Supervised Fact Extraction from Web Pages

Tom Sandell
tsandell@stanford.edu

Waleed Malik
wmalik@stanford.edu

Junaid Ahmed
junahmed@stanford.edu

Mentor: Pratyaksh Sharma

Abstract

The goal of our project is to recreate Wikipedia info boxes using the main text from Wikipedia pages, using a question answering framework. This model could then be expanded to extract facts from pages without info boxes, creating in essence a fact extraction framework for Wikipedia pages. By converting the information found in Wikipedia info boxes into the format of the SQuAD data set, we create a model that uses the BERT language representation to perform question answering and extract relevant facts from Wikipedia passages. Using the pre-built info boxes as ground truth data to train our model, our model takes a property (such as "birthdate") as a question for a passage and will return the value (such as "December 14") as the answer. We originally wrote a simple model using the BERT model followed by a linear layer, and evaluated this based on exact match and F1 score against a baseline that we created by writing specific models for properties in the info box. We then created more complicated models on top of the BERT model, and evaluated these models against this original simple BERT model. We found that using a question answering framework and a model using BERT and a highway layer, we were able to achieve an F1 score of 90.6, which beat our baseline model which had an F1 score of 42.7, while also improving recall because this works on all properties on Wikipedia. We found that adding a CNN on top of BERT also performed well, but was not as well as a simple linear layer or a highway layer. Finally, we found that increasing the passage length of a Wikipedia article increases both precision and recall of the facts we found.

1 Introduction

With the advent of intelligent agents and smart speakers, the consumer expectation around search and knowledge discovery is getting close to human cognitive ability. The ultimate digital assistant would be able to answer any question on the internet. This ability is a monumental step in machine-reading-comprehension (MRC) and to get there, we need to win in several smaller areas. First, we'd need to understand concepts and properties from webpages in isolation. Second, we'd aggregate the information across various pieces of data. From there, the agents need to understand the concept of authority, trust, or freshness: for example, if the same question can be answered by Wikipedia or FunCoolFacts.com, the agent should pick the more authoritative and trusted source. In this project, we aim to lay the building blocks of the first step, by gaining the ability to return specific properties given unstructured text.

For most of the research in this area, the approach has been to apply pattern matching, regular expressions, and NER to build known properties, as can be seen in Amira Abdelatey's paper, *Semantic Data Extraction from Infobox.* The drawback of these approaches is that they are especially brittle. When information is represented in a different format the regular expressions break easily. For example, a model can be trained to extract birth date from text such as "John X (1958-2010) was an influential..." and "John X was born in the year . . . " but would likely fail on an indirect reference

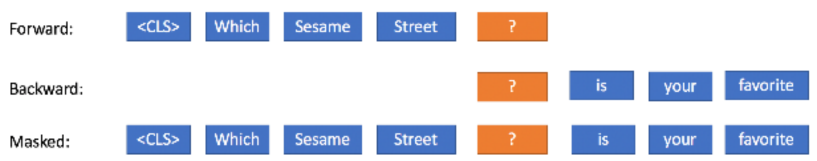

Figure 1: Bidirectional learning used by BERT

such as "In the land of middle year, near the small town of Baidu, the village welcomes a leader of its ruler family in 1958...". These models also require a lot of initial effort to get up and running. For each fact, there is a custom set of handwritten rules that needs to be tested thoroughly, which limits the number of facts that we are able to extract.

These approaches are also task specific and generally don't do quite well in corpuses of different domain whereas the motivation of our modelling work is that we should see transfer learning in action: the model training in one area should generally apply across all domains. Although we train only on Wikipedia pages, hopefully it should also work on FunCoolFacts.com.

