Enhancing Tiny Transformers: Strategies for Improving Question Answering with NER and POS

Mihai-Bogdan Bîndilă
no. 3264424
m.bindila@student.utwente.nl

Cristina-Nicoleta Racoviță
no. 3264440
c.racovita@student.utwente.nl

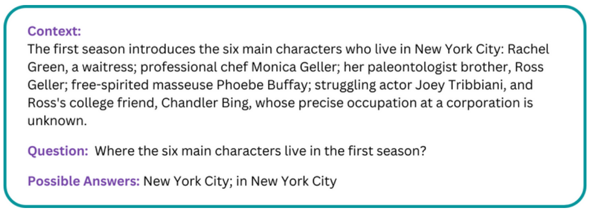
Abstract

Question Answering (QA) is a widely studied natural language processing (NLP) task, and recent advancements have often focused on employing larger and more complex models. In this paper, we explore the potential enhancement of a smaller transformer model, namely **TinyBERT**, by gradually incorporating **named-entities (NE) and part-of-speech (POS)** information. Our experiments cover two granularity levels (4-class/18-class) and multiple candidate tags for named-entities. We made no changes to the base model architecture and we only used context augmentation techniques (respectively, **in-between and concatenation** method) to integrate these supplementary features into the context and answer.

This enhanced TinyBERT was used to **extract factoid answers** on the Stanford Question Answering Dataset **(SQUAD)** to assess its **reading comprehension capabilities**. To achieve this goal, we evaluated the model using the **Exact Match** and the **F1** metrics. Our results show consistent improvements across all cases, with both metrics, reaching notable increases. Of the techniques considered, the most effective was the addition of multiple 4-class named-entities tags for spans of text, which resulted in a significant improvement of **3.07% for Exact Match** and **2.03% for F1**.

1 Introduction

Question Answering is a popular task in natural language processing with multiple applications ranging from interacting with search engines to talking with virtual assistants (Jurafsky and Martin, 2008). QA systems are supposed to answer questions with or without a specific context (Open or Closed QA). Based on the answer type, a QA model can be extractive (the answer is a span of words extracted from a context) or generative (the answer is generated by the model). Extractive QA focuses the most on factoid questions (simple facts expressed in short texts) (Jurafsky and Martin, 2008), (Pandya and Bhatt, 2021). In figure 1 we can see an example of a factoid question in which the QA model should extract the answer from a Wikipedia text.



Context:
The first season introduces the six main characters who live in New York City: Rachel Green, a waitress; professional chef Monica Geller; her paleontologist brother, Ross Geller; free-spirited masseuse Phoebe Buffay; struggling actor Joey Tribbiani, and Ross's college friend, Chandler Bing, whose precise occupation at a corporation is unknown.

Question: Where the six main characters live in the first season?

Possible Answers: New York City; in New York City

Figure 1: Factoid question example

Enhancing QA systems using named-entities or part-of-speech tags is widely studied in the literature and many strategies have already been developed, for example, using multiple candidates for a tag or external feature embedding integration. More of these will be explored in greater detail in Section 2 of the document.

Named-Entities Recognition (NER) is a type of tagging in which every word has a specific tag assigned by a model (such as Location, Person, Organization, etc). Part of speech tagging is a similar task where the only difference is the tag nature - the model labels every word with a part of speech (such as Noun, Verb, Preposition, etc). Figure 2 shows a more detailed example of tagging a sentence.

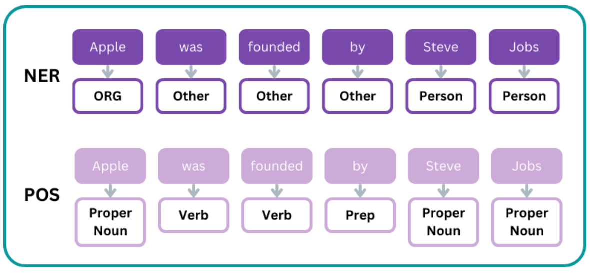


Figure 2: NE and POS tagging

1.1 Problem Statement

The latest proposed solutions for QA tasks have implied larger or more complex models, and we have the following research questions: *Can we improve the performance of a smaller transformer model by using NER and POS without changing its base architecture? If so, which is the most effective strategy?*

With this question in mind, we chose the most compact instance among the family of BERT-like models, namely **TinyBERT** (Turc et al., 2019), and we focused on extractive QA with factoid answers.

