Industrial Team Project Report

Children's Book Test

Authors:
Peizhen Bai, Mingjie Chen, Matthew Horrocks,
Jaspreet Lally, Yixuan Li, Li Zhu

Abstract

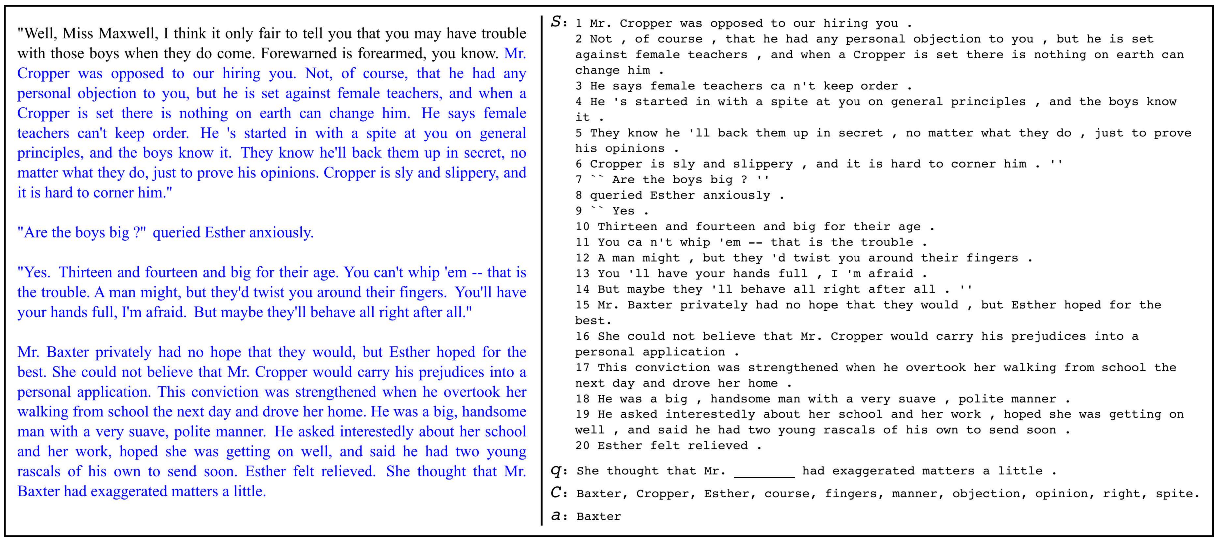
The Children's book test requires a computer to chose the correct word for a sentence, given some limited previous knowledge, which is a of large importance for natural language processing (NLP). The methods used to solve this problem include Random word, Most Frequent word, Recurrent Neural Network (RNN) language model, and the Trigram language model. The word to word version of the Trigram language model performed the best out of all the methods developed and tested. The improvement of computer systems and models in linguistic analysis is of large importance to NLP and the advancement of computer science work such as artificial intelligence.

Codes are available at:
`https://github.com/zhuli8805/CBT`
`https://github.com/superthierry/rnn_language_model`

We declare that this work is our own and we acknowledge the contribution of others where appropriate.

1 Introduction

One of the questions in the field of natural language processing is: what is the role of memory and context in language processing? The Children's Book Test (CBT) is a task designed to measure how well language models can utilize wider linguistic context. The task shows how well computer systems can interpret human speech and linguistic patterns. As such, this is crucial to advancements in computer science fields such as artificial intelligence and sentiment analysis. Books available from Project Gutenberg were used as the corpus from which the CBT is built, yielding our results (Facebook Research 2016).



"Well, Miss Maxwell, I think it only fair to tell you that you may have trouble with those boys when they do come. Forewarned is forearmed, you know. Mr. Cropper was opposed to our hiring you. Not, of course, that he had any personal objection to you, but he is set against female teachers, and when a Cropper is set there is nothing on earth can change him. He says female teachers can't keep order. He 's started in with a spite at you on general principles, and the boys know it. They know he'll back them up in secret, no matter what they do, just to prove his opinions. Cropper is sly and slippery, and it is hard to corner him."

"Are the boys big ?" queried Esther anxiously.

"Yes. Thirteen and fourteen and big for their age. You can't whip 'em -- that is the trouble. A man might, but they'd twist you around their fingers. You'll have your hands full, I'm afraid. But maybe they'll behave all right after all."

