Enhancing BiDAF with BERT Embeddings, and Exploring Real-World Data

Priyanka Sekhar
Stanford University
psekhar@stanford.edu

Ryan Mui
Stanford University
ryanmui@stanford.edu

Sumit Minocha
Stanford University
sminocha@stanford.edu

Abstract

In our project, we used BERT embeddings to enhance BiDAF for the task of question answering on real world data. We had 2 main goals: 1) maximize performance (measured by F1 and EM scores) of our combined BERT and BiDAF architecture on the SQuAD 2.0 leaderboard 2) explore our model's performance on our custom dataset, which contains real customer service interactions from a global cosmetics manufacturer, Benefit Cosmetics. This second goal is an additional exploration that extends the goals of the default project. Our BiDAF models with BERT embeddings achieved dev EM/F1 scores in the 50s after just 2 epochs, instead of the 30 required for the baseline BiDAF. Our best run on the industry dataset was with the BERT fine-tuned model, which achieved dev EM/F1 scores in the 70s on the combined SQuAD + industry dataset. However, this model still did poorly on the industry examples, which highlights the challenges of adapting research to real-world settings. Our best test set performance was with the BERT uncased small model fine tuned for 2 epochs. (EM/F1 72.257/75.586).

1 Introduction

The ability to answer questions given text information is an important problem in modern natural language processing (NLP). Solving this Question and Answering (QA) task requires that a machine "understand" both the question and the provided text well enough to extract the correct information. Ultimately, creating such a solution to this problem opens the doors to automating many services such as customer support and broadens the understanding of human language and machine understanding.

The first step in building any NLP system is representing words with vectors of numbers that a computer can digest. The method of embedding words to numbers can have significant impacts on the performance of a system. Two popular techniques for learning word embeddings are GloVe [6] and Word2Vec [7]. Both of these embedding methods learn embeddings by looking at the occurrences of nearby words. In this way, GloVe and Word2Vec learn word embeddings based on the local context of a given word. The issue with these embeddings however, is that they are unable to capture the multiple meanings of words in different contexts. A solution to this problem is found in BERT [1]. by pre-training a deep bidirectional transformer architecture, BERT is able to generate different word embeddings for a given word depending on its context. This breakthrough motivates our use of BERT embeddings on a standard BiDAF model for QA.

Finally, the progress made in research has limited practical importance if the results cannot be replicated in a real world setting. Thus, we explore the ability of our model to perform on a real

32nd Conference on Neural Information Processing Systems (NIPS 2018), Montréal, Canada.

world dataset from Benefit Cosmetics, a global cosmetics manufacturer selling at over 2,000 counters in more than 30 countries.

We found that although BERT embeddings can be used, they come with several preprocessing challenges and do not guarantee parity of performance with a fine-tuning of BERT. Our BERT + BiDAF models achieved EM/F1 in the 50s and required double the training time. The training time could be improved with more efficient preprocessing, and scores would likely improve with more training (since we were only able to train for 2 epochs per experiment) but the memory and resources required for these changes may be prohibitive for simple plug and play adaptation of BERT embeddings to some models. Furthermore, we found that the differences in distribution between industry datasets and research datasets may render state of the art models much less impactful than carefully considered, domain-specific heuristics.

2 Related Work

Both Word2Vec and GloVe are methods for learning word embeddings. The difference between the two is that GloVe is a count based model, while Word2Vec is a predictive model. In both cases, a single word embedding is learned for each word, and it cannot be adapted to the multiple meanings that a word might have. For example, the embedding for the word "tear" is the same whether the sentence is "a tear on the face" or "a tear in the jacket".

BERT builds off of the Transformer architecture and showcases the performance gains of pre-training transformers to perform several NLP tasks [10]. The a major benefit of such an architecture is that the BERT can address several different NLP tasks, achieving state of the art results by adding a task-specific top layer to a given model. It is this property which we explore by using BERT embeddings on a BiDAF model.

The BiDAF model we use is the same model described in the default project handout. It is based on the work of Seo et al., but it does not use character embeddings [8]. The model serves as our base model and allows us to quickly investigate the challenges of adapting out of the box BERT embeddings to models that are built to accept GloVe vectors as input.

Note that we investigated QANet as an alternative base model to BiDAF [4], but the amount of time it took to train was fairly long and others were having trouble replicating the published benchmarks for the SQuAD 2.0 dataset. Thus we opted to use the provided baseline BiDAF instead.

