COSC-572 Final Project Report: Question Answering on Tweet Extraction Task

Ivy Wang
Georgetown University
zw85@georgetown.edu

Tong Xiang
Georgetown University
tx39@georgetown.edu

Abstract

In this project we attempt the Tweet Sentiment Extraction Challenge which is an ongoing competition on Kaggle[1]. This challenge is an extractive question answering (QA), where the desired output is the *selected text* from the tweet that supports the *sentiment*. We explore both recurrent neural models as well as transformer-based models to make predictions on given data. We present experiment results with the jaccard score metric. We discuss in detail the choices of models and mechanisms. We critically analyze the limitations of both kinds of neural algorithms as it specifically applies to the nature of our dataset. We conclude with suggested extensions to the current methods in order to push the boundaries of the QA performance. Our code can be found on https://github.com/mathfather/Tweet-Sentiment-Extraction.

1 Related work

Question answering, sometimes called reading comprehension, is one of the most important tasks in Natural Language Understanding (NLU), requiring both understanding towards language and prior knowledge of the world (Rajpurkar et al., 2016). A large amount of datasets have been released in order to drive this field forward. The earliest data-driven approach to solve the problem of reading comprehension could be traced back to 1999 (Hirschman et al., 1999), where the authors created a dataset of 600 real 3rd–6th grade reading comprehension questions. Later on, more challenging datasets which required the ability of tracing information across sentences and knowledge inference were released (Richardson et al., 2013; Narasimhan and Barzilay, 2015; Wang et al., 2015). Besides constructing datasets by carefully designing, some researchers have also constructed cloze datasets through automatically generation of existing data (Hermann et al., 2015). But all these datasets mentioned above are either limited by their size, or hindered by their inability of sharing the same characteristic as explicit reading comprehension questions (Rajpurkar et al., 2016). The release of Stanford Question Answering Dataset (SQuAD) (Rajpurkar et al., 2016) has marked the milestone of the development in reading comprehension. It mitigated the drawback of small size and also maintained relatively high quality at the same time. In this dataset each question is posed based on a set of Wikipedia articles where the answer is a segment of text retrieved from the corresponding article. Since the most of these answers are explicitly extracted from the source, the machine reading system trained from SQuAD could suffer from the problem that it would tend to make unreliable assumption on questions for which the correct answer hasn't been stated in the given article (Rajpurkar et al., 2018). This weakness directly pushed forward to the creation of SQuAD 2.0, which combined the pre-existing SQuAD dataset with additional unanswerable questions. More recently, AI2 Reasoning Challenge (ARC) [2] was announced in order to encourage progress on deeper machine reading comprehension instead of relying on surface-level information (Clark et al., 2018). It contains 7787 questions, all of which are natural, grade-school science questions authored for human tests. In 2019, a yes/no reading comprehension dataset called BoolQ (Clark et al., 2019) was released. This further pushed the border of machine reading comprehension, since it required even deeper reasoning in order to answer these questions.

[1]https://www.kaggle.com/c/tweet-sentiment-extraction/overview/description
[2]https://leaderboard.allenai.org/arc/submissions/public

Original tweet	Selected text	Sentiment label
i wanna leave work already! Not feelin it 2day	wanna leave work al	negative
enjoy ur night	enjoy	positive
it's beeen onee year	it's beeen onee year	neutral

Table 1: Three tweets from the dataset, with their annotated *selected text* and *sentiment*

The evolution of reading comprehension methods has been continuing along with the development of datasets. Many of the early works depend on attention mechanism which was firstly introduced in Neural Machine Translation domain (Bahdanau et al., 2014), including Attention Sum (Kadlec et al., 2016), Gated Attention (Dhingra et al., 2016), Bi-Directional Attention Flow (Seo et al., 2017), Dynamic Co-attention Network (Xiong et al., 2016), and Attention over Attention (Cui et al., 2016). Most recently, the release of Bidirectional Encoder Representations from Transformers (BERT) (Devlin et al., 2018) has led the trend of pre-trained language models (PLMs). Newly proposed PLMs have achieved state-of-the-art on both SQuAD 2.0 leaderboard [3] and ARC leaderboard (Lan et al., 2019; Raffel et al., 2019). The modification version of A Lite BERT (Zhang et al., 2020) even outperformed human performance on SQuAD 2.0, with over 3 point improvement on F1 score.