Mr. Baxter privately had no hope that they would, but Esther hoped for the best. She could not believe that Mr. Cropper would carry his prejudices into a personal application. This conviction was strengthened when he overtook her walking from school the next day and drove her home. He was a big, handsome man with a very suave, polite manner. He asked interestedly about her school and her work, hoped she was getting on well, and said he had two young rascals of his own to send soon. Esther felt relieved. She thought that Mr. Baxter had exaggerated matters a little.

```
S: 1 Mr. Cropper was opposed to our hiring you .
   2 Not , of course , that he had any personal objection to you , but he is set
   against female teachers , and when a Cropper is set there is nothing on earth can
   change him .
   3 He says female teachers ca n't keep order .
   4 He 's started in with a spite at you on general principles , and the boys know
   it .
   5 They know he 'll back them up in secret , no matter what they do , just to prove
   his opinions .
   6 Cropper is sly and slippery , and it is hard to corner him . ''
   7 `` Are the boys big ? ''
   8 queried Esther anxiously .
   9 `` Yes .
   10 Thirteen and fourteen and big for their age .
   11 You ca n't whip 'em -- that is the trouble .
   12 A man might , but they 'd twist you around their fingers .
   13 You 'll have your hands full , I 'm afraid .
   14 But maybe they 'll behave all right after all . ''
   15 Mr. Baxter privately had no hope that they would , but Esther hoped for the
   best.
   16 She could not believe that Mr. Cropper would carry his prejudices into a
   personal application .
   17 This conviction was strengthened when he overtook her walking from school the
   next day and drove her home .
   18 He was a big , handsome man with a very suave , polite manner .
   19 He asked interestedly about her school and her work , hoped she was getting on
   well , and said he had two young rascals of his own to send soon .
   20 Esther felt relieved .
q: She thought that Mr. _______ had exaggerated matters a little .
C: Baxter, Cropper, Esther, course, fingers, manner, objection, opinion, right, spite.
a: Baxter
```

Figure 1: Left: A book excerpt. Right: The question formed with 20 sentences of context, the query sentence, a list of candidates, and the answer (Hill et al. 2016).

Books were allocated into one of three corpus types: training, validation, or test. Example 'questions' were created from chapters in a book by enumerating 21 consecutive sentences, as shown in Figure 1. The first 20 sentences of the question form the context; the 21st sentence has a word removed, called the query. Computer models are tasked with identifying the answer word from 10 candidate answers that appeared in the context and the query sentences. Four classes of questions are contained in each corpus type:

Named Entities (NE), Common Nouns (CN), Verbs (V), and Prepositions (P) (Facebook Research 2016).

2 Background

2.1 Coreference Resolution

A coreference is formed when two or more mentions in a text refer to the same entity. Nouns and named entities are the most difficult to identity, but these types of words contain the most information (Raghunathan et al. 2010). An example of this from the news site Politifact is shown below:

"On May 21, [Trump] tweeted: 'Crooked Hillary said that [I] want guns brought into the school classroom. Wrong!' [He] then made a seemingly contradictory claim on Fox News the next day, May 22: [He] doesn't advocate for guns in the classroom, but sometimes teachers should have guns in the classroom." (Carroll 2016)

Coreferencing should match the word 'I' to 'Trump' and 'He' in the example. However, most models incorrectly link entities due to proximity and linguistic attributes such as the same part-of-speech (POS), contrary to human assessment. This is shown by how the word 'I' would be linked to 'Hillary' and not 'Trump' due to 'I' being in closer proximity to 'Hilary' than 'Trump'.

Previous work in coreference resolution includes the work by Raghunathan et al. (2010), a multi-pass sieve approach was used, applying tiers of coreference models from highest to lowest precision. Each tier builds on the previous tier's entity cluster output. This makes the approach highly modular since one can introduce new coreference models without affect the other methods.

2.2 Trigram Language Model

2.2.1 *n*-Gram Language Model

Statistical language models have commonly applied to many applications related to the estimate of probability of natural word sequences (Peng and Schuurmans 2003). The model is given a corpus consisting a word sequence $w_1, w_2, \ldots, w_T$. By the chain rule of probability, the probability of the occurrence of a sequence $w_1, w_2, \ldots, w_s$ can be denoted as:

$$p(w_1, w_2, \ldots, w_s) = p(w_1) \prod_{i=2}^{s} P(w_i | w_1, \ldots, w_{i-1}) \quad (1)$$

Since there are T^s possible combinations of words $w_1, w_2, \ldots, w_s$, it is computationally impracticable to compute all possible probabilities with realistic numbers of T and s, such as $T = 10000$ and $s = 10$, then $T^s = 10^{40}$. So, n-gram language models are implemented to further simplify equation (1) by assuming that the conditional probability of a word is only affected by previous $n - 1$ words, which is a Markov n-gram independence assumption

$$p(w_i | w_1, \ldots, w_{i-1}) = p(w_1 | w_{i-(n-1)}, \ldots, w_{i-1}) \quad (2)$$

