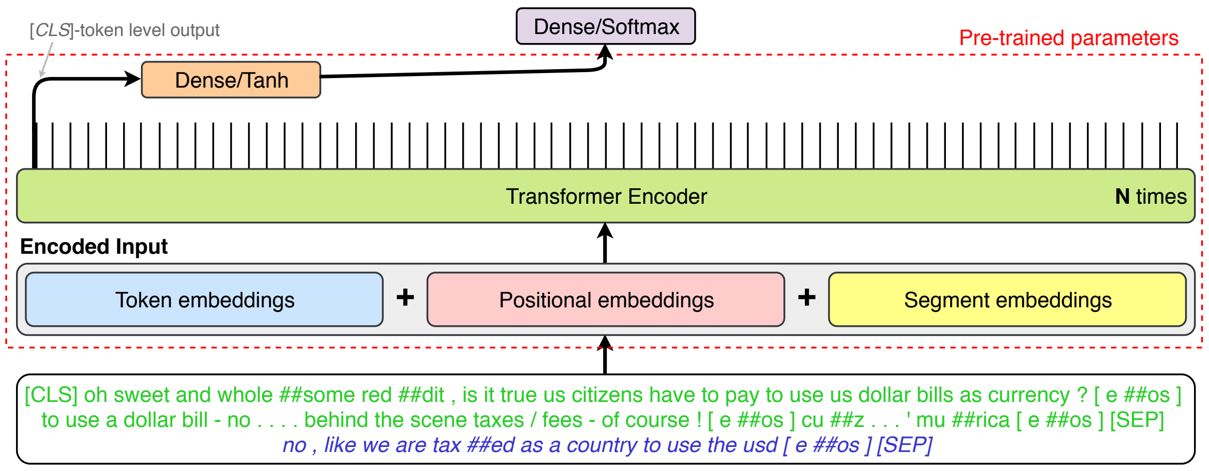
[2]The features included: a flag indicating whether a tweet is a source tweet of a conversation, the length of the tweet, an indicator of the presence of URLs and images, punctuation, the cosine distance to the source tweet and all other tweets in the conversation, the count of negation and swear words, and an average of word vectors corresponding to the tweet.

[3]`https://github.com/s/preprocessor`

[4]`https://spacy.io/`

[5]`https://github.com/huggingface/pytorch-pretrained-BERT`

[6]The explanation of token masking is simplified; details can be found in the original paper (Devlin et al., 2018).



[CLS]-token level output
Dense/Softmax
Pre-trained parameters
Dense/Tanh
Transformer Encoder
N times
Encoded Input
Token embeddings
+
Positional embeddings
+
Segment embeddings
[CLS] oh sweet and whole ##some red ##dit , is it true us citizens have to pay to use us dollar bills as currency ? [e ##os] to use a dollar bill - no . . . . behind the scene taxes / fees - of course ! [e ##os] cu ##z . . . ' mu ##rica [e ##os] [SEP]
no , like we are tax ##ed as a country to use the usd [e ##os] [SEP]

Figure 2: An architecture of BUT-FIT's system. The text segment containing $document_1$ is green, the segment containing $document_2$ (the target post) is blue. The input representation is obtained by summing input embedding matrices $E = E_t + E_s + E_p \in \mathbb{R}^{L \times d}$, L being the input length and d the input dimensionality. The input is passed N times via the transformer encoder. Finally, the $[CLS]$ token-level output is fed through two dense layers yielding the class prediction.

Our system follows the assumption that the stance of discussion's post depends only on itself, on the source thread post and on the previous thread post. Since the original input is composed of two documents, we experimented with various ways of encoding the input (see Section 5), ending up with just a concatenation of the source and the previous post as $document_1$ (left empty in case of the source post being the target post) and the target post as $document_2$. The discriminative fine-tuning of BERT is done using the $[CLS]$ token level output and passing it through two dense layers yielding posterior probabilities as depicted in Figure 2. A weighted cross-entropy loss is used to ensure a flat prior over the classes.