2 Task

The task we take on is a Kaggle competition, titled **"Tweet sentiment extraction"**: each sample provides a uniquely-identified tweet with an annotated sentiment label, and asks one to predict the 'selected text,' which is the chosen material from the tweet that best supports the sentiment characterization. Both training data and test data include annotated sentiment labels; only the training data includes annotated 'selected text,' while the test data does not. The selected excerpts can be single words, phrases including symbols and punctuation marks, as well as the original tweet. Examples can be found on table 1.

We view this problem as a simplified version of QA problem; for one there is no explicit question and corresponding context; but if we treat the sentiment label as question(e.g. which consecutive part of this tweet can support the given sentiment label), the problem then degenerates into an extractive QA problem. Though this task doesn't involves the understanding towards the questions (Gardner et al., 2019), we still think that QA will be an effective format in case of treating this task.

As for our metric, instead of accuracy, we use jaccard score for the evaluation. The intuition behind jaccard score is that given any two strings m and n, The higher the jaccard score is, the more similar m and n are. A more detailed description of jaccard score can be found on the Evaluation page of this competition.[4] As the organizers of the challenge does not provide a test set with annotated labels, we evaluate and report scores on our own validation data.

3 Approach

For this task we do not consider using language and/or feature engineering, as the final prediction demands matches to the original text, and it would entail much more time and effort to go through text pre-processing and then convert them back into original texts for evaluation. Therefore we consider using neural models to start with.

In solving this problem we use both RNN-based and Transformer-based neural models. The data processing is different for the two kinds of models, and is each detailed below.

3.1 Data processing

Recurrent models We work within the keras (Chollet and others, 2015) framework in all of our RNN-based models. We use GloVe-Twitter word embedding (Pennington et al., 2014). This word embedding is trained specifically on 2 billion Twitter texts and contains 27B tokens[5]. For the baseline model we first

[3]https://rajpurkar.github.io/SQuAD-explorer/
[4]https://www.kaggle.com/c/tweet-sentiment-extraction/overview/evaluation
[5]https://nlp.stanford.edu/projects/glove/

concatenate the *sentiment* and the *text* fields into pseudo-sentences, which are tokenized, vectorized, and padded to a maximum length. We feed these sentences as one input into the baseline system. For our non-baseline RNN system, we treat the text and the sentiment separately, but do the same as above in terms of tokenizing, vectorizing, and padding.

As for the labels, we take the ground-truth *selected text* fields and transform them into *start* and *end* indices. The start and end indices are treated as two separate outputs, both one-hot encoded. That is, the outputs of the systems will be two separate vectors of the maximum length. In the recurrent models the indices are counted based on a character-level rather than a word-level. We decided to do so in hopes to capture sub-word differences as we noted uses of irregular spacing, special symbols, and word abbreviations from the training data.

BERT models For models based on BERT (Devlin et al., 2018) we work with the PyTorch (Paszke et al., 2019) and Huggingface libraries (Wolf et al., 2019). We use BERT Tokenizer [6] to encode the text into sub-word units. The BERT Tokenizer adds special symbols for the start and end of the sentences(i.e. `[CLS]` for the start of the sentence and `[SEP]` indicating the end of the sentence), as well as provides 'segment ids' to distinguish the tweet section and padding section of the input. Furthermore, to save time and make experiments more efficient during the training of BERT models, we decided to truncate 5 examples from the training data, in order to threshold the maximum input length to 64 tokens. As Figure 1 shows, only very few sentences were actually over the length of 64 tokens.

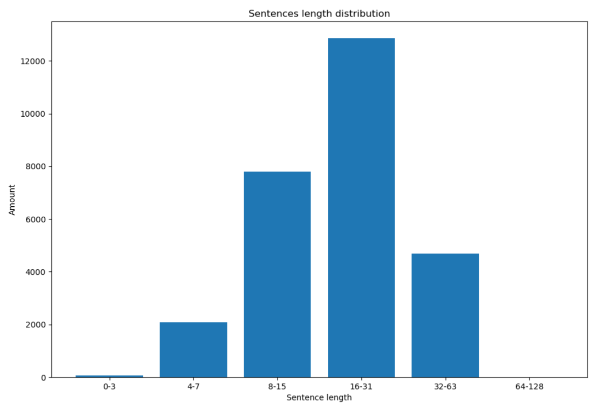

Figure 1: Sentence length distribution in tokens which are tokenized by BERT Tokenizer

As for the output labels we obtain the start and end indices at token-level, so as the final predictions for the BERT experiments.