2 Related Work

In Fader, Soderland, and Etzioni's paper *Identifying Relations for Open Information Extraction*, the authors go into detail about open extraction systems where facts can be extracted without a pre-specified vocabulary. The approach is novel in the manner that it does not require a lot of training data, however the approach is fundamentally limited as it lays down all possible inferences directly from sentences and gives them confidence from a simple logistic regression classifier. It is a robust and comprehensive approach for simple stated facts; but when information has to connected through words that are not appearing next to each other or even in the same sentence, the approach will fail. The approach also fails fundamentally as it is impossible to lay down all possible facts in advance; whereas our approach encodes the information just as the human brain encodes the information and retrieves it with the right context.

Open Information Extraction: The Second Generation by Etzioni et al. is an improvement to the reverb approach above, but it still is a collection of facts pre-processed and saved from text to refer to in question answering. It is an improvement in terms of leveraging CRF classifiers, sentence boundaries and takes a relationship pair as input; but still does not fundamentally point of connecting information from different parts of the text and the required relationship should be fairly obvious in the text.

Teaching Machines to Read and Comprehend by Hermann et al. comes closer to the deep learning way of approaching the semantic understanding and goes in-depth on deep LSTMs. The authors show that deep LSTM encoders preserve knowledge connection through longer sequences. The core concept of bidirectional encoding is the foundation of our BERT approach; but the approach falls short from constrained self-attention where every token gets encoded information around long sequences of contextual words.

Attention Is All You Need by Vaswani et al. really establishes the case of self-attention and transformers. The same idea is matured in BERT; but this paper originally set the course of events. The model laid out here has the fundamentals of BERT, but the model is not bi-directional and there are fewer layers. BERT's input embeddings are also more robust, as sum of token embeddings, segment, and position embeddings and transformers are a deeper interconnection than the various multi-head and dot-product attention. However, the core of the ideas are along similar lines.

Finally, the approach we are iterating on comes from *BERT : Pre-training of Deep Bidirectional Transformers for Language Understanding* by Devlin et al. BERT is the authority in modern day deep learning where the state of the art results on all NLP exercises of SQUAD are achieved. Self-attention over a large set of layers of bidirectional transformers get the best encoding of each word across the sequence and leads to best contextual retrieval of questions through semantic representation. The crux of the novelty is that BERT takes forward and backward sequences into each layer simultaneously and that leads to improvement in performance, as can be seen in Figure 1; neither LSTM nor self-attention had ever applied this left-right-simultaneous attention at all layers.

3 Approach

The foundation of our approach is leveraging BERT to extract properties from free-form text. As discussed, we chose to use BERT because it is the authority in modern day deep learning, and it was the best model which performed on the SQuAD and GLUE benchmarks. We then decided to use a question answering framework to extract facts, by using a property such as “birthplace” as a question, where the answer is the fact itself (such as “Detroit”). This is discussed more in section 4.1. We adapted BERT to the SQuAD question answering framework by using a variant of BERT which combines the BERT model with an additional layer at the end to predict start and end of positions of the answer in the text itself. While training, we would fine tune the preexisting BERT model for our problem and dataset, and completely learn the weights of any additional layers. We only fine tuned BERT because retraining BERT is iteratively a very lengthy process and that we would not be able to finish our experiments by the project timelines.

3.1 Baseline

As a baseline, we implemented a much simpler rule and context based model to find specific properties. We created models to extract four properties from Wikipedia pages: name, date of birth, date of death, and birthplace. We used nltk and Stanford NER Tagger to find the name and the birthplace of a given entity from the webpage, and used datefinder to find the date of birth and date of death. For example, to find the property "birthplace," our model would use Stanford NER Tagger to tag a location, and would return that as the answer. Instead of trying to write a new model for each of the thousands of properties on Wikipedia, we only compute the score of our baseline on the four properties that we model. This model is not particularly robust; it does very well with standard formatting, but if the formatting differs (for instance, if birth date is shown before death date), it breaks. This model’s recall is also not very good because we need to create specific rules per property. We expect our model to be more resilient and to have a higher recall. Once we implemented a model that used BERT with a simple linear layer to solve this problem, we tested this against the baseline and found that this performed significantly better. From that point forward, we used this primitive BERT model as our baseline to test our more complicated models against.

