Characters, Embedding Concatenation, and Span Filtering for better QA models across domains

Author
Name Here
EID
EMAIL

Abstract

This paper contributes 3 moderate changes to the Baseline Model given. Firstly a character embedding layer, a method of combining embeddings, and finally a simple NER system to filter spans via weightings. We show that each of these changes will improve the accuracy of the model for both in-domain and cross-domain datasets. Furthermore, we will demonstrate why these improvements have significant impact on the model and how future work could result in improved models not only for the Baseline Model but for state of the art models like Electra in common applications such as Amazon's Alexa.

1 Introduction

Most modern systems have understood that understanding context with word embeddings can be a powerful tool, and furthermore, character embeddings can help to increase performance. The Baseline model lacks character embeddings and thus suffers from out-of-vocabulary (or new) words. Another disadvantage of the Baseline Model the lack of intelligent span (or answer) filtering. However, even state-of-the-art models can struggle in validating their answer or giving intuitive reasons as to why the model predicted a certain span over another. This introduction section will cover each of these problems individually and in order, with a final section that briefly describes the overall structure of this paper.

1.1 Embeddings and Understanding

Intuitively, when someone is fluent in a language, understanding a language becomes more than understanding individual words as a whole. Fluent speakers are often able to piece together meaning by the pieces of the word itself and in fact this kind of understanding is often studied in various fields of linguistics like Morphology. The benefit of having such a rich knowledge of a language and how its words are formed allows an individual to intuit the meaning of a new word that the individual has not encountered before. Using context clues as well as the morphology of the word, the individual will be able to come up with some meaning for that unknown word. Even if the understanding of an unknown word is not entirely accurate it's often better than a random guess or ignoring it entirely.

The Baseline Model in its current implementation is able to extract the fluent speakers equivalent "context clues" through word embeddings; however, it does not look at the internal structure of the words. This means that the Baseline Model will fail on passages and questions that include too many words that were not seen in the training data. By embedding characters and words and then combining those two vectors, the model is able to become apt at distinguishing meaning from text even if that text contains mostly new words.

Naturally the question is how to best represent character embeddings to the model. The BiDAF model (Minjoon et al., 2018), which is tested in this paper, uses a convolutional network (Yoon Kim, 2014) to create the embeddings and then a highway network (Rupesh Kumar et al., 2015) to combine the character and word embeddings into one representation that the model can use. However, in this paper, we show that creating a network that mixes word embeddings and character embeddings could be detrimental to the overall performance of the model.

1.2 Question Understanding

When a person is posed a question, oftentimes the individual answering the question has some idea of what the answer should "look" like. That

doesn't mean the individual knows the answer right away, but rather has a template like structure that would reject ridiculous answers that don't fit into the template. For example, if someone were to ask for person X's age, the expected answer would be a number and not a location. Beyond the formation of a template for an answer, people often are able to distinguish when a question is malformed and requires clarification before an answer can be proposed. Following the same example, if the question was for person X's age, but all the potential answers were locations - a person would be able to consult with the who ever is asking the question and get more details to help clarify the confusion.

This poses an interesting problem for the Baseline Model as well as current state-of-the-art models. How can a model check if its answer makes sense given the question, and how can a model know when to ask for clarification? A few state-of-the-art models are already attempting to answer this question of validating the answer for datasets like SQuAD2.0 (Zhang et al., 2020), though focused almost exclusively on the ability to answer a question given a passage or not.

Our model shows that you can get small, but not negligible, improvements by validating the type of question and the expected answer using a very small template of question types. The goal being to create a mapping from question to answer types. Once the model has an expected answer type, each potential answer span is adjusted according to what type of entities are inside of it. If the span includes entities that match the answer type, that span should be rewarded and for spans that include opposing entities or no entities should likewise be penalized. The proposed change in this paper produces small improvements to the overall accuracy, but what is more promising is the intuition from this experiment for later tests and improved models.