3.2 Rule-based Post-processing

In this task we went back to observing the data after a few trials of model experimenting. In this process it was not hard to discover that there existed a correlation between the *neutral* tweets and the *selected text* field. As Figure 2 shows, most tweets that are 'neutral' actually has the entire tweet highlighted as being relevant. Intuitively this means that no words in particular flagged the tweet as sentiment-prone. Taking this overwhelming trend, we decided for the model to predict the entire tweet when the sentiment label is 'neutral.' This proves to be effective and simple, much to our relief. Besides, for those samples that have an predicted end index before the start index, we manually make the start prediction index to be 0.

3.3 Recurrent models

3.3.1 Baseline

Our baseline architecture is an LSTM-based neural model (Hochreiter and Schmidhuber, 1997). We choose LSTM over a vanilla RNN because we realize that the current dataset does not involve very long

[6] `https://github.com/huggingface/tokenizers`

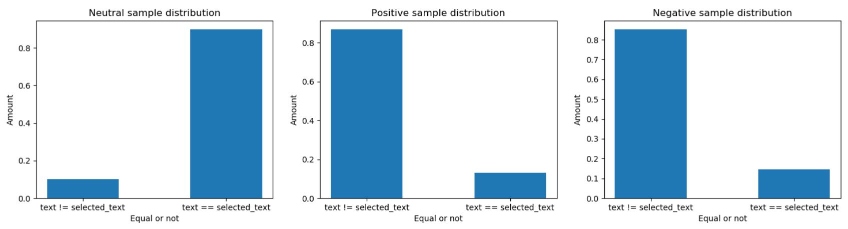

Figure 2: Original tweet selected vs. sentiment distribution

contexts (the maximum length of the tweets is 127 characters, while the 'context' field only contains one word), and LSTMs do not run into the problem of failing to remember long distance information as much as a vanilla RNN does.

This first model takes one embedded textual input and outputs two index probabilities. The architecture was inspired by the baseline for the multi-task language comprehension dataset GLUE (Wang et al., 2018), without attention implementation. It consists of an Embedding layer, three consecutive LSTM layers of different hidden sizes (128, 192, 256). It is followed by a softmax-activated Dense layer to predict the start indices. Then, the final hidden states from the LSTM is concatenated with the start predictions to go through another softmax-activated Dense layer, which yields the end indices predictions. We concatenate the output of the first Dense layer with the last hidden state to prevent the situation where the end index prediction appears earlier than the start index prediction.

3.3.2 BiLSTM-Sentiment as auxiliary input

Another recurrent model only slightly differs from the baseline. Instead of one input which is the pseudo-sentence of [sentiment; tweet], we experimented with a similar architecture that takes two separate inputs. Moreover we substitute the first two LSTM layers with Bidirectional LSTMs. In this case, only the tweet sentences go through the embedding layer, and the three BiLSTM layers. After the BiLSTM layers, the one-hot encoded sentiment input is fed in as an auxiliary input. The resulting concatenated vector goes through the two Dense layers as described above. We choose Bidirectional LSTM in hopes of taking advantage of information both from the left and from the right of the sentence. Since tweets are not really long, we think that BiLSTMs should be relatively good at noting information in both directions. We find that separating the sentiments and the tweets to be inputs at different parts of the system can give improvements to the performance.

3.3.3 BiLSTM-Concatenated text+sentiment embedding

This model which also shows improvement from the baseline also takes the text and the sentiment as two separate inputs. Instead of feeding the one-hot encoded sentiment labels as an auxiliary input that bypasses the previous BiLSTM layers, we concatenate the embedding for the sentiments with the embeddings for the tweets themselves. We further add dropouts to the first hidden layer.

3.4 BERTModel without sentiment labels

Upon transferring to using the BERT (Devlin et al., 2018), we first attempted not using the sentiment information in the model input. We directly make use of the sequence output of the last hidden layer from BERT by adding an linear layer on top of the BERT architecture. Each output node has two predictions: one for predicting whether it is a start index and one for predicting whether it is an end index. To transform the prediction scores into legal probabilities, we apply softmax over the two output prediction sequences. For performance and simplicity we choose the indices that have the highest probabilities to be our final predictions. The loss function for the prediction of these two indices is Binary Cross Entropy.