3.2 Fine Tuning BERT with Linear Layer

The simplest version of our model consisted of the BERT model with a single linear layer on top of it, which converted the BERT outputs into the SQuAD style question answering format. This can be seen in Figure 2. We used the BERT pretrained layers from bert-based-uncased, and the linear layer predicted start and end position. This was originally proposed as one of the tasks BERT did well on when finetuning. This approach was a great starting point and from there we modified the final layers, helped us learn more about our fact extraction task, and was the model that we used as our baseline to compare our more complicated models against. In this part of the approach, we also learned which hyperparameters worked best for the BERT model itself (for instance, batch size, learning rate, or sequence length). Particularly, we were curious whether increasing the sequence length would improve or hurt model performance, and we studied this for all models we implemented.

3.3 Fine Tuning BERT with CNN

The next experiment we did was to replace the linear layer with two CNNs, one to predict the start and the other to predict the end of the answer span. This model can be seen in Figure 3. Convolutional neural networks reduce the dimensionality of the input layers and transform towards the target dimensionality. This means that the CNN was faster than our linear layer to train, and we also wanted to see if by finding the spatial relationships of adjacent words, we could train to find the answers to facts with a higher precision. We also added a RELU non linearity and a pooling layer to the CNN’s. We tried both max and average pooling as the final pooling layer, but max pooling performed best for our task, as you will see in section 4.4. We assume the reason max pooling worked better is because words such as "born" or "died" are strong indicators that a fact will be close by. With average pooling,

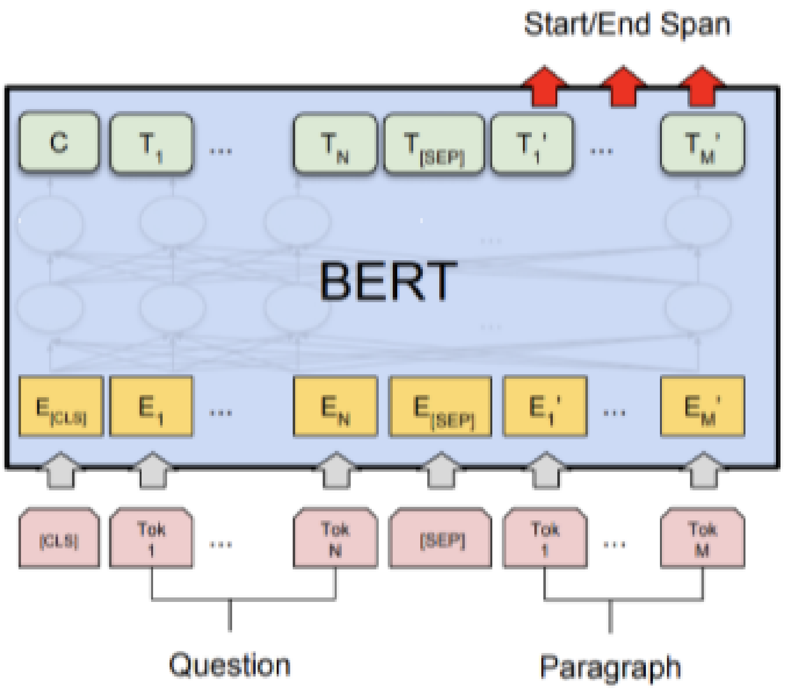

Figure 2: BERT finetuned with Linear for Question Answering

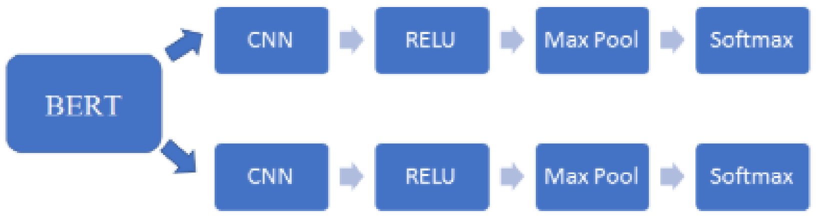

Figure 3: BERT finetuned with CNN

we may lose some of these higher signaling words, as we take in to account each filter instead of simply the most important filter. A drawback of our CNN model is that as the network is not fully connected, we lose some context from multiple steps, so the multi-step information flow is broken.