1.3 Outline

This subsection briefly covers the flow of the paper and what each section covers. Section 2 explicitly states each proposed change explored in this paper. Section 3 covers the implementation of each improvement including the original exploration and structure of the improvement. Section 4 covers the ablations of each improvement and how it affected the performance. Section 5 covers the results of the model comparing it against the baseline in all datasets with an analysis. Section 6 covers the discussion of the improvements, why they enhance the model, hypothesis on how to further improve each modification, and a suggestion to make the model more versatile for state-of-the-art-models like Electra. Finally, Section 7 is the conclusion of the entire paper.

2 Proposals

We propose a total of 3 changes to the Baseline Model. First, a character embedding layer using a convolutional network to address the problem of out-of-vocabulary words. Second, a better method of combining word and character level embeddings to address the problem of how models represent questions and passages in QA. Lastly, a weighting scheme to help filter out bad spans using a NER system to address interpret-ability as well as confidence levels in a models answer.

3 Implementation Details

This section explores the implementation details of the Character Embedding Layer, Concatenation Method, and Span Weighting With NER in this order.

3.1 Character Embedding Layer

The first implemented improvement to the Baseline Model is to allow for character embeddings, which requires changes to both the Preprocessing steps and Inference steps of the model. First, each word in each question and passage is decoded into a vector of length 24 (this is configurable) where each value of that vector ranges between 0 and 27. Each value in that vector represents a character in numerical form except for 0 which is used as a padding character for unknown characters or for words that are smaller than 24 characters. To get their numerical values, each character is mapped to a lookup table that just includes characters [a-z] (Yoon Kim, 2014),

This is a hyper-parameter that could be further tuned and if needed could encapsulate all English characters to better fit cross domain datasets. However, for the purposes of this project, a simple a-z character lookup table was sufficient. Just to be clear, a vector for the word "cat" would be [3, 1, 2, 0, ...]

This format of embedding words into character feature vectors worked well for our implementa-

tion though there are other ways of doing it worth mentioning, for example, one-hot vector encodings per character. However, this concise method worked best for this implementation.

Once the character feature vector reaches the PyTorch model, it is first passed into an embedding layer. The embedding dimension has 27 possible values (26 for the alphabet and 1 for the padding character), it reduces each feature vector to a vector of length 8. Besides the size of the input vector, these values are able to be tuned and could produce better accuracy if adjusted effectively. However, the model that worked best for this project was with an embedding layer of size 8.

Although the characters are technically embedded, they are not taking into account the characters around them nor are they extracting features about the composition of words from various combinations of characters. I.E. "ed" combined is usually an inflection of a verb, this is not present yet in the embedding feature vector. To accomplish this, our model, further embeds the characters with a Convolutional Layer (Minjoon et al., 2018). However, to convolve over the characters, the data has to be mutated to fit the input of the CNN Layer.

The convolutional layer wants the batch size and sequence vectors to form a matrix and be combined into a single dimension with the character embeddings (8 in our model) form the next axis. The last dimension specifies the size of the kernel that is to go over the Batch size * Context length, 1, Character dimension matrix. Something that might help intuit why this is, is to think of the first two dimensions as an image matrix that the layer will slide a convolutional kernel over trying to find a matching pattern.

After the convolution layer our model implements a max pooling layer which will take the most significant features. The last step is to reshape the data yet again to match that of the word embeddings.

3.2 Concatenation Method

Although this is the smaller of the three changes, it warrants its own section because of the significant impacts on performance. Our experimentation and resulting model has three modes that can be used to combine the word embeddings with the character embeddings. They are 80/20, highway, and Concat. The 80/20 mode was the first implementation choice, it takes the word embedding and weights that vector by 0.8, then it takes the character embeddings and weights those by 0.2, with the resulting weighted vectors the model sums them up for the resulting question and passage embeddings.

80/20 method of combining the word and character embeddings did not do too well in the end. Although it did provide improvements to the overall accuracy it was far behind the highway model. The highway model was inspired by (Rupesh Kumar et al., 2015) and it is the implementation in the original BiDAF paper (Minjoon et al., 2018) which we based the first two changes on. Our highway model followed closely to what the BiDAF paper suggested which is a two layer highway network that takes both the question and passages word and character embeddings.