3.5 BERTModel with sentiment labels

In order to make use of the sentiment information, we wanted our model to learn the information of *selected text* and *sentiment* at the same time, thus we decided to apply a multi-task scheme to our task.

Instead of only predicting start and end indices for each sample, we additionally predict the sentiment for each text at the same time. This is achieved by making use the sequence output as well as the pooled output of the BERT model. The pooled output (the hidden output for token `[CLS]`) is used for indicating the start of the sentence. But since `[CLS]` is concatenated with every sample in the dataset, its output embedding could be also viewed as the semantic embedding of the whole sentence. In this manner our model will predict the *start* and *end* indices while learning the sentiment information. For both extractive task and classification task, we use a weighed combination of three Binary Cross Entropy loss as our final loss function.

4 Results and Discussion

Table 2 shows our current experiment results on the above models. While the first three models did not use the rule-based post-processing, the rest four did. The epoch number is 10 for recurrent models and less for BERT models; the batch size is 64. Other hyperparameters vary depending on the specific model. Our best model was the BERT model without using sentiment information, giving a 0.662 jaccard score. In the meantime the BERT Model that uses the sentiment label does not achieve nearly as much as its counterpart without the sentiment label, although its score is still higher than all of the recurrent models. As for the recurrent family models, they all have jaccard scores in the 0.5's range, with the best of them (0.571) being the BiLSTM model with auxiliary sentiment input.

The baseline system, compared with its variations, shows that it definitely helps the system when the tweets and the sentiments are separate inputs. As for how the sentiment information should be separately dealt with, our experiments only started to reveal, but the conclusion remains open. For now, our results lead us to believe that the sentiment information only proves to be useful when it's not fed, or fed very late to the model.

Comparing our two Transformer models, they both converge very quickly, and while the second BERT model is capable of being multi-task, it does not seem to give a result as good as the single-tasked model. In this case the sentiment input might have hurt the performance, along with other factors.

Furthermore, a text-only neural model with rule-based sentiment utilization outperforms a text-and-sentiment-combined neural model. This is important to note here because the models that do not use post-processing perform worse than the ones that do use it.

Model	**Avg. Jaccard Score**
Baseline with one input	0.52
LSTM with auxiliary sentiment input	0.534
BiLSTM with auxiliary sentiment input	0.530
BiLSTM with auxiliary sentiment input	0.571
BiLSTM with concatenated embeddings	0.569
BERT without using sentiment label	**0.662**
BERT with using sentiment label	0.633

Table 2: Experiment Result

5 Error Analysis and Potential Extensions

The errors that we observe are mainly as follows. For the recurrent models, though the model architecture is set up to avoid *end* being less than or equal to *start* indices, it suffers from another type of inaccuracy, namely failing to detect word boundaries and outputting partial words. An example mismatched output is shown in table 3. As shown, the word 'love' was only being half-outputted, though the potentially positive-indicating word 'LOL' was indeed captured. An immediate remedy would be to base the indices on words rather than characters, though in that case arguably the model would fail to catch symbols, punctuation marks, and abbreviations. But seeing that the metric is word-based, it ignores partially correct words in total – so word-based indices may after all be an improvement to the jaccard score in particular.

Example pred	LOL - Birmingham was my 1st lo
Example gold	Birmingham was my 1st love..

Table 3: Example prediction, with sentiment being *positive*

A potential improvement we can next attempt is to implement customized loss functions and training metrics. Currently we are using Binary Cross Entropy against the gold labels; but with this function, the loss computation only has a binary judgement – whether right or wrong – rather than a relative, gradient judgment, which is more desirable. This is so because a relative, weighted metric would know to punish shorter distances (between predicted indices and gold indices) less and longer distances more.

Moreover, we note that the BERT embedding exceeds in effect compared to the GloVe-Twitter embedding. Even though the GloVe embedding we used for LSTM models are in the same domain as our training data, BERT model makes use of contextual embeddings. Next experiments are in line to try out BERT embeddings on recurrent models, and make a comparison of the two architectures under the same embedding.

Besides all these, we found an existing problem with our current multi-task model. While training the model we made use of both indices as well as sentiment information; the problem is that during the evaluation phase we didn't use the given sentiment label. That is to say, our model will go directly to predict the sentiment label and the indices at the same time order than make use the sentiment label to enhance the performance of index prediction. Modifications are needed in order to combine the given information on the test set.