3.3 Ensembling

Before submission, we trained 100 models differing just by their learning rates. We experimented with 4 different fusion mechanisms in order to increase the F1 measure and compensate for overfitting:
The `TOP-N` fusion chooses 1 model randomly and adds it to the ensemble. Then, it randomly shuffles the rest of the models and tries to add them into the ensemble one at the time, while iteratively calculating ensemble's F1 by averaging the output probabilities, effectively approximating the Bayesian model averaging. If a model increases the total F1 score, the model is permanently added to the ensemble. The process is repeated until no further model improving the ensemble's F1 score can be found. This procedure resulted in a set of 17 best models.
The `EXC-N` fusion chooses all models into the ensemble and then iteratively drops one model at the time, starting from that which dropping results in the largest increase of the ensemble's F1. The process stops when dropping any other model cannot increase the F1 score. Using this approach, we ended up using 94 models.
The `TOP-N`$_s$ is analogous to the `TOP-N` fusion, but we average pre-softmax scores instead of output class probabilities.
The `OPT-F1` fusion aims at learning weights summing up to 1 for the weighted average of output probabilities from models selected via the procedure used in the `TOP-N` strategy. The weights are estimated using modified Powell's method from the SciPy package to maximize the F1 score on the development dataset.

4 Experimental Setup

We implemented our models in PyTorch, taking advantage of the Hugging Face re-implementation (see Footnote 5), with the *"BERT-large-uncased"* setting, pre-trained using 24 transformer layers, having the hidden unit size of $d = 1024$, 16 attention heads, and $335M$ parameters. When building the ensemble, we picked learning rates from the interval $[1e{-}6, 2e{-}6]$. Each epoch iterates over the dataset in an ordered manner, starting by the shortest sequence. We truncate sequences at maximum length $l = 200$ with a heuristic – firstly we truncate the $document_1$ to length $l/2$, if that is not enough, then we truncate the $document_2$ to

	#Θ	Acc_{test}	macro $\mathrm{F1}_{dev}$	macro $\mathrm{F1}_{test}$	$\mathrm{F1}_S$	$\mathrm{F1}_Q$	$\mathrm{F1}_D$	$\mathrm{F1}_C$
Branch-LSTM	453K	84.10	-	49.30	43.80	55.00	7.10	91.30
FeaturesNN	205K	82.84	$45.46 \pm 1e{-2}$	$44.55 \pm 2e{-2}$	40.29	40.12	17.69	80.43
BiLSTM+SelfAtt	28M	83.59	$47.55 \pm 6e{-3}$	$46.81 \pm 6e{-3}$	42.21	45.20	17.75	81.92
BERT_{base}	109M	84.67	$51.40 \pm 1e{-2}$	$53.39 \pm 3e{-2}$	43.49	59.88	18.42	90.36
$\mathrm{BERT}_{big-noprev}$	335M	84.33	$52.61 \pm 2e{-2}$	$52.91 \pm 4e{-2}$	42.37	55.17	24.44	90.15
$\mathrm{BERT}_{big-nosrc}$	335M	84.51	$53.72 \pm 2e{-2}$	$55.13 \pm 3e{-3}$	43.02	56.93	26.53	90.51
BERT_{big}	335M	84.08	$56.24 \pm 9e{-3}$	$56.70 \pm 3e{-2}$	44.29	57.07	35.02	90.41
BERT_{big} EXC-N*	-	85.50	58.63	60.28	48.89	62.80	37.50	91.94
BERT_{big} TOP-N*	-	85.22	62.58	60.67	48.25	62.86	39.74	91.83
BERT_{big} OPT-F1	-	85.39	62.68	61.27	48.03	62.26	42.77	92.01
BERT_{big} TOP-N_s	-	85.50	61.73	**61.67**	49.11	64.45	41.29	91.84

Table 2: Overview of the results. The values for each single model were obtained by averaging results of 11 models. We report the mean and the standard deviation in these cases. #Θ denotes the number of parameters. Columns $\mathrm{F1}_S$ to $\mathrm{F1}_C$ report individual F1 scores for each class. All ensemble models have the F1 score optimized on the development dataset. BiLSTM+SelfAtt contains 4.2M parameters, without pre-trained BERT embeddings. $\mathrm{BERT}_{big-nosrc}$ and $\mathrm{BERT}_{big-noprev}$ denote system instantiations with an empty source and an empty target post, respectively. Note that the accuracy is biased towards different training data priors as shown in Table 1. SemEval submissions are denoted by *.

the same size. We keep the batch size of 32 examples and keep other hyperparameters the same as in the BERT paper. We use the same Adam optimizer with the L2 weight decay of 0.01 and no warmup. We trained the model on the GeForce RTX 2080 Ti GPU.