2 Related Work

Part-of-speech tagging is a common task used to improve QA systems. In (Creus-Costa and Hwang, 2017), the authors developed a bi-LSTM (Bidirectional Long-Short Term Memory) model, for which the POS tags are provided as a concatenated one-hot vector for each embedding. The idea behind this technique is that for most questions, we can expect nouns as the part-of-speech answer (Who? Where? What? Every question for this type can be answered with a noun).

After **Large Language Models** (LLM) have become so popular, QA tasks were mainly solved by fine-tuning the existing transformers (BERT (Devlin et al., 2018), TinyBERT (Turc et al., 2019), and Albert (Lan et al., 2019)).

The authors of BERT conducted two types of experiments by fine-tuning the model on SQuAD (Rajpurkar et al., 2016). They tried directly fine-tuning BERT or doing this after learning the TriviaQA dataset. Additionally, they tested ensembles of BERT. The best results in terms of F1 score, 91.1%, and 92.2%, were obtained when the model or the ensemble was initially fine-tuned on TriviaQA.

In (Turc et al., 2019), 24 compact variations of BERT were pre-trained and then trained again to learn the labels predicted by BERT in a knowledge distillation process. This process is performed using an unlabeled transfer dataset with the same distribution as the fine-tuning dataset. The teacher model learns the data, and the student receives as ground truth labels the one predicted by the teacher. They demonstrated that this technique called pre-trained distillation helps the student model perform significantly better, being a complementary learning task for the fine-tuning phase on the target dataset.

A well-known way to improve machine learning and deep learning models is using external features by adding them as raw tokens in the original texts (Molla Aliod et al., 2009) or fusing their embeddings with the context embeddings (Xu Gezheng, 2021).

Some external features are used to enhance these fine-tuned transformers, such as named-entity information. In (Molla Aliod et al., 2009), multiple candidates for a named-entity tag are used because some sentences can be ambiguous, and mistakes can appear when predicting the entity for a specific word. For example: **"Apple is opening a new store in the Big Apple."** is an ambiguous sentence since the first and the last "Apple" refer to two different things (company versus location). These tags were inserted in the original text without changing the architecture of the base machine learning model.

Fusion solutions have one main idea implemented in several distinct ways All of them have two types of embeddings: external feature embeddings and word/text embeddings. In (Xu Gezheng, 2021), the feature fusion is developed using a neural network with three sub-layers: one for extending the feature embeddings to the same dimension as word embeddings, the second one fuses these previous results in one representation, and the last one catches the inter-dependency between the feature-enriched tokens.

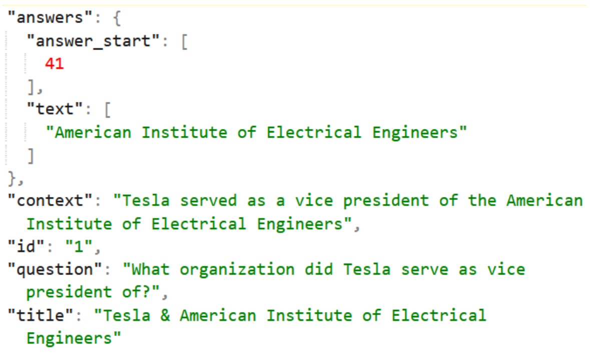
3 Data

Standford Question Answering Dataset (SQUAD) (Rajpurkar et al., 2016) is a reading comprehension dataset with 100,000+ questions and answers created by crowdworkers based on Wikipedia articles. The dataset is split into train, validation, and test set. We have access only to train data, which has 87599 examples, and validation data composed of 10570 instances. All of them have the JSON structure shown in Figure 3, with the mention that the **title** was not used in our case.