In the highway network, both layers are linear and of the size two times the embedding dimension to account for the input being a concatenated vector of word and character embeddings. The block layer has an activation function of ReLU while the gate layer is activated with Sigmoid. The two layers are combined following the equation in (Rupesh Kumar et al., 2015) and this is done twice with two different sets of layers.

The highway network did much better than the 80/20 method of concatenating the two vectors, but we tried one last method of combining the two embeddings which ended up being the best way performance wise and the simplest. The last way to combine the two vectors was to simply concatenate the two together (which is also the first step for the highway network implementation). This implementation achieved the highest accuracy at about 1% better than the highway network.

An important note to these implementations is that for all of them except the 80/20 method, all hidden dimensions past the embedding layers were increased by 2. So some of the benefits and improvements achieved from the various concatenation methods when compared to the 80/20 method could be attributed in part to the increased network size.

3.3 Span Weighting With NER

Span Filtering using a NER framework supplied by SpaCY is done post training, during the inference part of the model. A lot of exploration came into this section, which was boiled down to a sim-

ple implementation inspired by (Diego Molla et al., 2006).

Most of the exploration circled around trying to find out the best way to use a Named Entity Recognition system to better match spans with the answer. Attempting to match named entities from the question to the answer provides little help because one, the answer is usually a different entity type than the question posed, and two the distance from the answer and the matching named entity could be far depending (so locality won't help). However, if you are able to deduce what type of entity best fits the answer to the question, you can iteratively go over each of those spans that includes an entity that would best answer the question and choose from those. The challenges with that type of model is, how do you choose the correct question type, and how do you choose which entity labels answer each question.

Essentially our model attempts to decode a question into one of five question types. This is done with a simple lookup table for words in the question that indicate the type of question that is being asked. For example, question that are looking for a "PERSON" or "ORGANIZATION" often times include the word "who" in the question itself. This type of intuition lead our algorithm to have 5 classes of questions, with about 5 key words per class. The first class to match is the class assigned to that question. From there, each entity in the passage that does not match any of the labels the question is looking for gets a reduction in its probability (which acts more like a score than a probability now) for both start probs and end probs. This shifts the weight that the spans will initially look over towards entities that are part of the question type grouping. Next, any span selected out of the passage, if the span includes (partially or entirely) an entity of the type the question is looking for, that span is rewarded in its joint probability (the score of the span). If the span includes no entities it is penalized and if the span includes more than 1 entity matching the list it is penalized lightly (to enforce shorter but still correct answers)

An important discovery here is the importance of the NER system being used for this method of filtering to be efficient. Using SpaCY's small model did not find all the named entities in the passage, which would lead to penalized spans when they in fact had the correct answer. To remedy this in our model, we used the best model SpaCY can offer, its large model, but improvements to the NER system could still improve the accuracy of the weighting scheme.

4 Ablations

This section covers various improvements to the model that were important to the paper. The first is how each improvement effected the performance of the model. The second is how the Concatenation Method effected the performance of the model.

4.1 Individual Improvements

EM Scores	SQuAD	Adversarial	Newsqa	Bioasq
Baseline	46.14	34.08	18.99	9.24
Embeddings	50.75	37.77	**21.32**	14.56
NER	**50.83**	**38.05**	21.3	**14.56**

F1 Scores	SQuAD	Adversarial	Newsqa	Bioasq
Baseline	58.57	44.63	30.63	16.97
Embeddings	63.54	49.13	33.54	23.29
NER	**63.65**	**49.37**	**33.54**	**23.29**

Each model was trained on a model with 128 hidden dimension size, 100d word embedding and character embeddings, using the Concat combination method, bidirectional LSTM using the SQuAD training dataset.

Each improvement to the model increased the accuracy with character embeddings giving a boost of 5.32% and 6.32% to the Bioasq datasets EM and F1 scores respectively. Minimally the character embeddings improved the accuracy of the Newsqa's EM and F1 scores by 2.33% and 2.91% respectively.

Adding the NER span weighting implementation on top of the character embedding model improved the adversarial EM/F1 by 0.28% and 0.24% respectively. However, it negatively impacted the EM score for Newsqa by -0.2%. These results, although they are consistent, are small enough to warrant a significance test before any definitive statements are made.