3 Approach

3.1 BERT Embeddings

In our project, we use BERT [1], a current state of the art language representation system to generate word embeddings. We take the output from the hidden states to generate new embeddings for each text input. These embeddings are then fed into the remaining parts of our modified BiDAF model, described in section 3.3.

The BERT Base model consists of 12 encoder layers (or Transformer Blocks), where the output of each encoder layer along each token's path can be used as a feature representing that token. These transformers are pre-trained on mask and next sentence pre-training objectives, enabling them to learn and encode bidirectional meaning in ways previous state of the art approaches could not. Because we can extract output from each of the 12 transformer layers, there are several possible ways to construct our word embeddings, which we will explore in section 4.3.

Unfortunately, training BERT takes considerable amount of time and resources. Thus, we use a pre-trained BERT model by huggingface [2] to generate the hidden state output for each input at train time. We then combine these hidden states as described in 4.3 and test the performance of our new embeddings when used as input for the provided baseline BiDAF.

Our Original Contribution The BERT paper only formally reports in detail the impact of different BERT embedding combinations for entity recognition and classification tasks. However, other authors have suggested that the impact of different combinations of BERT embeddings may be task specific [5]. For instance, early layers may be better at low-level part-of-speech tagging, while higher level

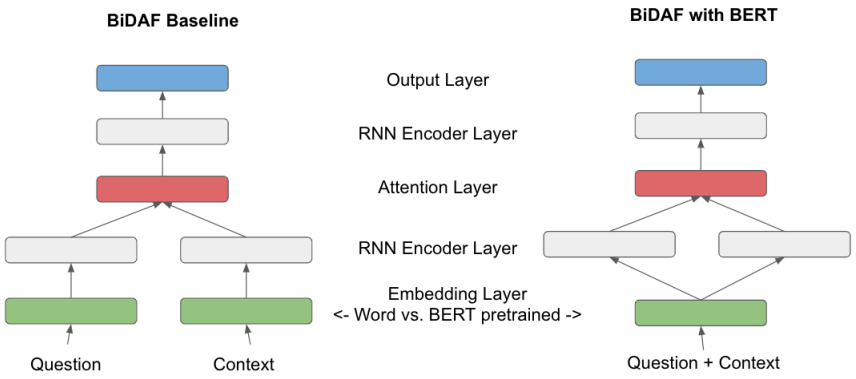

Figure 1: BiDAF architecture modification using BERT embeddings.

combinations may be better for text generation. Thus, we extend upon the work of the BERT paper by looking at the impact of different embedding combinations specifically for the task of QA.

3.2 Industry Application

Benefit Cosmetics is an international company that receives hundreds of questions daily to its Facebook page and Instagram account, and even more to its customer support email. The company is exploring ways of automating some of these responses. Working with industry data poses unique challenges, including brand preservation. The company currently uses manual templates that a support representative copies and pastes.

Our Original Contribution: We investigate the best ways to translate SOTA QA models to this industry pain point. We generate dev/test sets using our partner's custom data (see section 4.1) and tune/evaluate our models and the baseline models on these sets in addition to submitting to the SQuAD leaderboard. We then provide analysis on the risks and strengths of automation as well as analysis on infrastructure needed to productionize such models.

3.3 BiDAF

We modify the provided baseline BiDAF as shown in Figure 1. The provided baseline takes as input a question and context, which it converts to word embeddings. Instead, we take the combined question and context and convert them to BERT embeddings before passing them on to subsequent layers of the BiDAF model. Input embeddings for a given batch are generated during the train step, before the forward pass of the BiDAF model. Consequently, we investigate how readily useful the BERT embeddings are for generic models.

4 Experimental Setup

4.1 Data

We trained the varied BERT + BiDAF models introduced above (sections 3.1, and 3.3) on 2 datasets. The first is the default project SQuAD 2.0 dataset [13], and the second is a custom Benefit Cosmetics dataset (described in sections 4.1.1, and 4.1.2).

4.1.1 Custom Benefit Cosmetics Dataset

The custom dataset is constructed (following steps in section 4.1.2) from csv files that were provided to us by Benefit Cosmetics [11]. These files contain $40,000$ messages, within about $20,000$ threads of Facebook and Instagram direct message (DM) interactions (Benefit Cosmetics US and UK accounts). See figure 2 for some hand-picked example direct message interactions. As a quick note, not all

questions have responses from a company account, and such questions are immediately followed by a different message thread from another user.