Interestingly, some questions from the validation set have multiple correct answers or no golden answer. In the last-mentioned case, the model should predict an empty string as an accepted answer.

4 Method

Our work focuses on improving the performance of compact Bidirectional Encoder Representations from the Transformers (BERT) model by enriching



```
"answers": {
  "answer_start": [
    41
  ],
  "text": [
    "American Institute of Electrical Engineers"
  ]
},
"context": "Tesla served as a vice president of the American
  Institute of Electrical Engineers",
"id": "1",
"question": "What organization did Tesla serve as vice
  president of?",
"title": "Tesla & American Institute of Electrical
  Engineers"
```

Figure 3: SQuAD data structure

the input context with POS and NE tags in various ways. To achieve that, we use the smallest pre-trained instance of the BERT family of models, namely TinyBERT (Turc et al., 2019), that was ported to the Hugging Face library during the work presented in (Bhargava et al., 2021). This model was pre-trained on English Wikipedia and BooksCorpus datasets, together having 3.3 billion words. We chose this model because, in the context of the QA task we have not found studies about improving the performance of compact transformers by augmenting the input data with external tags.

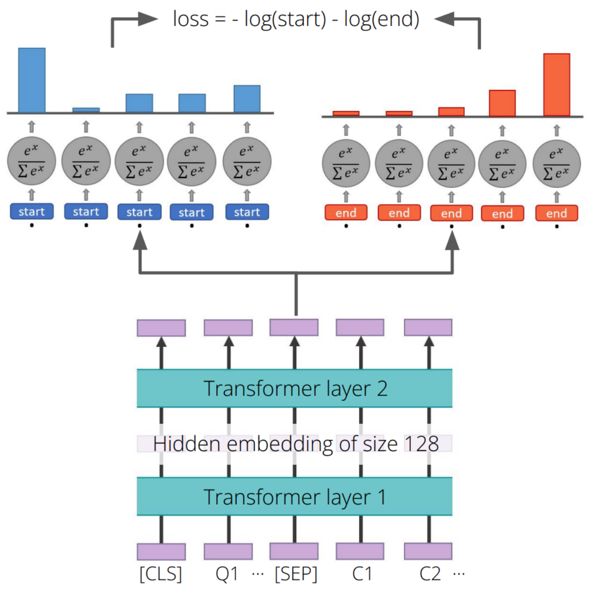


Figure 4: TinyBERT architecture

In terms of architecture, it has a hidden embedding with a size of 128 values and is composed of two transformer layers, each having two self-attention heads. In total, TinyBERT has only 4.4 million parameters, 25 times less than BERT base (Devlin et al., 2018). Its architecture is illustrated in Figure 4.

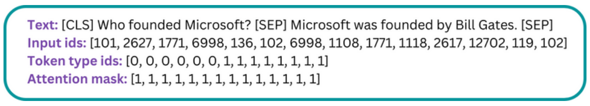
4.1 Fine-tuning

Fine-tuning this model requires extensive data preprocessing and results post-processing steps.

4.1.1 Data Preprocessing

The first step is represented by tokenizing the questions and contexts with an uncased tokenizer that does not remove stop words or punctuation, followed by a division into blocks of a fixed length. Each block contains the entire question and a fraction of the context separated by the *[SEP]* token. The chunks are extracted using a moving window with a stride of 128 to avoid splitting the context in the middle of an answer.

Based on the tokenizer's output, we have to extract for each input the start and end position of the answer. If the context fraction does not contain the entire answer, the start and end positions are zero, otherwise, we store the positions relative to the current chunk. As can be seen in Figure 5, the model takes as inputs a list of token IDs, a list of token types IDs, where zero denotes a question token and one a context token, and an attention mask where zero entries correspond to padding tokens.



Text: [CLS] Who founded Microsoft? [SEP] Microsoft was founded by Bill Gates. [SEP]
Input ids: [101, 2627, 1771, 6998, 136, 102, 6998, 1108, 1771, 1118, 2617, 12702, 119, 102]
Token type ids: [0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1]
Attention mask: [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]

Figure 5: Example of input for a question-context pair

When it comes to predictions, the model outputs logits for the start and end positions of the answer. Tokens outside the context should be masked out and take the start-end token pairs that have the highest sum of logarithms of logits. A principal condition for these candidate answers is having a length between 1 and the maximum set length.