Thus

$$p(w_1, w_2, \ldots, w_s) = p(w_1) \prod_{i=2}^{s} P(w_i | w_{i-(n-1)}, \ldots, w_{i-1}) \quad (3)$$

Despite some imperfections, an intuitive maximum likelihood estimate of $p(w_1 | w_{i-(n-1)}, \ldots, w_{i-1})$ for a given training corpus is

$$p(w_1 | w_{i-(n-1)}, \ldots, w_{i-1}) = \frac{count(w_{i-(n-1)}, \ldots, w_i)}{count(w_{i-(n-1)}, \ldots, w_{i-1})} \quad (4)$$

Where *count*() denotes the number of occurrences of a particular sequence in a training corpus.

2.2.2 Trigram Language Model

Because of the limitation of computing capability and time, a trigram language model is applied in this study. When $n = 3$, equation (3) can be rewritten as

$$\begin{gathered} p(w_1, w_2, \ldots, w_s) = \prod_{i=1}^{s} p(x_i | x_{i-2}, x_{i-1}) \\ x_0 = x_{-1} = * \end{gathered} \quad (5)$$

where $*$ is a special symbol 'START' that denotes the beginning of sentences, and

$$p(x_i|x_{i-2}, x_{i-1}) = \frac{count(w_{i-2}, w_{i-1}, w_i)}{count(w_{i-2}, w_{i-1})} \tag{6}$$

2.2.3 Smoothing Methods

Due to the heavy tailed nature of language, many $p(w|u, v)$ becomes zero due the number of occurrences of u, v, w are zero, which leads to the probability estimates being a sparse dataset (Zipf 1949). Thus, a finite training corpus is not likely to cover all trigrams encountered in queries. For any trigram where $count(u, v, w) = 0$, this will result in zero probabilities of any sequence containing u, v, w, which means many trigram probabilities are systematically underestimated (Peng and Schuurmans 2003). Therefore, smoothing methods are introduced for assigning non-zero probabilities to novel trigrams.

One simple and straightforward smoothing technique is the general additive smoothing, which is derived from Laplacian smoothing (Vatanen, Jaakko, and Virpioja 2010). Instead of adding 1 to the counts of all possible trigrams, additive smoothing uses a smaller value $\lambda \in (0, 1)$ for alleviating overestimation of the probabilities of unseen trigrams

$$p(x_i|x_{i-2}, x_{i-1}) = \frac{count(w_{i-2}, w_{i-1}, w_i) + \lambda}{count(w_{i-2}, w_{i-1}) + \lambda\ T} \tag{7}$$

where T is the size of vocabulary in the training corpus.

2.3 RNN Language Model

Recurrent neural network has a wide application in NLP (natural language processing) task, including automatic speech recognition (ASR), machine translation (MT) and optical character recognition (OCR) and so on (Raghunathan et al. 2010). Socher, Perelygin, and Wu (2013) succeed in using RNN to syntactic analysis. Then, a perceptron layer is added beyond the previous work, in order to improve expressing ability. Irsoy and Cardie Irsoy and Cardie (2014) expand RNN to 3 layer RNN, make it a deep network structure.

Mikolov, Yih, and Zweig (2013) succeed in improving RNN in language model task by speeding up training and testing phases, and result accuracy is also been improved. Using recurrent neural network to map input sequences to output sequences has a large research potential, for application in sequence recognition, production, or time series prediction.

The goal of statistical language model is to predict next word in textual data given context (Bengio, Simard, and Frasconi 1994). Although text prediction task can be handled with building RNN language model, there has been struggle in other methods, such as describe natural language as parser tree, and so on. Brown has made n-gram language model to do the text prediction task and able to extract word classes (Brown et al. 1992). Vatanen, Jaakko, and Virpioja (2010) has build a n-gram language model by combining n-gram set into unigram language model and get a good result in speech recognition.

In language model domain, there are still other approach attempts. People who focus on linguistic method believe word prediction task can be solved by constructing model using linguistic knowledge. However, this method needs huge amount of human work and expert background in linguistic. Additionally, the model is in low dimension. The statistic method also has drawbacks. Usually it is unexplainable what really happened inside the statistic model. Under the Markov assumption, the model often only depends on 2 preceding words to predict the next word.

RNN has the advantages of using previous word information and high dimension embeddings. RNN is different from FNN(feed-forward neural network), back propagation process is added into the neural network. In order to deal with time series sequence, RNN builds the relation between neural cells in same layer. So RNN can use previous information to do the task such as word prediction, name entity recognition, sentiment analysis and so on.