In each file, the message text is in one column, and the corresponding sender IDs (either the company account or the consumer user ID) are in the other, for a total of 2 columns. Users mostly ask about price, product-related complaints, and shipping availability. Additionally, many of these messages contain slang and abbreviations. Question structure is inconsistent. For example, sometimes a question is asked in the middle of a sentence, other times at the end, sometimes the input lacks a '?' character, and some examples are much longer than others. Given that this is real world data, it makes sense that the rhetoric is much more conversational, although this definitely comes with challenges (see relevant discussion in section 6.3.2).

4.1.2 Indusry Preprocessing Architecture

To get the csv files into a format that was readable by our model, we created a preprocessing pipeline that wrote the contents of each input csv into a JSON file that had the same structure and nested sub-fields as the official SQuAD 2.0 train and dev sets.

This involved the following multi-step process. First, the raw Benefit Cosmetics csv is ingested and some specifically designed heuristics are used to bucketize these DMs into Question, Context, and Answer (QCA) triplets. A percentage of these QCA triplets (randomly subsampled) are mixed or spliced together with the official SQuAD training dataset, and the remaining triplets are mixed with the dev set. The hybrid result is then written to new train and dev JSON files, which now consist of examples from both the Benefit Cosmetics dataset and SQuAD 2.0.

Examples of rules or heuristics we used to construct accurate QCA triplets included checking if the message contained a '?' character, splicing out from the candidate contexts what seemed to be templated phrases by Benefit Cosmetics like 'Hey gorgeous' or 'xoxo', and extracting the 'ground truth' answer (ideally 3 to 5 words) from the context by extracting all text up to the first punctuation as the golden answer. Right off the bat, there were plenty of compromises we had to make; limitations of some of these preprocessing decisions are discussed in section 6.3.1.

4.2 Evaluation Method

For evaluation we report our best F1/EM scores on SQuAD 2.0 dev/test datasets and provide qualitative analysis on the custom industry datasets for each of our models (BiDAF baseline, BERT fine tune, and modified BiDAF models).

4.3 Embedding Experiments

To give our combined BERT embeddings and BiDAF model the best chance of achieving high scores, we performed a feature-based test similar to the one conducted in the original BERT paper [1]. We examined three high performing combinations of embedding vectors generated from from the pre-trained Transformer's hidden layers. Specifically, we extracted 1) just the first layer, 2) just the last layer, and 3) the sum of the last 4 layers. (see figure 3) [1]. These embeddings were fed as input into our modified BiDAF model, and the performance was compared to our vanilla BiDAF and default BERT fine tune outcomes in Section 5.

Vanilla Baseline Submission Our first leaderboard dev submission obtains an EM/F1 of 57.906/61.493, which improves upon the unmodified baseline scores of 55/58 (as stated in the handout). This required no code modifications, and we simply used the provided GloVe word embeddings.

BERT Fine Tune We fine tuned the huggingface Pytorch implementation of BERT on the SQuAD 2.0 dataset for 2 epochs using a learning rate of 3e-5 and doc stride of 128. Due to memory constraints, we restricted the max sequence length to 384 and the batch size to 6.

BiDAF with BERT Embeddings Models were trained using the same hyperparameters as the baseline BiDAF model, with the exception of num_epochs, which we set to 2 in order to compare performance with the Bert fine-tuned run. Due to memory constrains on the NV6, we had to limit the batch size to 3 and the max sequence length to 96. The doc stride was left at 128, as was the default in the BERT fine-tune run.

Q: *How much does Benetint cost?*
A: *Hi Ammara! Our benetint cheek & lip stain is $30 USD! xo https://www.benefitcosmetics.com/us/en/product/benetint-cheek-lip-stain?gclid=EAIaIQobChMIr8bspIe84AIVB9lkCh2RZAEYEAAYASAAEgIr6vD_BwE*

Q: *Can you recommend something for me*
A: *No Answer*

Q: *Hello, I would like to know if the Lizy P eyebrow kit is available in Bolivia?*
A: *Hi gorgeous! Unfortunately it is not, but thank you for reaching out. To find the closest Benefit boutique location near you, please email us at customercare@benefitcosmetics.com! xo*
Q: *Do you have branches in Jordan?*
A: *Hi there! Unfortunately we no longer ship internationally but please email us at customercare@benefitcosmetics.com to find the Benefit location closest to you! xo*

Q: *Hey I love your product I use the brow wiz pencil it's great my only thing is it runs out so much faster than any brow product I've probably ever use. I use the gel but my brows still seem to fade by the end of the day.*
A: *Hi gorgeous! Thanks so much for sharing your feedback -- we also recommend using our BROWVO conditioning primer to help brow product stay!*

Figure 2: Examples of Benefit Cosmetics direct message interactions.