4.2 Concatenation Methods

EM Scores	SQuAD	Adversarial	Bioasq
Baseline	46.14	34.08	9.24
80/20	48.35	36.77	8.64
Highway	49.95	37.16	13.43
Concat	**50.83**	**38.05**	**14.56**

F1 Scores	Squad	Adversarial	Bioasq
Baseline	58.57	44.63	16.97
80/20	60.92	47.66	17.01
Highway	62.16	46.83	21.37
Concat	**63.65**	**49.37**	**23.29**

Each model was trained with a 128 hidden dimension size, 100d word and character embeddings, bidirectional LSTM, with character embeddings turned on, on the SQuAD data set.

The Concatenation method outperformed all other variations of combining word and character embeddings beating out a two layer highway network by as much as 1.3% and 1.92% on the Bioasq EM/F1 score respectively.

5 Results

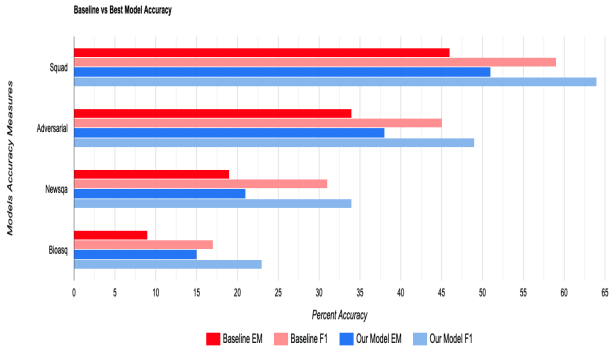

Figure 1: Graph showing results of the Baseline Model and the Best model. Results are compared against a locally trained Baseline Model. Parameters that differ from the defaults for the models are, 128 hidden dimension size, 100 dimension embeddings, 100d glove embeddings.

Our model was able to achieve on average a 4% accuracy increase in both the EM and F1 scores for all datasets, with the exception of the Newsqa which received a 2.31% / 2.91% increase to the EM/F1 scores respectively.

5.1 Analysis

We expected to see a spike in cross domain accuracy with character embeddings because of the ability to understand words that were not seen during training. This was quite evident in the results, especially with Bioasq which received the highest increase in performance at 5.32% / 6.32% for its EM/F1 respectively. Interestingly, Newsqa, although cross-domain did not receive a much of a boost. We believe that this can be attributed to two things from our model. First, the character lookup table does not account for special characters, and secondly the we believe that the answers for a lot of the Newsqa questions are named entities. With Character Embeddings our model is able to learn new words from the similar structure of words it saw during training, but if Newsqa had words that did not follow a similar structure, names, locations, organizations, different languages, etc. then our Character Embedding layer would not be of much use.

We also saw huge benefits in combining the Character Embeddings with the Word Embeddings directly. Although expanded on further in the Discussion Section, we could not come up with a clear cut explanation as to why this method triumph over the others. We do have some intuitive explanations that follow in the Discussion, but mainly we believe it to be related to reading the passage twice each in a different representation. We also believe that this phenomena could be related to training. For instance, the meaning of a words morphology is usually independent from the question being asked. One interesting side note for this improvement is that the 80/20 method actually hurt the performance of the Bioasq dataset reducing the accuracy by 0.6%. The reasons for the decrease could be related to the same reason why the Highway Network failed to out perform the Concat method, essentially combining word and character embeddings are unintuitive to the model. In the 80/20 method, we assumed the correct distribution and meaning from the character and word embeddings, rather than trying to learn it through a neural network, which is why we think these results share the similar impact and even hurt the performance in some cases.