6 Conclusion

In this project we applied both recurrent models and Transformer models with a number of variation to solve the Tweet sentiment extraction task as a Question and Answering problem. We set out to use these two families of neural models in particular because recurrent models are good at sequential text data, and Transformer In this problem we take the tweet sentence and the provided sentiment labels to be the input(s), and the start and end indices of the selected excerpt to be the output. The excerpts are flag words and phrases that best indicate the labeled sentiment. In our approach, to tailor to the current Twitter dataset more, we combined rule-based prediction along with the text-only neural networks, and obtained much better results. We truncated the maximum input length to 64 instead of over 100 for Transformer-based models to save the training time of the Transformer models.

Further, we note that evaluation of the task is tricky and less straightforward than other NLP tasks. We think one of the next steps is to customize our own loss function to punish relative, distance-based accuracies instead of absolute, binary accuracies.

A serious challenge revealed from our series of experiments was the way in which the sentiment information combines with the tweet information. Though both are text information, the tweets have much higher variation, and resembles natural language more. On the other hand, the sentiment information is more categorical than textual. (One reason we refrained from coding a typical key-query attention layer was that it might have been an overkill for this task). Given the current abilities of our model, we still think there are much room of improvement. In the future we look to explore better ways to incorporate the two aspects of the dataset, as well as refining the model architecture, and finding appropriate training metric that corresponds to the evaluation metric.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

François Chollet et al. 2015. keras.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord.

2018. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv preprint arXiv:1803.05457*.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. 2019. Boolq: Exploring the surprising difficulty of natural yes/no questions. *arXiv preprint arXiv:1905.10044*.

Yiming Cui, Zhipeng Chen, Si Wei, Shijin Wang, Ting Liu, and Guoping Hu. 2016. Attention-over-attention neural networks for reading comprehension. *arXiv preprint arXiv:1607.04423*.

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.

Bhuwan Dhingra, Hanxiao Liu, Zhilin Yang, William W Cohen, and Ruslan Salakhutdinov. 2016. Gated-attention readers for text comprehension. *arXiv preprint arXiv:1606.01549*.

Matt Gardner, Jonathan Berant, Hannaneh Hajishirzi, Alon Talmor, and Sewon Min. 2019. Question answering is a format; when is it useful? *arXiv preprint arXiv:1909.11291*.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Advances in neural information processing systems*, pages 1693–1701.

Lynette Hirschman, Marc Light, Eric Breck, and John D. Burger. 1999. Deep read: A reading comprehension system. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics*, pages 325–332, College Park, Maryland, USA, June. Association for Computational Linguistics.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9:1735–80, 12.

Rudolf Kadlec, Martin Schmid, Ondrej Bajgar, and Jan Kleindienst. 2016. Text understanding with the attention sum reader network. *arXiv preprint arXiv:1603.01547*.

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*.

Karthik Narasimhan and Regina Barzilay. 2015. Machine comprehension with discourse relations. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1253–1262.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. 2019. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*, pages 8024–8035.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2019. Exploring the limits of transfer learning with a unified text-to-text transformer. *arXiv preprint arXiv:1910.10683*.

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

Pranav Rajpurkar, Robin Jia, and Percy Liang. 2018. Know what you don't know: Unanswerable questions for squad. *arXiv preprint arXiv:1806.03822*.

Matthew Richardson, Christopher JC Burges, and Erin Renshaw. 2013. Mctest: A challenge dataset for the open-domain machine comprehension of text. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 193–203.

Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. 2017. Bidirectional attention flow for machine comprehension. *ArXiv*, abs/1611.01603.

Hai Wang, Mohit Bansal, Kevin Gimpel, and David McAllester. 2015. Machine comprehension with syntax, frames, and semantics. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 700–706.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2018. Glue: A multi-task benchmark and analysis platform for natural language understanding. In *BlackboxNLP@EMNLP*.

Thomas Wolf, L Debut, V Sanh, J Chaumond, C Delangue, A Moi, P Cistac, T Rault, R Louf, M Funtowicz, et al. 2019. Huggingface's transformers: State-of-the-art natural language processing. *ArXiv, abs/1910.03771*.

Caiming Xiong, Victor Zhong, and Richard Socher. 2016. Dynamic coattention networks for question answering. *ArXiv*, abs/1611.01604.

Zhuosheng Zhang, Junjie Yang, and Hai Zhao. 2020. Retrospective reader for machine reading comprehension. *arXiv preprint arXiv:2001.09694*.