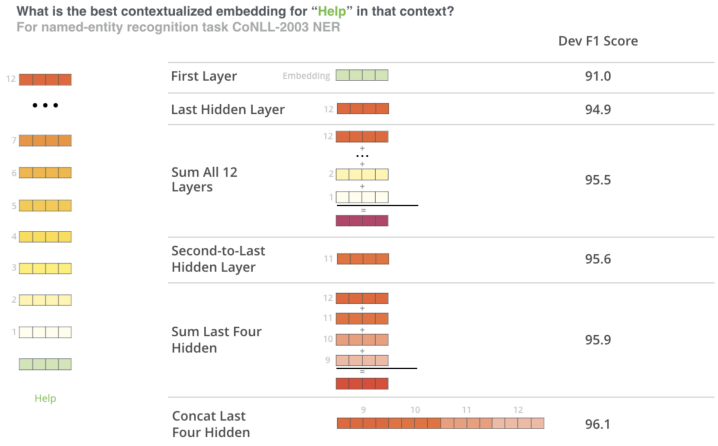

Figure 3: The different word embedding vectors, generated by concatenating different combinations of the pre-trained Transformer's hidden layer outputs [5].

Industry Dataset Training To get as close of a comparison as possible to our other experiments, we tried to mirror the hyperparamaters of previous experiments (while still accounting for memory constraints) and only swap out the default SQuAD 2.0 dataset for the hybrid dataset constructed as described in section 4.1.2. We fine tune the huggingface BERT model on this dataset as an upper bound comparison, and we then train our modified Bert Embeddings + BiDAF models on this hybrid dataset and examine the output. An NV6 machine was used for 2 epochs, along with a batch size of 3, a max sequence length of 96, and a doc stride of 128.

5 Results

Table 1: Embedding Experiments

Model	EM	F1
BERT Fine Tune Reported [9]	82.126	84.820
BERT Fine Tune Empirical	73.001	76.069
Baseline BiDAF Reported	55	58
Baseline BiDAF Empirical	57.906	61.493
Last Layer 2 epochs	53.241	53.870
First Layer 2 epochs	51.629	51.686
Sum Last Four Layers 2 epochs	52.929	54.895

Table 1 outlines EM and F1 scores for all of our embedding experiments. “Reported” indicates the published result in the original document. "Empirical" indicates the value we observed during our run. All values are reported on their respective dev sets.

We also fine tuned BERT on our industry + SQuAD 2.0 mixed train set and mixed dev set, both containing a combination of research and industry examples. The fine tune achieved results similar to the BERT Fine Tune Empirical model in Table 1 (around 70/75 EM/F1). However, when we ran our BERT embedding + BiDAF model on the mixed industry/SQuAD train/dev sets, the EM/F1 were less than 10. Potential reasons for this behavior are discussed in our qualitative analysis.

Training of our modified BERT + BiDAF models took about 12 hours per epoch, which is significantly more than the 22 minutes per epoch of the vanilla BiDAF baseline and almost double that of the fine-tune BERT base model on an NV6 machine with a single GPU.

Best submission to Test and Dev Both were from the full end to end fine tune of the huggingface submission. Best dev submission: 73.001 EM 76.069 F1 Best test submission: 72.257 EM 75.586 F1

6 Analysis

6.1 Quantitative Performance

Each of the embedding experiments achieve EM and F1 results comparable to the vanilla baseline within just 2 epochs instead of the default 30 used for training the baseline (although at the cost of much longer training times for the non-baseline models). However, the model that very clearly outperformed the rest was the fine tune of BERT. We were able to reap the benefits of the architecture calculating the embeddings by calling a forward pass of the BERT model. This embedding pre-generation in the processing step led to a significant reduction of the previously 12 hour/epoch train time. Our results therefore indicate that the embeddings might assist in faster convergence, but they do not suggest that we can achieve the same state of the art performance of original the original BERT model simply by substituting embeddings.

Comparing our embedding experiments (last layer, first layer, sum of the last 4 layers) we saw that the experiment that produced the best EM score involved the last layer, and the one that produced the best F1 score was the sum of all 4 layers.

The models trained on the hybrid industry datasets seem to get comparable EM and F1 scores, but this may be because the model is simply ignoring most of the industry examples (further discussed in the qualitative section below).