4.1.2 Hyperparameters

Below are shown the data processing hyperparameters used for model fine-tuning and hyperparameter optimization:

- **chunk length:** 384 tokens
- **stride:** 128 tokens
- **number of candidate answers:** 1024
- **max answer length:** 128 characters

The values of chunk length and stride are taken from the original paper of TinyBERT (Turc et al., 2019), and the max answer length is set considering that 99% of answers have less than 121 characters.

As in (Devlin et al., 2018), the training objective for the question-answering task at hand is the sum of the log-likelihoods of the correct start and end positions (Figure 4). We fine-tuned TinyBERT for three epochs and minimized the loss function through backpropagation. We use mini-batch gradient descent, with a batch size of eight, optimized with AdamW, with a linear weight decay from 2e-5. We chose this learning rate to be consistent with the one used during the pre-training process.

4.2 POS Tagging and NER

	4-Class NER	18-Class NER	POS
Model	ner-fast	ner-ontonotes-fast	pos
F1 score	92.75%	89.27%	98.19%
Training Dataset	Conll-03	Ontonotes	Ontonotes

Table 1: Flair models

This task was implemented using **Flair Framework** (Akbik et al., 2019), which uses models based on Flair embeddings and Long-Short Term Memory - Conditional Random Fields (LSTM-CRF) architecture. We chose the models presented in Table 1 because of their high performance and low prediction time. Also, it was important for consistency to use a tool that allows us to retrieve the whole set of tag candidates together with their probabilities. Another advantage is that this library can be used for both tagging strategies.

We applied those labeling methods in every context. For the NER approach, we extracted more than one tag, storing only the labels that have more than 10% probability (the rest being noise).

The POS model predicts around 41 possible tags (such as DT - Determiner; JJ - Adjective; MD - Modal; NNP - Proper noun, singular; etc), which gave us a great level of granularity. When it comes to NER models, we use two based on the number of predicted classes. The 4-tag NER model predicts the following labels:

- **PER**: person name - Elon Musk
- **LOC**: location name - Enschede, London
- **ORG**: organization name - Google, Microsoft, Amazon
- **MISC**: other name - iPhone, Google Search, World Cup

The other NER model predicts 18 possible tags that encapsulate more entity types such as NORP: national origin or religious/political group; GPE: geo-political entity; LANGUAGE; DATE; TIME; PERCENT; MONEY; QUANTITY; EVENT; etc. We chose two levels of detail for named-entity predictions to test how this characteristic impacts the performance.

5 Experiments and Results

5.1 Metrics

The most used metrics for QA tasks are Exact Match (Equation 1) and F1 score (Equation 2) (Jurafsky and Martin, 2008).

The first one computes the percentage of predicted answers that match the gold answer (the correct answer). For example, if the golden answer is **"New York"** and the inferred answer is **"New Jersey"**, even though one word matches between the two responses, the Exact Match of that input is 0. So, only if the answer matches all the words exactly, the metric result is 1.

The F1 score calculates the average of tokens that overlap between predicted and gold answers. This metric treats the prediction and the gold answer as a bag of words, and computes the F1 score for each question, then returns the average score of all questions (Jurafsky and Martin, 2008).

$$\text{Exact Match} = \frac{\text{No of Predicted golden answers}}{\text{No of predicted answers}} \tag{1}$$

$$F1 = \frac{2 * \text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}} \tag{2}$$

5.2 Model Optimisation

Pushing the performance boundaries of a model implies the optimization stage. We completed it before conducting any experiments so that, throughout the study, we used the optimum found hyperparameters. Considering that we did not have access to the test dataset, we only used the train and validation splits described in Section 3. To save time and to be consistent, we use a fixed random seed and perform a single training round for the optimization setting.