Lastly, the Span Filtering with NER improvement had very small impact, having at 0.28% improvement to the Adversarial EM score. Although this number is small enough to be considered noise

without a proper significance test, the improvements were consistent in the in-domain datasets. It did not; however, help with cross-domain datasets. The reason for this is due to the hyper-parameters we set during our models training process on the way questions are penalized and rewarded. In the SQuAD datasets, often times smaller answers are better than larger ones. Our model penalizes any span that has more than 1 matching typed entity because that span is likely to be too long (the model could pass back the entire passage if there wasn't a size limit and the passage was filled with matching entities). This assumption is not true for all datasets and infact can hurt the performance on the dataset as seen with the Newsqa EM score which deceased by 0.02% (again that's a very small change, so further significance testing would be required). On top of this, our model was further tuned to over-fit the SQuAD datasets because of our implementation and how we were relating questions to answers. In our implementation, we expected most questions to contain simple tells on what the question was asking for, words like "who" to indicate a person, or "where" to indicate a location. Bioasq and Newsqa could be asking more detailed or nuanced questions that are not fitting into the correct question template our model set up. We will talk about how we can overcome these challenges in the current NER implementation in the Discussion Section.

6 Discussion

This section is separated by Character Embedding Layer, Concatenation method, Span Weighting with NER, and Answer Type Classification Model in that order.

6.1 Character Embedding Layer

As expected, the character embeddings improved the model in all datasets. Even the three various ways to combine the character and word embeddings all showed improvement over the baseline which is quite telling on just how important it is to have a model with character embeddings. The obvious reason for why character embeddings helped improve the performance is due to the understanding of new words through morphology, this is expressed heavily in the Bioasq dataset because of the unique vocabulary in it (this is also the dataset that got the highest improvement in accuracy). There is still a lot to explore with the character embeddings though. For example, the embeddings in the proposed model only took into account letters between a-z, no hyphens, commas, apostrophes, etc. This is a real limitation of the model that could improve the model even more on domains like Newsqa and Bioasq which often use special characters in their documents. Beyond that, having a large enough character embedding layer with enough data from a more varied dataset could help improve the quality of character embeddings. For example, a lot of the names that might exist in the Bioasq often times follow scientific naming patterns, this type of morphology is unlikely to be present in the SQuAD training dataset and thus is lost on the character embedding of this model. However, a 4% performance increase in the model with a relatively small character embedding layer is still a high improvement.

6.2 Concatenation Method

As mentioned before, interestingly, the way the character and word embeddings combined only improved the model no matter what method was chosen. However, a simple concatenation of the two vectors seems to produce the best results. The 80/20 method intuitively makes sense as to why it under performs because there is an underlying assumption of how important the data in the two vectors are. Neural Networks (in this case a Highway Network) would allow us to better understand the relationship between the characters and represent their relationship if it were complex. This sounds like the prime implementation, however, it performs against the Concat method. It could be a sign that the morphology and context clues of the words in the sentence do not share a complex relationship in question and answering. Intuitively this explanation makes a bit of sense, because a person reading a sentence usually doesn't look at the morphology of the first word to better understand the last. Instead, people use the morphology of the word only to better understand that single word (usually). In this case, concatenation may be the best implementation for a QA model, its able to see the context twice, once in the word embedding format and then again in the character format. Although the explanation lacks a mathematical theory, perhaps "reading" the context twice in two different forms is helping the model make correct associations with the input when faced with words outside of its vocabulary.

6.3 Span Weighting With NER

The least impactful change, but the one that leads to interesting insights, is the span weighting using a NER system during inference. The implementation in the proposed model is lack-luster and most likely under performing on datasets other than the adversarial and SQuAD dev datasets due to overfit parameters for questions in that domain. A good example of this is answer length. Currently the model rewards answers with one, and only one, entity that matches the expected answer label of the question. In the case of the SQuAD datasets, this was optimal as a lot of the answers had various forms that mostly included only one or two named entities in them. However, this is not the case for answers in the Newsqa or Bioasq datasets which is why the performance is stagnant or worse on those.

6.4 Answer Type Classification Model

Span weighting can be improved in various ways, but the one that seems most promising is something akin to the model proposed in (Zhang et al., 2020). In their paper, they use a model that attempts to guess if a question has an answer, and if not, it elects to return a separate answer from any of the given spans (in their case the null string). We propose an improvement that would consist of a new layer that takes the embeddings of a question and produces an expected answer label as output. Interestingly, although we did not find any datasets that had labels for questions -¿ answer type due to "answer type" being subjective to the NER system being used, we did come up with a solution to train without having to use any other data. During training, the ground truth labels need to be updated with a NER systems entity labels for any entity inside of that ground truth span. This allows, with slight modification of the training data, for the model to learn not only what words are likely to be in the span, but also what entity labels should be in the span. An important distinction though, is that this layer does not impact span logits, it only predicts the expected answer type from a question, in this way, the model produces 3 outputs, start logits, end logits, and expected entity type. From there, the implementation of span filtering can look for spans that are close to the entity that the model predicts the question wants.