5 Results and Discussion

We compare the developed system to three baselines. The first one is the branch-LSTM baseline provided by the task organizers[7] – inspired by the winning system of RumourEval 2017. The second baseline (FeaturesNN) is our reimplementation of the first baseline in PyTorch without the LSTM – posts are classified by means of a 2-layer network (ReLU/Softmax), using only the features defined in Footnote 2. In the third case (BiLSTM+SelfAtt), we use the same input representation as in our submitted model but replace the BERT by an 1-layer BiLSTM network followed by a self-attention and a softmax layer, inspired by Lin et al. (2017).

The results are shown in Table 2. BERT models had to cope with a high variance during the training. This might be caused by the problem difficulty, the relatively small number of training examples, or the complexity of the models. To deal with the problem, we decided to discard all models with F1 scores of less than 55 on the development dataset and we averaged the output class probability distributions when ensembling. Our initial experiments used sequences up to the length of 512, but we found no difference when truncating them down to 200.

What features were not helpful: We tried adding a number of other features, including those indicating positive, neutral, or negative sentiment, and all the features used by the FeaturesNN baseline. We also tried adding jointly learned POS, NER, and dependency tag embeddings, as well as the third segment embeddings[8]. We also experimented with an explicit $[SEP]$ token to separate the source and the previous post in the BERT input. However, none of the mentioned changes led to a statistically significant improvement.

6 Conclusions and Future Directions

The system presented in this paper achieved the macro F1 score of 61.67, improving the baseline by 12.37%, while using only the source post of discussion, the previous post and the target post to classify the target post's stance to a rumour.

A detailed analysis of the provided data shows that the employed information sources are not sufficient to correctly classify some examples. Our future work will focus on extending the system by a relevance scoring component. To preserve the context, it will evaluate all posts in a given discussion thread and pick up the most relevant ones according to defined criteria.

[7]http://tinyurl.com/y4p5ygn7

[8]We tried adding the learned representations to the input the same way the segment/positional embeddings are added.

Acknowledgments

This work was supported by the Czech Ministry of Education, Youth and Sports, subprogram INTER-COST, project code: LTC18054.

References

Isabelle Augenstein, Tim Rocktäschel, Andreas Vlachos, and Kalina Bontcheva. 2016. Stance detection with bidirectional conditional encoding. *arXiv preprint arXiv:1606.05464*.

Hareesh Bahuleyan and Olga Vechtomova. 2017. Uwaterloo at semeval-2017 task 8: Detecting stance towards rumours with topic independent features. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 461–464.

Yi-Chin Chen, Zhao-Yang Liu, and Hung-Yu Kao. 2017. Ikm at semeval-2017 task 8: Convolutional neural networks for stance detection and rumor verification. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 465–469.

Ju-han Chuang and Shukai Hsieh. 2015. Stance classification on ptt comments. In *29th Pacific Asia Conference on Language, Information and Computation Proceedings of PACLIC 2015: Poster Papers*, page 27.

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

Jiachen Du, Ruifeng Xu, Yulan He, and Lin Gui. 2017. Stance classification with target-specific neural attention networks. International Joint Conferences on Artificial Intelligence.

Omar Enayet and Samhaa R El-Beltagy. 2017. Niletmrg at semeval-2017 task 8: Determining rumour and veracity support for rumours on twitter. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 470–474.

William Ferreira and Andreas Vlachos. 2016. Emergent: a novel data-set for stance classification. In *Proceedings of the 2016 conference of the North American chapter of the association for computational linguistics: Human language technologies*, pages 1163–1168.

Genevieve Gorrell, Kalina Bontcheva, Leon Derczynski, Elena Kochkina, Maria Liakata, and Arkaitz Zubiaga. 2019. SemEval-2019 Task 7: RumourEval: Determining rumour veracity and support for rumours. In *Proceedings of SemEval*. ACL.

Kazi Saidul Hasan and Vincent Ng. 2013. Stance classification of ideological debates: Data, models, features, and constraints. In *Proceedings of the Sixth International Joint Conference on Natural Language Processing*, pages 1348–1356.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Elena Kochkina, Maria Liakata, and Isabelle Augenstein. 2017. Turing at semeval-2017 task 8: Sequential approach to rumour stance classification with branch-lstm. *arXiv preprint arXiv:1704.07221*.

Zhouhan Lin, Minwei Feng, Cicero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, and Yoshua Bengio. 2017. A structured self-attentive sentence embedding. *arXiv preprint arXiv:1703.03130*.

Marianela García Lozano, Hanna Lilja, Edward Tjörnhammar, and Maja Karasalo. 2017. Mama edha at semeval-2017 task 8: Stance classification with cnn and rules. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 481–485.