For current input x^t,the previous state $s^{(t-1)}$ is regarded as a parameter and pass into the current cell $s^{(t)}$.

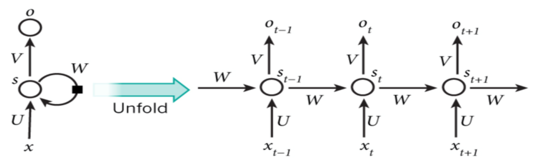


Figure 2: The Structure of RNN (wildml.com 2016)

$$e^{(t)} = x^{(t)}\boldsymbol{L} \tag{8}$$

$$s^{(t)} = sigmoids(s^{(t-1)}H + e^{(t)}I + b_1) \tag{9}$$

$$\widehat{y} = softmax(s^{(t)}U + b_2) \tag{10}$$

The structure of RNN is shown above in Figure 2. x^t is the input of current word. $\boldsymbol{L}$ is the embedding matrix. $s^{(t)}$ is the state of current cell in the hidden layer. $s^{(t-1)}$ is the previous information of the word sequence, RNN takes $s^{(t-1)}$ as a parameter. $\widehat{y}$ is a probability distribution for every word in the corpus to be the right answer.

3 Method

3.1 Random Number

It was necessary to create a method that randomly chooses an answer, in order to see if the other methods were producing significant results. When the sentences were read in by the model, the possible answers were stored in a list. A random number between 0 and 9 was generated, using the "radint" command was imported from the library called "random". The word that was in the randomly generated number's slot, in the list of possible answers, was then returned.

3.2 Most Frequent

When creating a complex search algorithms, it is common practise to create a simpler model as a baseline. This allows for comparisons of how well the code does but also of the run time the code, which may be as important than the accuracy in some situations. Every word in the 20 sentences was read into a dictionary, which counted the number of times it appeared. The possible answers were then looked up and their counts compared, then the word with the highest count was returned. If there were multiple words with the same highest count, the word that appeared first in the possible answers was returned. If a word didn't appear in sentences it was assumed to have a count of zero. If all the possible answers had a count of zero, "NA" was returned instead.

3.3 Trigram Language Model

To make the best use of Trigram Language Model, several approaches were applied on this task. Basic approaches include bigram model, trigram model, reversed bigram model and reversed trigram model. For further approaches, we introduced POS(part-of-speech) tagging techniques, which is based on the tagging results by Stanford CoreNLP tool (Manning et al. 2014). With this tagging technique, we can extend the Trigram Language Model with the additional attribute of POS and make interactive prediction between word and POS, and find out how the model perform with these additional attributes. In addition, some common but less useful words are suspected of having certain effects on the model while another potential influence is suspected from plural and suffix, thus stop words and word stemming pre-processes are attempted as well.

In the basic approach, candidate words are rated based on previous one word (bigram model), previous two words(trigram model), next one word (reversed bigram model) and next two words (reversed trigram model). For each candidate word, four values of p can be got from four models. Considering the rates as evaluation indicator, and then the highest rated word can be chosen from candidates as the most possible answer. For example, a query line in the CBT dataset would like:

“ Thank you , sir , ” said Chester firmly , “ but I must XXXXX at once if you ’ll kindly direct me .
go
gave|go|got|live|made|met|moved|returned|see|sit

In this example, ’XXXXX’ represents the missing word, ’go’ is the correct answer, and the following ten words are the candidate words. The language models calculate the rates for each candidate word. In this example, we get four surrounding words: ‘I’, ‘must’, ‘at’, ‘once’. With the idea described above, four rates can be got from the four sub-models:

$$q_1(\text{XXXXX}|\text{must}),\ q_2(\text{XXXXX}|\text{I, must})$$
$$\text{and for reversed models: } q'_3(\text{XXXXX}|\text{at}),\ q'_4(\text{XXXXX}|\text{once, at})$$

Sum the four rates for each of the candidate words, and the word having the highest rate is the choice made by the basic language model approach. This is a word-to-word approach as the candidate words are evaluated based on the surrounding words.