The search space has two dimensions: batch size and learning rate. Below are written the values for each dimension:

- **batch sizes:** 1, 8, 16, 32, 64, 128

- **learning rates:** 2e-5, 3e-5, 5e-5, 1e-4, 3e-4

The set of values is chosen concerning other studies that fine-tuned a model from the BERT family on SQuAD (Turc et al., 2019), (Devlin et al., 2018), and (Lan et al., 2019).

Finally, the best results, **F1 score - 52.50%** and **Exact Match - 39.15%**, are obtained with a batch size of 8 and a 3e-4 learning rate. The worst result reached even an F1 score of 31.85% and an Exact Match of 19.87% while using a batch size of 128 and a learning rate of 3e-5.

When comparing this optimized baseline with the performance obtained while using text augmentation strategies, the metrics are computed again with several random seeds and reported using confidence intervals.

5.3 Context Augmentation Strategies

Based on the techniques seen during the literature review, we tried multiple approaches to find the best for our dataset and model. These methods were mainly tested in the context of classical NLP approaches and we want to bridge the gap by combining them with TinyBERT.

Our covered scenarios are presented in depth in the following paragraphs. We place the POS tags after the context words or after every word from the context. In the case of NE tags, we encapsulate between groups of context words. The NER experiments include taking multiple candidates and different levels of granularity.

5.3.1 Data Preprocessing

Except for the case when we add the POS tags at the end, for every implemented strategy, the answer start position and text of the example should be updated before training the model with them. This happens because the context changes when adding the new tags; therefore, the answer alters, and we need to update them to make sense of the new context.

Let's consider as an example the data object from Figure 3, the new context after adding the new tags (we consider that we add new in-between POS tags) is: *"Tesla [NNP] served [VBD] as [IN] a [DT] vice [NN] president [NN] of [IN] the [DT] American [NNP] Institute [NNP] of [IN] Electrical [NNP] Engineers [NNPS]"*; therefore, the new answer_start should be *82* and the new answer text should be *"American [NN] Institute [NN] of [IN] Electrical [NN] Engineers [NNS]"*.

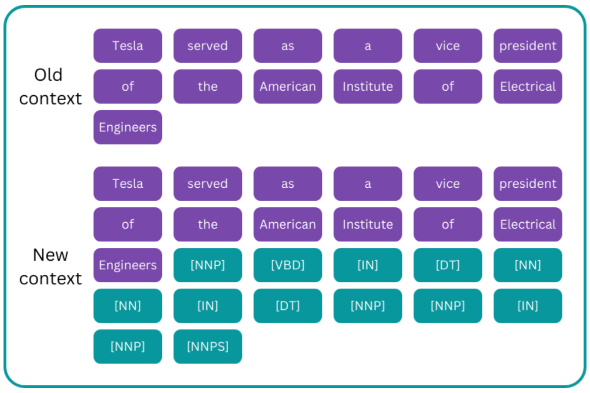


Figure 6: POS tags placing after context

5.3.2 POS after Context

We combine the approach presented in (Zhu and Cheung, 2021), where additional tags are appended after the context and the idea from (Creus-Costa and Hwang, 2017) where POS tags are used for the QA task. POS after is the simplest method we implemented because no additional data preprocessing is needed before the model training. We put all the POS tags after the context, as can be observed in Figure 6; thus, the answer text and position are not influenced by the concatenation. The main drawback of this method is observed when the question-context pair is longer than the set maximum chunk length, leading to a loss of information because the relation between distant tokens and tags cannot be modeled.

5.3.3 POS in-between Context

Considering the issue of the aforementioned data augmentation, we also place POS tags after every word of the context, in the same way it is presented in Figure 7. In this manner, the adjacent placement enables a direct link between tokens and their part of speech.