Implementing a model that predicts entity types would have tremendous impacts, if it was able to predict the entities reliably which is mostly dependent on the NER system being used in the first place. Beyond improving accuracy on questions, which as seen in this experiment results in smaller accuracy impacts, this model could produce more information into why a specific span was returned as the answer. Debugging a QA system would become easier if the model understood what type of question it was trying to answer and the debugger could easily spot what the model thought the question was asking for. As of now, modern models hide the interpretation of the question until the span is selected, this model would allow for more insight into that learned decision. Secondly, a model like this could be implemented to ask for further clarification on the question if all the spans included entities sufficiently far enough away from the entity predicted. This would be extremely helpful in systems like Siri, Alexa, etc. for when they are unsure about a question due to noise in the voice to text translation layers, or for co-reference resolution issues. It could also help in discourse, if intelligent systems wish to carry longer conversations with humans, they would have to behave more like humans in that even a great listener has to ask what someone means from time to time. This model could easily track when its answers are becoming less

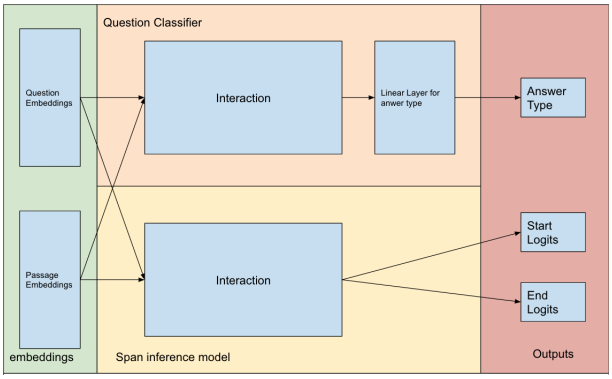

Figure 2: Simple example model for how a Reading Comprehension model could be constructed to produce both span logits and an answer type label. The outputs are then passed into a span filtering algorithm that attempts to match entities from a NER system with the answer types predicted from the model. It's important to note that the answer type labels and the NER system entities have to be from the same schema, i.e. for training we suggest using the same NER system to produce training labels and for predicting entities in the passage.

confident and ask for more input to establish better confidence in its answers.

7 Conclusion

Our model saw an improvement of on average 4% with the three changes proposed. With a clear impact from character embeddings, showing that understanding the structure of a word is incredibly important. The results also suggest that there may be a benefit in digesting the word embeddings and character embeddings twice as reflections of the same input, which would go against what is suggested in the BiDAF model (Minjoon et al., 2018). Finally, although not as impactful, a span weighting mechanism can help important impacts in validation and explanation as well as clarification methods for future systems.

References

Zhuosheng Zhang, Junjie Yang, and Hai Zhao 2020. *Retrospective Reader for Machine Reading Comprehension*, https://arxiv.org/abs/2001.09694

Bowen Wu, Haoyang Huang, Zongsheng Wang, Qihang Feng, Jingsong Yu, Baoxun Wang. 2019. *Improving the Robustness of Dee Reading Comprehension Models by Leveraging Syntax Prior*. Association for Computational Linguistics:53–57

Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, Hananneh Hajishirzi. 2018. *Bi-Directional Attention Flow For Machine Comprehension*, https://arxiv.org/abs/1611.01603

Yoon Kim. 2014. Convolutional Neural Networks for Sentence Classification. https://arxiv.org/abs/1408.5882

Rupesh Kumar Srivastava, Klaus Greff, Jürgen Schmidhuber 2015. *Highway Networks*. https://arxiv.org/abs/1505.00387

Diego Molla, Menno van Zaanen and Daniel Smith 2006. *Named Entity Recognition for Question Answering*. Proceedings of the Australasian Language Technology Workshop 2006:51–58