Another approach involves the POS tagging technique. Both training data and test data are preprocessed by Stanford CoreNLP, and the candidate words are rated with their POS in this approach:

Thank:VB you:PRP sir:NN said:VBD Chester:NNP firmly:RB but:CC
I:PRP must:MD XXXXX:NNP at:IN once:RB if:IN you:PRP 'll:MD
kindly:RB direct:VB me:PRP
go:VB
gave:VBD|go:VB|got:VBD|live:VB|made:VBD
|met:VB|moved:VBD|returned:VBD|see:VB|sit:VB

It is very similar to the basic approach, but POS of the missing word is the one to be predicted based on the POS of surrounding words, as a POS-to-POS approach. And the rates would like:

$$q_5(\text{POS(XXXXX)}|\text{MD}),\ q_6(\text{POS(XXXXX)}|\text{PRP, MD}),$$
$$q'_7(\text{POS(XXXXX)}|\text{IN}),\ q'_8(\text{POS(XXXXX)}|\text{RB, IN})$$

Furthermore, based on the previous two approaches, another advanced and mixed trigram model can be built, which evaluates word based on surrounding POS and evaluates POS based on surrounding words. Then additional eight rates can be got:

$$q_9(\text{XXXXX}|\text{MD}),\ q_{10}(\text{XXXXX}|\text{PRP, MD}),$$
$$q'_{11}(\text{XXXXX}|\text{IN}),\ q'_{12}(\text{XXXXX}|\text{RB, IN}),$$
$$\text{and, } q_{13}(\text{POS(XXXXX)}|\text{must}),\ q_{14}(\text{POS(XXXXX)}|\text{I, must}),$$
$$q'_{15}(\text{POS(XXXXX)}|\text{at}),\ q'_{16}(\text{POS(XXXXX)}|\text{once, at})$$

To build the evaluation data, two corpus are used for training. The four training datasets in CBT are taken as a larger corpus to have general idea of how is the language like in the field of children's story. Intuition is that contexts, the first 20 sentences, are important to the answers as well, and they are used as another training corpus. Thus, the total number of rates is doubled, and the overall rate is then be sum of all previous mentioned rates. The highest rated candidate is chosen as the answer.

$$S = \sum_{i=1}^{32} w_i q_i \tag{11}$$

3.4 RNN Language Model

The RNN language model was implemented on tensorflow library, and used cross entropy to calculate the loss. Then SGD(Stochastic Gradient Descent) algorithm was used to train the parameters for the RNN.

The CBT dataset had 5 million words in total, all of them were counted and transfer to one hot vector. Then they were encoded to word embeddings. The size of embedding was set to 50. These embeddings of word were the representatives of word, and also could be seen as the feature extracted by RNN. The activation function in the hidden layer was sigmoid function. All the word embeddings were pass into input-placeholder and label-placeholder ,the difference was that the label in the label-inputplaceholder was the next word for current input word in the input-placeholder, read in word by word until the end of words sequence. The training loop had been set to 10. After 10 iterations of training, the perplexity of the model decreased to 130 from 300. After training the model, we generated another same model, used the left part of given 21 st sentence as the given input dataset, restored the weights that had been trained and stored in the disk, and ran the new model without optimizer. The next word's probability distribution was available. Found the highest probability candidates in the list of candidates, the top one candidate was the prediction word.

Loop through all the test dataset for all classes of word, for every query, check whether the prediction was equal to the given right answer, finally the precision was available.

4 Results

4.1 Overall

With our approaches (Table 1), trigram language model provided highest accuracy for all the tests, 0.57 for common noun, 0.536 for named entity, 0.632 for verb and 0.638 for preposition. However, POS tagging did not help trigram language model to improve the results as expected, and RNN did not perform well for named entity, but relatively better for verbs.

Method	Named Entity	Common Noun	Verb	Preposition
Random	0.058	0.097	0.105	0.118
Most Frequent	0.024	0.258	0.289	0.289
RNN	0.040	0.190	0.390	0.330
Trigram (word to word)	0.536	0.570	0.632	0.638
Trigram (POS to POS)	0.125	0.146	0.137	0.129
Trigram (mixed)	0.382	0.492	0.555	0.584

Table 1: The results of all the methods used in this paper

4.2 Trigram Language Model

Approach	Training Set	NE	V	CN	P	Ave
Trigram Basic	context (20 sentences)	0.4408	0.3436	0.4076	0.2080	0.3500
Trigram Basic	training data	0.3296	0.6620	0.4504	0.6500	0.5230
Trigram Basic	training & context	0.5364	0.6324	0.5700	0.6376	0.5941
Trigram POS	context	0.1816	0.3620	0.3532	0.3388	0.3089
Trigram POS	training data	0.1416	0.4816	0.2812	0.4168	0.3303
Trigram Wmixed	context	0.1348	0.2996	0.3444	0.2104	0.2473
Trigram Wmixed	training data	0.1612	0.5668	0.4020	0.6160	0.4365
POS & Wmixed	training & context	0.3828	0.5552	0.4924	0.5840	0.5036

Table 2: The results of approaches based on trigram language model

The table 2 is the results of approaches based on trigram language model. The 'Trigram Basic' means an approach with trigram, bigram, reversed bigram, and reversed trigram models. The last row 'Ave' represents the average accuracy of the NE (name entity), V (verb), CN (common noun) and P (preposition).