Saif Mohammad, Svetlana Kiritchenko, Parinaz Sobhani, Xiaodan Zhu, and Colin Cherry. 2016. Semeval-2016 task 6: Detecting stance in tweets. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 31–41.

Saif M. Mohammad, Svetlana Kiritchenko, and Xiaodan Zhu. 2013. Nrc-canada: Building the state-of-the-art in sentiment analysis of tweets. In *Proceedings of the seventh international workshop on Semantic Evaluation Exercises (SemEval-2013)*, Atlanta, Georgia, USA.

Mitra Mohtarami, Ramy Baly, James Glass, Preslav Nakov, Lluís Màrquez, and Alessandro Moschitti. 2018. Automatic stance detection using end-to-end memory networks. *arXiv preprint arXiv:1804.07581*.

Vikram Singh, Sunny Narayan, Md Shad Akhtar, Asif Ekbal, and Pushpak Bhattacharyya. 2017. Iitp at semeval-2017 task 8: A supervised approach for rumour evaluation. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 497–501.

Ankit Srivastava, Georg Rehm, and Julian Moreno Schneider. 2017. Dfki-dkt at semeval-2017 task 8: Rumour detection and classification using cascading heuristics. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 486–490.

Marilyn A Walker, Pranav Anand, Robert Abbott, and Ricky Grant. 2012. Stance classification using dialogic properties of persuasion. In *Proceedings of the 2012 conference of the North American chapter of the association for computational linguistics: Human language technologies*, pages 592–596. Association for Computational Linguistics.

Feixiang Wang, Man Lan, and Yuanbin Wu. 2017. Ecnu at semeval-2017 task 8: Rumour evaluation using effective features and supervised ensemble models. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 491–496.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*.

Yukun Zhu, Ryan Kiros, Rich Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *Proceedings of the IEEE international conference on computer vision*, pages 19–27.

A Supplemental Material

A.1 Dataset Insights

The dataset contains a whole tree structure and metadata for each discussion from Twitter and Reddit. The nature of the data differs across the sources (for example, the Reddit subset includes upvotes).

When analysing the data, we spotted several anomalies:

- 12 data points do not contain any text. According to the task organizers, they were deleted by users at the time of data download and been left in the data not to break the conversational structure.

- The query stance of some examples taken from subreddit DebunkThis[9] is dependent on the domain knowledge. The class of some examples is ambiguous; they should be probably labelled by multiple classes.

A.1.1 Domain knowledge dependency

Examples from subreddit DebunkThis have all the same format ”Debunk this: [statement]”, e.g. *”Debunk this: Nicotine isn't really bad for you, and it's the other substances that makes tobacco so harmful.”*. All these examples are labelled as queries.

[9]https://www.reddit.com/r/DebunkThis/

A.1.2 Class ambiguity

The source/previous post *”This is crazy! #CapeTown #capestorm #weatherforecast https://t.co/3bcKOKrCJB”* and the target post *”@RyGuySA Oh my gosh! Is that not a tornado?! Cause wow, It almost looks like one!”*, labelled as a comment in the dataset, might be seen as a query as well.

A.2 Additional Introspection

Figures 3, 4, 5, and 6 demonstrate attention matrices A, derived from the multi-head attention defined as:

$$A = \frac{QK^\top}{\sqrt{d_k}}, \tag{1}$$

where $Q, K \in \mathbb{R}^{L \times d_k}$ are the matrices containing query/value vectors and d_k is the key/value dimension. The insights are selected from the heads at the first layer of the transformer encoder.

