1. Introduction

Natural Language Processing (NLP), as one of the artificial intelligence topic, is becoming more and more popular recently. It can also be divided into several parts like text to speech, speech recognition, information retrieval, etc. Our project focuses on getting the information from a passage of text. Our project is aimed to training machine to Read and Comprehend. When an article is given, the machine could automatically answer the questions. This project is one of the topic in Nature language processing, enabling computers to derive meaning from human or natural language input.

Dataset: In our project, we will be working with question answering using the Machine Comprehension Test (MCTest) dataset created by Microsoft [1]. For the story generate, they ensured that the data is the same quality as another set and also limited the text to children’s stories, so that it could be easily understand. Limited vocabulary: The lowercase words in the story, questions, and answers were stemmed from list of approximately 8000 words that a 7 year old is likely to know. Multiple-sentence questions: At least two of the questions need multiple sentences to answer. As each story is fictional, the answers can only be found in the story itself, which requires high-level machine comprehension without any world knowledge.

2. Review of literature

2.1 Traditional baseline methods:

Majority baseline (maximum frequency) picks the entity most frequently observed in the context document.

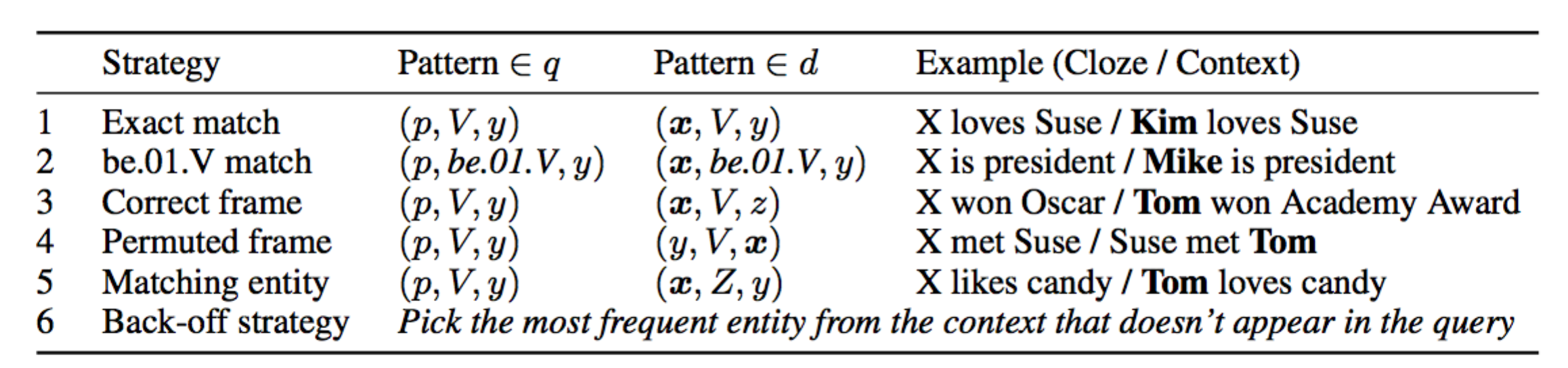
Exclusive majority (exclusive frequency) chooses the entity most frequently observed in the context but not observed in the query. The idea behind this exclusion is that the placeholder is unlikely to be mentioned twice in a single Cloze form query[2].

Word Distance Benchmark: The main idea is that choose the shortest distance among all of the answers to the question’s key words.

2.2 Traditional methods:

2.2.1 Frame-Semantic Parsing:

Frame-semantic parsing attempts to identify predicates and their arguments, allowing models access to information about “who did what to whom”. And we could see some of the language strategy patterns as below.



But this method has some limitations for can not possess the capability to generalize through a language model beyond exploiting one during the parsing phase[2]. And what is more, teaching machines to read natural language documents remains an elusive challenge. Machine reading systems can be tested on their ability to answer questions posed on the contents of documents that they have seen, but until now large scale training and test datasets have been missing for this type of evaluation. In other words, those are limited on training dataset.