3.4 Fine Tuning BERT with CNN and Linear

To address the lack of a fully connected layer at the end, we tried removing pooling and replacing it with a linear layer. This can be seen in Figure 4. This model would take longer to train than the CNN, but we were hoping that it would expose relationships between filters in a better way than average pooling did, and would solve the issue of our CNN based model losing context from BERT. However, we run the same risk as we did in average pooling that the most important filters could be "watered down" by less important filters.

3.5 Fine Tuning BERT with Highway

Our last and best (which we will discuss in section 4.4) core model change experiment was adding a highway layer to BERT. This can be seen in figure 5. The flow of our model went from the BERT model to three additional layers: two linear layers and a single highway layer. The goal of the

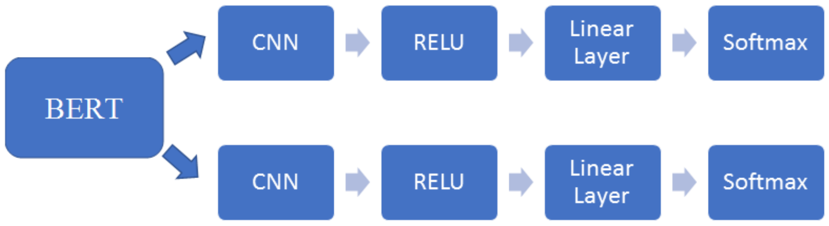

Figure 4: BERT finetuned with CNN and Linear

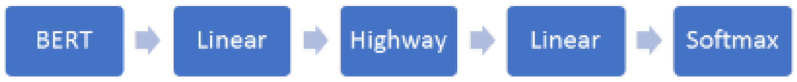

Figure 5: BERT finetuned with Highway

highway layer was to provide a direct connection of neurons between the two linear layers, so that we could learn additional weights in the middle layer without losing too much information. We were worried about adding too many layers on top of BERT, as BERT is already state of the art and we didn't want to lose important information from its embeddings. Thus, a highway model was a way to add an additional layer without losing information. As a side effect, this layer was also the most complicated model we trained, and it had the highest number of layers, which also created a deeper network which allowed us to learn more weights for identifying key patterns in fact extraction.

4 Experiments

4.1 Datasets

We found a data set created by the Linguistic Data Consortium called the TAC KBP Reference Knowledge Base and were able to get the license to use this data through Stanford. This data, based on a 2008 snapshot of Wikipedia, contains a set of entities with a name and title, the parsed data from the info box, and a stripped version of the actual text from the Wikipedia website. We preprocessed this data to find the properties from the info box that we would also be able to find in the Wikipedia text itself. This included removing facts that aren't found in the text; for instance, often people have a birthday listed in their info box but not mentioned in the plain text. For training purposes, we also truncated Wikipedia passages to only a certain number of characters, either 200 characters for faster iteration or 500 characters. Next, we formatted this data so that it would match the format of the SQuAD data set, which included finding the index that the "answer" to a fact is located in the text passage. In cases where the "answer" to the fact was located multiple times in the Wikipedia passages, we assumed the index was the first position in which the "answer" was found.