From the table 2, the 'Trigram Basic' with training data and context has the highest accuracy (0.5941). And we calculated the scores with the context or training data individually, which are 0.35 and 0.523. It is suggested that combining the two training set could get a higher accuracy. Next, the 'Trigram POS"s values are relatively lower (0.3089 and 0.3303). After combing the POS with Wmixed, the effect becomes better (0.5036) but it does not exceed 'Trigram Basic'. This is because some POS's evaluation indicator q are not allocated the appropriate weight w.

Stop Words	Stemming	NE	V	CN	P	Ave
FALSE	FALSE	0.5364	0.6324	0.5700	0.6376	0.5941
FALSE	TRUE	0.3964	0.5688	0.4496	0.6268	0.5104
TRUE	FALSE	0.3900	0.2076	0.3924	0.0020	0.2480
TRUE	TRUE	0.2932	0.1640	0.3244	0.0056	0.1968

Table 3: The results of approaches with Stop words and Stemming

As the table 3 shows, we introduced the influence of stop words and stemming preprocesses. This table is based on the approach of 'Trigram Basic', and we also use the same method to other approaches. The results could be found in the appendix.

In this table, the first line is the accuracy based on the original data without any preprocessing. After introducing stemming, the values of accuracy decline overall, instead of rising as we predicted. Especially, the decrease of NE is fastest, from 0.5364 to 0.3964. Then the second one is CN's, from 0.5700 to 0.4496. Although the accuracies of V and P go down as well, their declines are not significant. Based on this situation, we infer that stemming could avoid the influence of plural and suffix, but it cannot distinguish name entity and some common nouns very well, which has a negative effort to result.

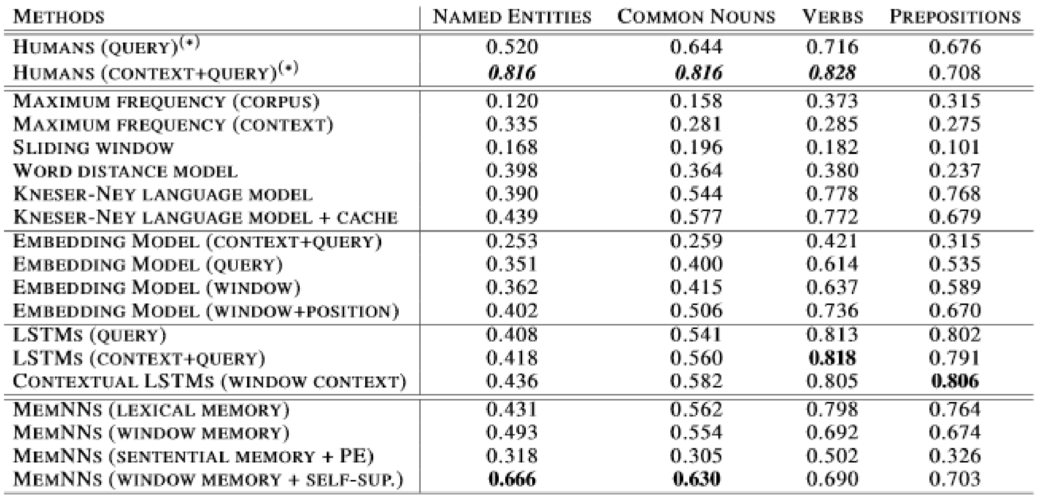
Then we explore that whether the stop words have certain effects on this language model and the third line of this table shows the result. It has more significant decline than stemming, especially to the accuracy of P, from 0.6376 to 0.002. This is because most of the prepositions are attributed to stop words automatically. In this case, the inference of language model become independent.

In general, although the result did not become better by stemming and stop words, that is not to say that these two preprocessing methods are not available. Many ways could improve them. For example, it is feasible to avoid to stem name entity and remove prepositions from stop words.

4.3 RNN Language Model

As shown in the figure 3, the results of RNN language model is not good as LSTM and memory network, especially the name entity of RNN language model, which is worse than the random pick up. It is well known that recurrent networks are difficult to train and therefore are unlikely to show the full potential of recurrent model (Sundermeyer, Schl, and Ney 2012). In order to improve the memory capacity of RNN, a information gate could be added into the hidden layer of the RNN.The information gate can decide what information is to remain and forgot. Another way to improve the results is to use bi-recurrent neural network, from both sides to the missing word, instead of just use the word sequence before the missing word as the input in the query to do the prediction task.