2.2.2 Comprehension with Syntax, Frames, and Semantics:

Semantic: frame semantic parsing (Das et al., 2014) is aimed to extract frame-specific predicate-argument structures from sentences. [3]

Syntax: given each candidate answer, we attempt to transform the question to a statement using the  extraction rules[4].

But according to the error analysis suggests that deeper linguistic analysis and inferential reasoning can yield further improvements on this task.

2.2.3 Comprehension with Discourse Relations[5].

The model jointly identifies relevant sentences, establishes relations between them and predicts an answer. The key idea is to implant discourse analysis into a joint model for comprehension. And the drawback of this model is that the accuracy varies significantly according to the question type. Which means not really stable.

2.2.4 Comprehension for Learning Answer-Entailing Structures[6]

We posit that there is a hidden (latent) structure that explains the relation between the question, correct answer, and text. For this given structure, the correctness of the answer is evident. Since the structure is latent, it must be inferred. Firstly, we should presents a unified max-margin framework that learns to find these hidden structures (given a corpus of question-answer pairs), and then uses what it learns to answer machine comprehension questions on novel texts. But the drawback of this method is that since inference is cheaply modelled via alignment structure, we lack the ability to deeply reason about facts or numbers.

2.3 Modern method for neural network:

2.3.1 The Deep LSTM Reader embed long sequences into a vector representation which contains enough information to generate a full translation in another language. The result is that this model processes each document query pair as a single long sequence. Given the embedded document and query the network predicts which token in the document answers the query. So this one is the old version of neural networking

2.3.2 The Impatient Reader focuses on the passages of a context document that are most likely to inform the answer to the query. We can go further by equipping the model with the ability to reread from the document as each query token is read. The result is an attention mechanism that allows the model to recurrently accumulate information from the document as it sees each query token, ultimately outputting a final joint document query representation for the answer prediction.

2.3.3 The Attentive Reader can be viewed as a generalization of the application of Memory Networks to question answering. That model employs an attention mechanism at the sentence level where each sentence is represented by a bag of embeddings. This model could fix the bottleneck for the width of different words. This attention model first encodes the document and the query using separate bidirectional single layer LSTMs. This task has been tricky because of the number of parameters that must be fine-tuned.

Although this method cannot be backwards checking as impatient reader but according to some article, these two methods have the similar results, so our group used this model.

3. Description of the project task and the used methods

For data preprocessing, we use python NLTK library.

For deep learning part, we use blocks and fuels, Theano framework for deep learning part.

4. Implementation details and experimental setup

For preprocessing part, we firstly setup the NLTK working environment with python and later import some libraries like stem, tokenize, corpus, etc. Then we do the following steps for text preprocessing.

1) split the story dataset into one story one file, and do the same thing on the related answer dataset

2) label correct answers obtained from corresponding answer file

in each story file

3) convert ‘n’t’ to ‘not’ using Regular Expression and numbers to words using a recrusive function implemented by ourselves on story, questions and answers in each story file

4) lowercase all text of stories, tokenize them, then build a vocabulary file without repeat words

5) find out entities in each story and replace them with entity identifiers, then add the corresponding entity and entity identifier list below the questions

6) delete punctuations on the text of each story, the reason for doing this after step 5 is that punctuations will help to identify the entities

7) remove stop words, lemmatization and stemming on each story dealing with NLTK library functions

8) preprocess both questions and answers using the entity list gained from step 5

5. Results, performance evaluation, and analysis

6. Discussion of results and conclusions

7. References

[1] Matthew Richardson, Christopher J.C. Burges, Erin Renshaw. MCTest: A Challenge Dataset for the Open-Domain Machine Comprehension of Text.

[2] Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman and Phil Blunsom. Teaching Machines to Read and Comprehend.

[3] Hai Wang, Mohit Bansal, Kevin Gimpel, David McAllester. Machine Comprehension with Syntax, Frames, and Semantics.

[4] Moschitti et al., 2007; Wang et al., 2007; Heilman and Smith, 2010

[5] Machine Comprehension with Discourse Relations

[6] Mrinmaya Sachan, Avinava Dubey, Eric P. Xing, Matthew Richardson. Learning Answer-Entailing Structures for Machine Comprehension.