6.2 BERT Embeddings in BiDAF Qualitative

Each of the BERT + BiDAF experiments did well with proper nouns (names, locations, etc.) and numerical answers such as dates but both the BERT + BiDAF and baseline models tended to err on the side of “no answer” for more complex questions. This indicates that several of the core properties of the models architecture may be preserved despite the embedding replacement. Some examples are included below.

Table 2: Sample Outputs for BERT Embeddings

Question	Prediction	Expected
In what country is Normandy	"France"	France
When was Scotland invaded by William?	"1072"	1072
What major conquest did Tancred play a roll in?	""	Jerusalem

This is likely because the embeddings really are just a better representation of input, but have no direct impact on subsequent layers. Thus the model may have a better starting point from which it can learn and thus reach comparable results in fewer epochs, but the underlying architecture will still learn the same macro trends. The ability to detect proper nouns and dates likely comes from the fact that the key word in the question or a close synonym is located in the context. E.g. for "In what country is Normandy" the notable context sentence is "Normandy is a region in France" where country and region are very related. However when trying to decipher the phrase "Tancred was instrumental in the conquest of Jerusalem" for question 3, the model struggles to equate "instrumental" with "play a roll in." The sentence structure is slightly different in the question and the key context phrase, and thus the model assumes there is no answer available. These models may improve with longer train times.

6.3 Industry Qualitative

While working with industry datasets has exciting potential real-world applications, we definitely ran into our fair share of challenges. These challenges can be attributed mainly to decisions made in the preprocessing pipeline (section 4.1.2) as well as the nature of the data (section 4.1.1).

6.3.1 Preprocessing

As the data we received was in the form of csv files (not quite compatible with our models), we had to come up with heuristics in order to figure out which pieces of data we could use in our model. The heuristics we described in section 4.1.2 did well to extract somewhat accurate QCA triplets, however some very evident preprocessing errors did occur, which almost definitely had a negative impact on the final performance of the model. For example:

Q: "do i have to pay in dollars? im from the philippines. where are benefit cosmetics available?"

A: "hi... We are Turkey based company.we have all natural product range. we have a wide range of handmade natural soaps, olive oil based beauty and bodycare soaps, massage oils,massage creams,body lotions,body butter,haircare products, clay masks, world- known Turkish rose products and many other beauty products.Based on the quantity we can do private label production. We export to 10 countries. We look for new markets and new customers. Regards, http://www.hamamtamam.com/

Note that there are 2 questions asked, and the answer ignores the first. This is just one example of the challenges of parsing.

6.3.2 Nature of the Data

Other challenges arose as a result of this data being in many ways over-conversational (slang, abbreviations, etc.). For example, the abundant misspellings lead to <unk> characters being much more likely in the default model, thus warranting the use of character or sub-word embeddings. When these sub-word embeddings were implemented via our BERT + BiDAF architecture however, the misspellings often led to non-sensical subword tokenizations that hindered model performance.

We did see relative success of the model on user questions related to price as well as shipping availability (product complaint related inquires usually led to a “no answer”). Example of successes below:

Q: "hi.. for how much u r selling the holiday set?" A: "$72"

Q: "What is the price of the POREfessional value size kit?" A: "$54"

A quick caveat was that sometimes the company representative would simply respond to these price or product questions with a link to the product page, which made training a little noisy. Even so, these questions comprised roughly 15% of all the QCA triplets, so while the overall performance

of our model on the industry dataset was unremarkable, it was actually quite promising to see good results on this narrow band of questions, since a system that automates responses to around 15% of direct messages could be very lucrative for a large corporation.

Another data-related challenge we observed was that in SQuAD, most questions are a single sentence with a much longer context from which the answer can be found. In most of these industry examples however, the question is actually much longer than the context, and it often contains a lot of irrelevant information that may impacting the ability of the model to learn good answers. In fact, the state of the art unmodified BERT model predicted "no answer" for a large portion of the industry questions in our mixed dev dataset, which exposes a potential shortcoming in models trained on SQuAD. Such models may only be extremely effective when the question/context/answer formats match almost exactly the research formats, which is rarely true in real settings. Thus, the translation of BERT to widespread industry application may require significant per-company research, depending on their data distribution.

Of note though, our BERT with BiDAF model did much better at avoiding a default "no answer" when compared to the vanilla BERT model, with 50% of our industry-only dev set achieving viable predictions. However, some of the model's attempted responses to some of the more complex questions started with random letters or characters like ',', '.', '?', most likely resulting from misspellings, slang, and preprocessing decisions.