5 Analysis



METHODS	NAMED ENTITIES	COMMON NOUNS	VERBS	PREPOSITIONS
HUMANS (QUERY)$^{(*)}$	0.520	0.644	0.716	0.676
HUMANS (CONTEXT+QUERY)$^{(*)}$	***0.816***	***0.816***	***0.828***	0.708
MAXIMUM FREQUENCY (CORPUS)	0.120	0.158	0.373	0.315
MAXIMUM FREQUENCY (CONTEXT)	0.335	0.281	0.285	0.275
SLIDING WINDOW	0.168	0.196	0.182	0.101
WORD DISTANCE MODEL	0.398	0.364	0.380	0.237
KNESER-NEY LANGUAGE MODEL	0.390	0.544	0.778	0.768
KNESER-NEY LANGUAGE MODEL + CACHE	0.439	0.577	0.772	0.679
EMBEDDING MODEL (CONTEXT+QUERY)	0.253	0.259	0.421	0.315
EMBEDDING MODEL (QUERY)	0.351	0.400	0.614	0.535
EMBEDDING MODEL (WINDOW)	0.362	0.415	0.637	0.589
EMBEDDING MODEL (WINDOW+POSITION)	0.402	0.506	0.736	0.670
LSTMS (QUERY)	0.408	0.541	0.813	0.802
LSTMS (CONTEXT+QUERY)	0.418	0.560	**0.818**	0.791
CONTEXTUAL LSTMS (WINDOW CONTEXT)	0.436	0.582	0.805	**0.806**
MEMNNS (LEXICAL MEMORY)	0.431	0.562	0.798	0.764
MEMNNS (WINDOW MEMORY)	0.493	0.554	0.692	0.674
MEMNNS (SENTENTIAL MEMORY + PE)	0.318	0.305	0.502	0.326
MEMNNS (WINDOW MEMORY + SELF-SUP.)	**0.666**	**0.630**	0.690	0.703

Figure 3: The results obtained by other methods (Hill et al. 2016)

In comparing our results to those shown in Figure 3 from Hill et al. (2016), three methods performed better for CN–MemNNs (Window Memory + Self-Sup.), Contextual LSTMs (Window Context) and Kneser-Ney Language Model + Cache. The difference between our Trigram (word to word) model and their MemNNs model is 6%. For NE, only the MemNNs model did better by 13% than our Trigram (word to word) model. Concerning the V query set, eight of the models in Figure 3 outperformed our best model Trigram (word to word). The best model from Hill et al. (2016) outperformed our Trigram (word to word) by 18%. Finally, for P, nine of the other work's models had higher precision than our best model (17%).

For the limited amount of time assigned to evaluating the project's task, our Trigram (word to word) model still requires improvement in order to outperform the work from Hill et al. (2016). However, our results are fairly close in precision to their work considering this limited time, particularly in regards to NE.

6 Work Allocation

In the beginning of the project, work was allocated based on the time frame and consultation with the supervisor Andreas. In the beginning of the project, we reviewed and presented a paper each on previous or similar work for the project's task. Originally, we split the work according to an Agile sprint method where work was divided into the following sprints: Baseline Approaches by Matt and Li; Advanced Approaches by Mingjie, Peizhen and Yixuan; Quality Assessment and Revision by Jaspreet and Li; and Report Writing by Matt and Jaspreet. Due to time constraints and previous research knowledge, the allocation of sprints was reassigned. The Baseline Approaches was not as time-intensive as previously thought, so Matt was singularly assigned to this sprint. Advanced Approaches needed more resources, so Mingjie, Peizhen, Yixuan and Li. Quality Assessment and Revision was assigned to Jaspreet only. The Report Writing sprint was reassigned Jaspreet and Matt with contributions from the rest of the group, with regards to the methods they used.

7 Conclusion

In conclusion, the methods we developed and tested on the CBT corpora given the limited amount of time and resources show how choosing the correct candidate for the query is not as simple as initially thought. Even though human evaluation of the same queries is significantly larger in precision than the results we obtained, the ability of computer systems to solve linguistic problems is improving with new technologies and methods. We recommend using the Trigram (word to word) model as a basis for future work. However, it should be noted that our code required 50 hours of processing time to run all 10,000 queries in the CBT corpora. Also, 10 hours was required to build the index; therefore, we recommend ample time for evaluating the desired method.

Future work that would be pursued given more time would is an exploration of other methods. Coreferencing would be used to eliminate candidates and increase the probability of the computer system selecting the correct candidate. This method would be especially useful for NE and CN. The Memory Network method would require a generous amount of time and so would be for future work. A final step could be to implement a multi-sieve approach for generating a cluster output given all the methods used which would be especially beneficial to the task.