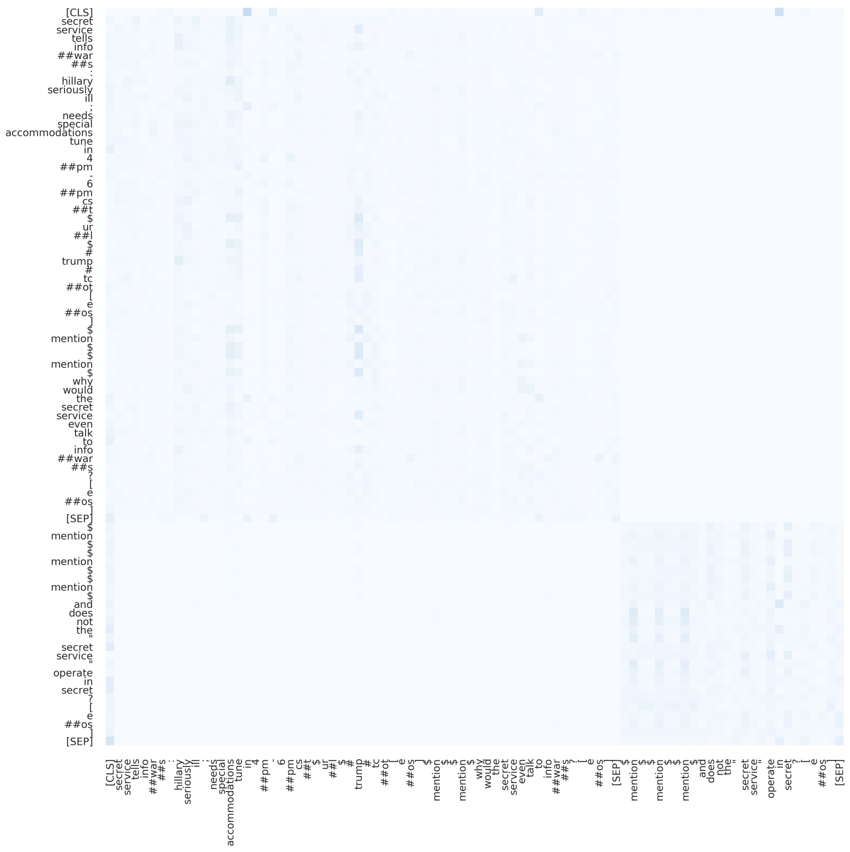


Figure 3: Intra-segment attention – the attention is made only between the subword units from the same segment.

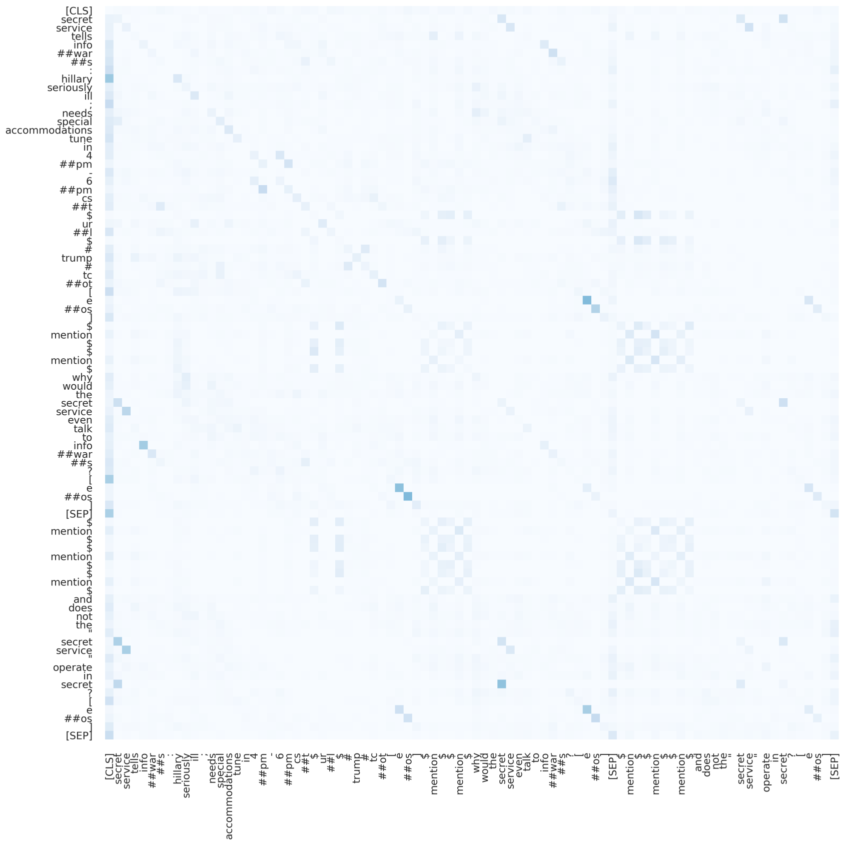


Figure 4: Attention matrix capturing the subword similarity.