4.2 Evaluation Method

Like most SQuAD models, we evaluate our model based on both exact match and F1 scores. The exact match calculation is exactly what it sounds like – we check to see if the answer that we give is an exact match of what the answer is in the info box. The F1 score is looser, as it simply looks for the average overlap between our response and the answer in the info box. This score is the average between the precision and recall of our answers, where precision is defined to be the percent of words in our answer that are also in the ground truth answer, and recall is defined to be the percent of the words in the ground truth answer that are found in our answer. For example, if the ground truth answer is "Los Angeles, California" and our model predicted "Los Angeles," we'd have high precision but lower recall, but if we predicted "Los Angeles, California, US," we'd have high recall but lower precision. This example also shows why the F1 scores are necessary, as facts often don't have just one ground truth answer- both of those answers would be given a score of zero for exact

	Exact Match	F1
Baseline	30.78137	42.65923
BERT and Linear Layer	84.47493	87.75892
BERT and CNN: Avg Pooling, Kernel 5	77.28237	82.18032
BERT and CNN: Max Pooling, Kernel 3	82.02515	86.09059
BERT and CNN: Max Pooling, Kernel 5	83.08998	86.86971

Table 1: Exact Match and F1 Scores for various models with passages truncated after 200 characters

	Exact Match	F1
BERT and Linear Layer	84.48681	90.37682
BERT, CNN, and Max Pooling	83.12102	89.51992
BERT, CNN, and Linear Layer	83.07775	89.45088
BERT and Highway Model	84.77079	90.62593

Table 2: Exact Match and F1 Scores for various models with passage length 500

match. However, these answers are mostly correct, and our model should be given some credit for them. Thus, we focused more on improving F1 score than exact match.

4.3 Experimental Details

We implemented each model on top of BERT in three steps—first, as a sanity check, we trained the model on a tiny dataset of 200 passages of length 200 characters and evaluated on 50 passages. This sanity check never resulted in high f1 scores, which was expected, as 200 passages were way too few to properly train our models as they had a significant number of weights, but it ran within 30 seconds, which allowed us to quickly find coding errors such as dimensionality issues. Next, once we were confident in our model, we would train our model on a dataset consisting of all of our passages trimmed after 200 characters. We ran this with a maximum sentence length of 384 words. This would take between 2 and 8 hours, and we were able to tune hyperparameters and model decisions based on the results of this test. For instance, we used this step to learn that max pooling worked better than average pooling when implementing a CNN, and that the ideal kernel size of the CNN in our scenario was 5. Finally, once we were confident in all the details of our model, we tested our model on all passages trimmed after 500 characters, with a maximum sentence length of 500. At first, we tried using a maximum sentence length of 384 for this set as well, but our model would simply break - it would end up with exact match scores equivalent to simply guessing two random numbers as the beginning and end of the passage. Thus, we learned that we had to increase the maximum sentence length of the model. There were 895,000 passages in our dev set and 37,000 in our test set, and it took between 12 and 48 hours to run, depending on the model. We trained all of our experiments on a 24 core, 4 GPU Azure VM after implementing our models locally, and then trained for 2 epochs with a learning rate of 3e-5. We were able to use a similar approach with all of our models because each of our models took in the same input (the embeddings from fine-tuned BERT) and the same output (logits which were then converted to the start and end positions of the answer). We used the bert-base-uncased pretrained weights for our BERT portion of the model, that were downloaded from the original BERT paper. Bert-base-uncased contains 12 layers, 768 hidden states, 12 heads, and 110 million parameters. We would like to have used bert-large-uncased for our model but we did not have the compute to finish this efficiently as it is 24 layers and contains 3 times as many parameters as bert-base-uncased.

4.4 Results

Even when we ran our model on passages truncated after 200 characters, it was clear that the BERT based model significantly outperformed our baseline model, as can be seen in Table 1. Simply running BERT fine tuning with a linear layer afterwards to convert the output into question answering format vastly outperformed our baseline. This was already much better than we expected, and encouraged us to try to implement other models to try to beat out this basic BERT model, which in essence became our new baseline. We also used these passages to flesh out the implementation details of many of our models; it was here that we learned that max pooling performed better than average pooling, and that our CNN performed best with a kernel of 5. We also tested various learning rates, sequence lengths,

Question	Model Answer	Correct Answer
graduation year	1983	1983
harvard law enrollment year	1998	1998
district	13th	13th
primary competitor	Clinton	Hillary Clinton
chicago law school years	2004	1997 to 2004
best friend	and	?