References

Bengio, Y, P. Simard, and P. Frasconi (1994). “Learning long-term dependencies with gradient descent is difficult”. In: *IEEE Transactions on Neural Networks* 5.2, pp. 157–166. ISSN: 10459227. DOI: 10.1109/72.279181. arXiv: arXiv:1211.5063v2. URL: http://ieeexplore.ieee.org/document/279181/.

Brown, Peter F. et al. (1992). “Class-based N-gram Models of Natural Language”. In: *Comput. Linguist.* 18.4, pp. 467–479. ISSN: 0891-2017. URL: http://dl.acm.org/citation.cfm?id=176313.176316.

Carroll, Lauren (2016). *Would Trump require schools to allow guns?* URL: http://www.politifact.com/truth-o-meter/statements/2016/nov/01/hillary-clinton/trump-said-he-would-require-schools-allow-guns-cli/ (visited on 12/12/2016).

Facebook Research (2016). *bAbI.* URL: https://research.fb.com/projects/babi/.

Hill, Felix et al. (2016). “The Goldilocks Principle: Reading Children’s Books with Explicit Memory Representations”. In: *Under review of ICLR*, pp. 1–13. arXiv: 1511.02301. URL: http://arxiv.org/abs/1511.02301.

Irsoy, Ozan and Claire Cardie (2014). “Deep recursive neural networks for compositionality in language”. In: *Advances in Neural Information Processing Systems* 3.January, pp. 2096–2104. ISSN: 10495258. URL: http://www.scopus.com/inward/record.url?eid=2-s2.0-84937828128%7B%5C&%7DpartnerID=tZOtx3y1.

Manning, Christopher D et al. (2014). “The Stanford CoreNLP Natural Language Processing Toolkit”. In: *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pp. 55–60. ISSN: 1098-6596. DOI: 10.3115/v1/P14-5010. arXiv: arXiv:1011.1669v3. URL: http://aclweb.org/anthology/P14-5010.

Mikolov, Tomas, Wen-tau Yih, and Geoffrey Zweig (2013). “Linguistic regularities in continuous space word representations”. In: *Proceedings of NAACL-HLT* June, pp. 746–751. URL: http://scholar.google.com/scholar?hl=en%7B%5C&%7DbtnG=Search%7B%5C&%7Dq=intitle:Linguistic+Regularities+in+Continuous+Space+Word+Representations%7B%5C#%7D0%7B%5C%%7D5Cnhttps://www.aclweb.org/anthology/N/N13/N13-1090.pdf.

Peng, Fuchun and Dale Schuurmans (2003). “Combining Naive Bayes and n-Gram Language Models for Text Classification”. In: *In 25th European*

Conference on Information Retrieval Research (ECIR), pp. 335–350. ISSN: 0302-9743. DOI: 10.1007/3-540-36618-0_24. URL: http://link.springer.com/10.1007/3-540-36618-0%7B%5C_%7D24.

Raghunathan, Karthik et al. (2010). "A Multi-pass Sieve for Coreference Resolution". In: *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*. EMNLP '10. Cambridge, Massachusetts: Association for Computational Linguistics, pp. 492–501. URL: http://dl.acm.org/citation.cfm?id=1870658.1870706.

Socher, Richard, Alex Perelygin, and Jy Wu (2013). "Recursive deep models for semantic compositionality over a sentiment treebank". In: *Proceedings of the . . .* Pp. 1631–1642. ISSN: 1932-6203. DOI: 10.1371/journal.pone.0073791. arXiv: 1512.03385. URL: http://nlp.stanford.edu/%7B~%7Dsocherr/EMNLP2013%7B%5C_%7DRNTN.pdf%7B%5C%%7D5Cnhttp://www.aclweb.org/anthology/D13-1170%7B%5C%%7D5Cnhttp://aclweb.org/supplementals/D/D13/D13-1170.Attachment.pdf%7B%5C%%7D5Cnhttp://oldsite.aclweb.org/anthology-new/D/D13/D13-1170.pdf.

Sundermeyer, Martin, Ralf Schl, and Hermann Ney (2012). "LSTM Neural Networks for Language Modeling". In: *Proc. Interspeech*.

Vatanen, Tommi, J V Jaakko, and Sami Virpioja (2010). "Language Identification of Short Text Segments with N-gram Models". In: *Lrec 2010*, pp. 3423–3430.

wildml.com (2016). *RNN*. URL: http://www.wildml.com/2015/09/recurrent-neural-networks-tutorial-part-1-introduction-to-rnns/ (visited on 12/12/2016).

Zipf, George Kingsley (1949). *Human behavior and the principle of least effort : an introduction to human ecology*. 1st ed. Addison-Wesley Press.