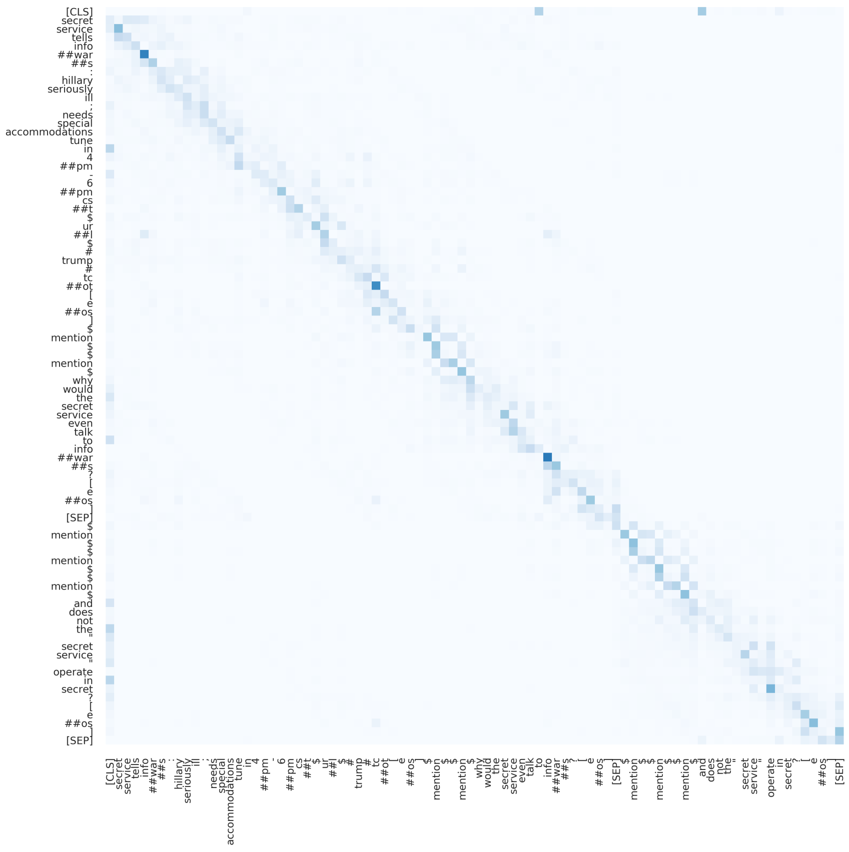


Figure 5: ’Soft’ local context aggregation.

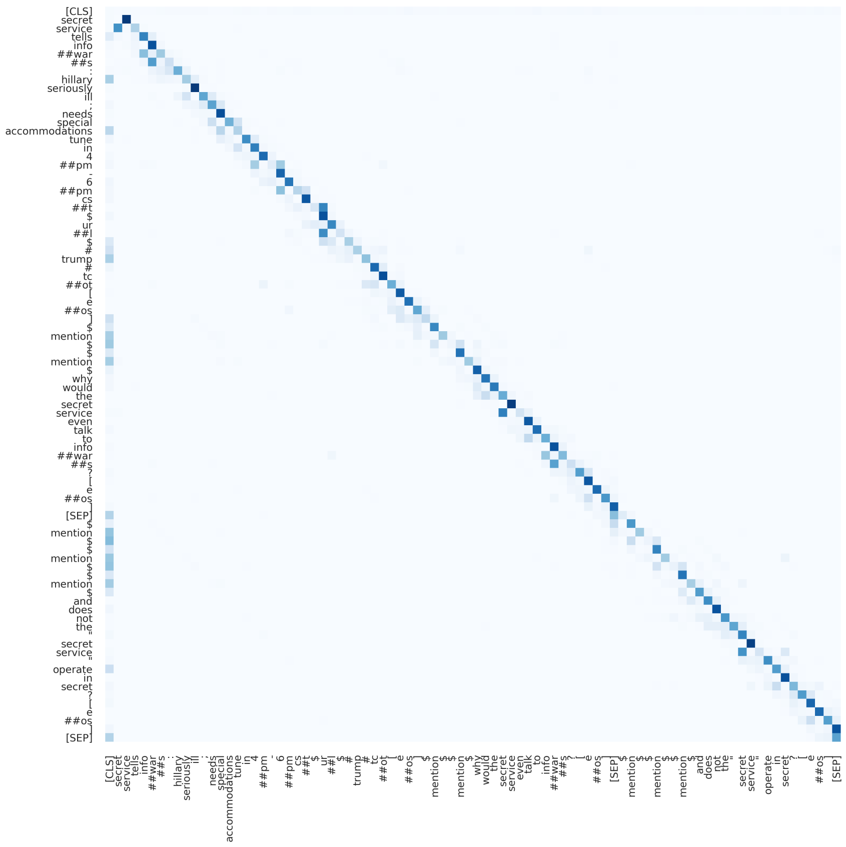


Figure 6: ’Hard’ local context aggregation – the signal is mostly sent further to another transformer encoder layer.