Table 3: Example questions and answers for Barack Obama's Wikipedia Page

and number of parameters for our layers on this dataset. Some of these tests we ran are shown in Table 1, to give a better understanding of why we implemented the models how we did.

Next, we ran our four main models on passages truncated after 500 characters, and found that this performed even better than on passages truncated after 200, as can be seen in in Table 2. This was somewhat counter-intuitive, as we expected that answers found in the first 200 words would be easier to find than answers found in the 201-500th word. However, because there were more training examples to train our parameters, these models performed the best of any we trained. This also tells us that our approach could be extrapolated to work on even longer Wikipedia passages, but we didn't have the time or compute power to run these experiments. As can be seen in Table 2, the model that performed the best was BERT with a Highway model on top, performing better than any CNN model or BERT with a linear layer on top. We were hoping that a CNN would work best, as there are no CNN's built onto BERT (whereas there are linear layer equivalents), and we thought that this would pick up on the important spatial relationship between adjacent words and would be able to look for distinct patterns in a sentence. However, because BERT is already bi-directional, this spatial relationship was already fairly fleshed out, and the CNN performed worse than a simple linear layer. In theory, the highway model works best because it is able to regulate the flow of information; however, since this was only model we trained with 3 additional layers on top of 12 BERT layers, we think that it might work best simply because it has the highest number of weights to train.

5 Analysis

One worry we had in our project was that our test and training data was weighted poorly. We were very concerned that although we had high F1 scores, this was because most of the properties in the first 500 words were some variation of "name"—which is easy to train (just choose the first couple words). Thus, we calculated the exact match and F1 scores of our best model, BERT with a highway layer, after eliminating any question that was a subset of "name." (Including not only "name," but also things like "player name" or "place name" which are also fairly easy to train.) Our model didn't do quite as well on this dataset, but still performed with an F1 score of 85.2774 and an exact match score of 77.10428, which meant we weren't completely overfitting to these properties. We also tested our model while normalizing the amount of times each property appeared in the test set, by setting a cap of each property at 20. We took this approach to make sure we weren't overfitting to non-name properties that appear in many info boxes, such as "hometown." This also performed quite well, with an F1 score of 83.59619 and an exact match score of 78.98918. Thus, we were confident that our model wasn't overfitting too much to the most popular questions.

Some of the questions that we performed worst on were queries such as "mascot" (exact match 66.6) and "starring" (exact match 54.17) where the words appeared in the text, but potentially not near much context. These were also somewhat "tail" queries, where we did not have much training data. Another category of questions we performed poorly on was specific numerical questions, such as "area water sq mi" (exact match 26.59). To improve these queries, we would like to train on more examples. The fact that we did the worst on was "country abbrev." There were 45 of these in our test set, and even our F1 score was 0.0. This, however, makes some sense because there is no context near "country abbrev" for our model to pick up on. This shows a limitation of our model, in that it can only find questions with context- even with a significant number of training examples for "country abbrev," our model couldn't figure it out. On the other hand, our model performed great on facts such as "discoverer" (exact match 96.92) and "birthplace" (exact match 98.78) where the question was almost always within a similar context.

We then created a demo where we could input any passage and question, including questions that were never trained on, to see how our model would work on specific data. Table 3 shows our output for Barack Obama's Wikipedia page [1]. As can be seen, this did extremely well, as it was able to answer facts such as "primary competitor" which we never trained on, and it could differentiate between "harvard law enrollment year" and "chicago law school years." However, this was not perfect, and was unable to find both the start and end of "chicago law school years." We think this was a limitation due to our training data, which almost never included an answer with two separate years formatted as YYYY to YYYY. Additionally, we were able to test our model on nonsensical input, and, as expected, received nonsensical output.