6.4 Challenges of Using Pre-trained BERT embeddings

The BERT paper suggests that its embeddings can be used essentially out of the box for any NLP task, but as a guest lecturer noted there were a few unforeseen challenges in making the pre-trained embeddings compatible with a representative model such as BiDAF.

The first challenge was in tokenization. The BERT tokenizer does not simply split by word, it further splits input into "word pieces" for tokenizing. The challenge for us was then mapping the y1s and y2s from the BiDAF input processing to the correct start and end indices within the BERT tokenized output. For example, if the sentence [i love learning] has y1 and y2 (0, 1), but BERT tokenized the input to [I lo ve learn ing] after the preprocessing and data loading but before generating embeddings, the target y1 and y2 would be incorrect for this example. Furthermore, the provided BertTokenizer tokenization does not have a built in batch tokenization capability in the Pytorch repo, and thus cannot be used directly in forward passes of existing models. At first we attempted preprocessing from scratch, but the level of involvement and the discrepancies between the BiDAF and BERT representations led us to model our preprocessing after the huggingface run_squad.py [2].

The second limitation lies in the embedding size. The default 768 may have worked well on TPUs with terabytes of memory, but holding large batches with these embedding dimensions and training within a reasonable amount of time proved challenging for our combined implementation. To make our model compatible, we set the hidden size of 768 for all subsequent layers, but this limitation may not be best for all cases. Moreover, when we tried to concatenate the last four layers of BERT output as shown at the bottom of Figure 3 before passing into the RNNEncoder we ran out of memory with a batch size of just 1, since the hidden size of 3072 was simply too large for NV6 machines.

7 Conclusion and Improvements

While the BERT embeddings may be powerful, they are not quite "plug and play" with models that assume GloVe embedding input. They may be effective as more developers start to use the embeddings and thus build more standard processing pipelines and memory optimizations, but for now, obtaining the reported performance may require significant additional preprocessing, prohibitively expensive resources, and modifications to a model's architecture. The models were slow, but we believe the runtime could be significantly decreased by moving the BERT embedding generation from the train step to the preprocessing step. Despite drawbacks, BERT embeddings seem like a potential way to speed up model convergence. Additionally, we could have greatly improved our scores on industry data with more structured responses, fewer typos, and a clear bank of answers. However, these changes would require a lot of overhaul on the part of Benefit Cosmetics. This project highlights the importance of properly structured data when attempting to apply NLP research to industry problems and the resource challenges that come with adapting BERT embeddings to custom use cases.

8 Appendix

We want to thank Toto Haba and his team at Benefit Cosmetics [11] for providing us with company-specific datasets that allowed us to tackle this problem of testing the limits of SOTA QA models in an industry setting.

References

[1] 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.

[2] huggingface, The Big--Extending-Repository-of-Transformers: PyTorch pretrained models for Google's BERT, OpenAI GPT and GPT-2, Google/CMU Transformer-XL, 2018, Github repository, https://github.com/huggingface/pytorch-pretrained-BERT

[3] Adams Wei Yu, David Dohan, Minh-Thang Luong, Rui Zhao, Kai Chen, Mohammad Norouzi, and Quoc V. Le. 2018. Qanet: Combining local convolution with global self-attention for reading comprehension. arXiv:1804.09541.

[4] BangLiu, Re-implement "QANet: Combining Local Convolution with Global Self-Attention for Reading Comprehension", 2018, Github repository, https://github.com/BangLiu/QANet-PyTorch

[5] Jay Alammar. The Illustrated BERT, ELMo, and co. (How NLP Cracked Transfer Learning). http://jalammar.github.io/illustrated-bert/

[6] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global Vectors for Word Representation.

[7] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S. Corrado, and Jeff Dean. "Distributed representations of words and phrases and their compositionality." In Advances in neural information processing systems, pp. 3111-3119. 2013.

[8] Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. Bidirectional attention flow for machine comprehension. arXiv preprint arXiv:1611.01603, 2016.

[9] SQuAD Leaderboard. https://rajpurkar.github.io/SQuAD-explorer/.

[10] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In Advances in Neural Information Processing Systems, pages 6000–6010.

[11] Proprietary datasets (csv files) from Toto Haba, SVP, Global Digital at Benefit Cosmetics.

[12] Preprocessing script to make a custom SQuAD dataset (convert a csv containing Question, Context, and Answer triplets to a JSON, formatted like SQuAD 2.0) written with the help of Chris Chute.

[13] Chris Chute, Starter code for Stanford CS224n default final project on SQuAD 2.0, Github. https://github.com/chrischute/squad