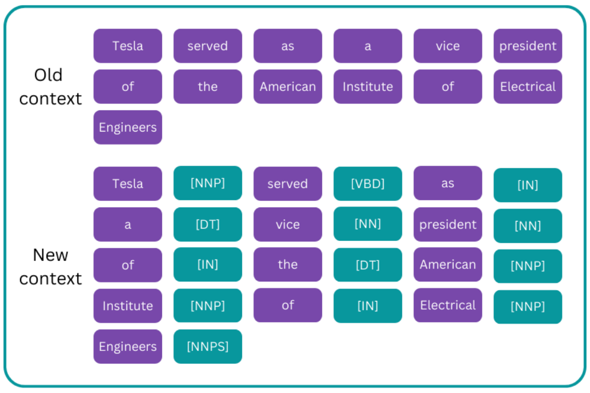


Figure 7: POS Tags placing in-between context words

5.3.4 NER Encapsulated in-between Context

This is the most complex strategy we developed. Augmenting the context by marking spans of text with one or more named-entities is a well-known technique proven in the case of a classical machine learning approach (Molla Aliod et al., 2009). We want to find out their implications in the context of language models. Therefore, we define four cases based on the granularity level and the number of tag candidates: *4-class tags and one label candidate, 18-class tags and one label candidate, 4-class tags and up to four label candidates, 18-class tags and up to four label candidates*.

	2 tags	3 or 4 tags
4 classes	9.59%	0.19%
18 classes	6.76%	0.07%

Table 2: Percentage of tokens with multiple tags

After the tagging process with four classes of tags, around 15% of tokens all tokens (including symbols) have one. In the case of 18-class NE, 22.5% of the words have an associated label. Table 2 shows the proportion of tokens with more than one tag out of the tagged ones.

The augmentation method is different than the ones used in the previous paragraphs. The main difference is that an NE can be assigned to a word or a group of words, and not all tokens should have an NE. This method adds tags before and after single or groups of words. This strategy ensures a positional relationship between a span of text and the corresponding labels.

For example, for **American Institute** the tags will be placed like this: **<ORG> American Institute </ORG>**, and if we use multiple tags we insert them in this way: **<PER><LOC>Washington</LOC></PER>**.

5.4 Results

We tested the implication of each augmentation strategy by fine-tuning the model with the optimum model hyperparameters found in Subsection 5.2 on each variation of the dataset. The only modifications of data preprocessing hyperparameters are related to the maximum answer length and stride, which was increased to 256 due to longer answers. The maximum accepted answer was set based on the size of the longest answer from the training set.

The F1 and Exact Match metrics are reported with 95% confidence intervals for the validation set after training and evaluating each model 10 times with different random seeds. To have an unbiased estimation of the computed performance scores, the metrics are measured after removing from the answer the POS and NE tags introduced in training.

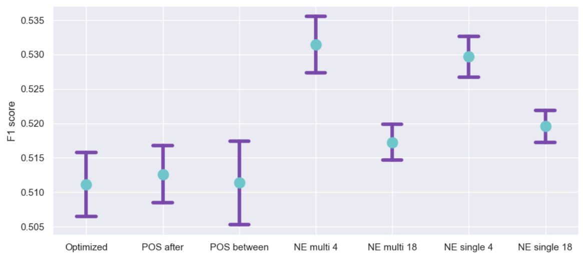


Figure 8: F1 scores obtained for performed experiments

Before optimizing the model, it had an Exact Match of 0.3078 ± 0.0084 and an F1 score of 0.4305 ± 0.0092. A significant performance leap is recorded after optimizing the hyperparameters, as can be observed in Figure 8 and Figure 9.

The augmentation methods have an identical ranking in terms of both measured metrics.

On average, adding POS tags in the context improves the model only marginally. However, this kind of tags added between tokens makes the model very variable, aspect observable due to the wide confidence interval.

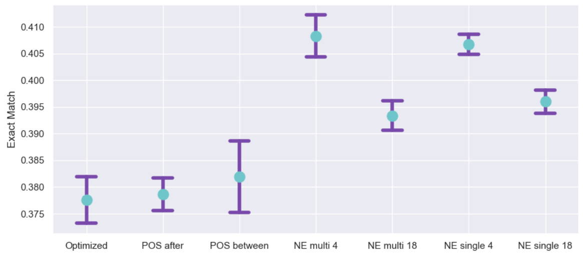


Figure 9: Exact match values of each experiment

When it comes to metrics recorded when NE tags are included in the context, we can notice a significant improvement. The hierarchy is as follows: multiple 4-class tags, single 4-class tags, single 18-class tags, and last but not least, multiple 18-class tags. Compared with the optimized model without data augmentation, the best method recorded an **F1 score improvement of 2.03%** and an **Exact Match increase of 3.07%**.