6 Conclusion

By adapting fact extraction to a question answering framework and using a fine tuned model based on BERT, we were able to train a model to predict the exact answers to info boxes in Wikipedia with 84% accuracy and an F1 score of 90.6, which greatly exceeded our expectations. We found that increasing the length of the passage improves the accuracy of the facts extracted from the passage, and that adding a highway layer on top of BERT can help improve the facts extracted versus simply adding a linear layer. In the future, we'd like to see if the correlation between sequence length and F1 scores continues if we greatly increase the Wikipedia passage size, or if this is simply because we are using fairly small passages to begin with. We learned that adding a CNN on top of BERT can work fairly well, but is not quite as good as simply adding a linear layer, as it loses some information—however, it trains much faster, so it can be a quicker solution. Our goal for this project is to expand it from here; we are currently choosing the questions to ask on the passages by looking at info boxes, but this is not a strategy that would work without very good training data. Thus, we would like to train another model that would predict which facts to extract from a passage. In conjunction, with our current trained model this would build an end to end fact extraction framework. Then, we could try to apply this to websites that aren't Wikipedia, to see how good this does across the web. This gets us closer to the vision of being able to answer any fact on the internet.

References

[1] Simpson, Heather, et al. (2014) *TAC KBP Reference Knowledge Base LDC2014T16.* Philadelphia: Linguistic Data Consortium.

[2] huggingface, 2019. pytorch-pretrained-BERT (Mar 2019). https://github.com/huggingface/pytorch-pretrained-BERT

[3] Horev, Rani (2018) *BERT Explained: State of the art language model for NLP.* Towards Data Science

[4] Devlin, Chang, Lee, & Toutanova. (2018) *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding* arXiv preprint arXiv:1810.04805

[5] Rajpurkar, Zhang, Lopyrev, & Liang. (2016) *Squad: 100,000+ questions for machine comprehension of text.* arXiv preprint arXiv:1606.05250.

[6] Stanford Natural Language Processing Group, 2019. Stanford Named Entity Recognizer (Mar 2019). https://nlp.stanford.edu/software/CRF-NER.shtml

[7] Natural Language Toolkit, 2019. NLTK (Mar 2019). http://www.nltk.org/

[8] PyTorch, 2019. PyTorch (Mar 2019). https://pytorch.org/

[9] Koumjian & Corradini, 2016. datefinder (Mar 2019). https://datefinder.readthedocs.io/en/latest/

[1]The Wikipedia passage we used for this test was "Obama was born in Honolulu, Hawaii. After graduating from Columbia University in 1983, he worked as a community organizer in Chicago. In 1988, he enrolled in Harvard Law School, where he was the first black president of the Harvard Law Review. After graduating, he became a civil rights attorney and an academic, teaching constitutional law at the University of Chicago Law School from 1992 to 2004. He represented the 13th district for three terms in the Illinois Senate from 1997 to 2004, when he ran for the U.S. Senate. He received national attention in 2004 with his March primary win, his well-received July Democratic National Convention keynote address, and his landslide November election to the Senate. In 2008, he was nominated for president a year after his campaign began and after a close primary campaign against Hillary Clinton. He was elected over Republican John McCain and was inaugurated on January 20, 2009. Nine months later, he was named the 2009 Nobel Peace Prize laureate."

[10] Abd El-atey, El-etriby, & S. kishk (2012) *Semantic Data Extraction from Infobox Wikipedia Template.* International Journal of Computer Applications (0975 - 8887)

[11] Fader, Soderland, & Etzioni (2011) *Identifying Relations for Open Information Extraction.* Empirical Methods in Natural Language Processing

[12] Etzioni, Fader, Christensen, Soderland, & Mausam (2011) *Open Information Extraction: The Second Generation.* 22nd International Joint Conference on Artificial Intelligence

[13] Hermann, Kocisky, Grefenstette, Espeholt, Kay, Suleyman, & Blunsom (2015) *Teaching Machines to Read and Comprehend.* arXiv preprint arXiv:1506.03340v3

[14] Vaswani, Shazeer, Parmar, Uszkoreit, Jones, Gomez, Kaiser, & Polosukhin (2017) *Attention Is All You Need.* arXiv preprint arXiv:1706.03762v5