6 Discussion and Conclusion

6.1 Limitations and Challenges

The main drawback of the proposed methods is related to the significantly increased number of tokens. Even though we do not expand the number of model parameters, augmenting the input text implies a longer training time. Moreover, adding additional token dependencies alters the initial context and reduces the maximum accepted text length. After analyzing our results, we can observe a negative correlation between the percentage of added tags in validation answers and measured metrics. These percentages shown in Figure 3 illustrate that too many tags can become noise for the model when retrieving the correct answer.

	one tag	**multiple tags**
4 classes	19.35%	20.11%
18 classes	28.70%	29.50%

Table 3: Percentage of tags in validation answers

During the development phase, we faced many challenges, mainly related to the dataset augmentation. We found many questions with original incorrect answers, and we had to fix this issue by removing answers that did not appear in the context. In general, the process of adding tags to context and answer was non-trivial due to the discrepancy between the tokenizers of TinyBERT and Flair.

6.2 Ethical Concerns

From an ethical perspective, this type of data augmentation can be used to promote biased answers by surrounding only them with named-entities. This augmentation can increase the chances of that span of text being predicted as the correct answer. introduce in the text all tags predicted by an unbiased sequence tagger.

6.3 Conclusion

In this study, we bridged the gap between well-known text augmentation techniques for extractive question answering with factoid answers and state-of-the-art compact transformers. We experimented with 6 different enhancement methods of the input text with POS and NE tags and analyzed their implications. Our main contribution is represented by demonstrating their effectiveness even for TinyBERT, which has a compressed architecture. Moreover, we found the best technique, namely inserting NE tags with 4 classes before and after recognized entities.

6.4 Future Improvements

This work can be improved in several ways. On the one hand, the context can be enriched with constituents, which have been proven to be effective for this type of dataset (Rajpurkar et al., 2016). On the other hand, one can reduce the number of named-entity tags by filtering them based on the question type. Another avenue is related to fusion methods (Xu Gezheng, 2021) that do not modify the input text, but increase the model complexity.

References

A. Akbik, Tanja Bergmann, Duncan A. J. Blythe, Kashif Rasul, Stefan Schweter, and Roland Vollgraf. 2019. Flair: An easy-to-use framework for state-of-the-art nlp. In *North American Chapter of the Association for Computational Linguistics*.

Prajjwal Bhargava, Aleksandr Drozd, and Anna Rogers. 2021. Generalization in nli: Ways (not) to go beyond simple heuristics.

Joan Creus-Costa and Philip Hwang. 2017. Question answering on the squad dataset with part-of-speech tagging.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805.

Dan Jurafsky and James H. Martin. 2008. Speech and language processing, 2nd edition.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2019. Albert: A lite bert for self-supervised learning of language representations. *arXiv preprint arXiv:1909.11942*.

Diego Molla Aliod, Menno Zaanen, and Daniel Smith. 2009. Named entity recognition for question answering. pages 51–58.

Hariom A Pandya and Brijesh S Bhatt. 2021. Question answering survey: Directions, challenges, datasets, evaluation matrices. *arXiv preprint arXiv:2112.03572*.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100, 000+ questions for machine comprehension of text. *CoRR*, abs/1606.05250.

Iulia Turc, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Well-read students learn better: The impact of student initialization on knowledge distillation. *CoRR*, abs/1908.08962.

Wang Yanmeng Ouyang Yuanxin Xiong Zhang Xu Gezheng, Rong Wenge. 2021. External features enriched model for biomedical question answering.

Wei Zhu and Daniel Cheung. 2021. Lex-bert: Enhancing bert based ner with lexicons. *arXiv e-prints*, pages arXiv–2101.