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]. How to get dataset: The authors of stories are all workers, residing in the United States. The average worker is 36 years old, more educated than the United States population in general (Paolacci et al., 2010), and the majority of workers are female. Workers were instructed to write a short (150-300 words) fictional story, and to write as if for a child in grade school. Workers were also asked to provide four reading comprehension questions pertaining to their story and, for each, four multiple-choice answers. Workers were requested to provide “reasonable” incorrect answers that at least include words from the story so that their solution is not trivial. Finally, workers were asked to design their questions and answers such that at least two of the four questions required multiple sentences from the story to answer them. That is, for those questions it should not be possible to find the answer in any individual sentence.

The MCTest corpus contains two sets of stories, named MC160 and MC500, and containing 160 and 500 stories respectively. MC160 was gathered first, then some improvements were made before gathering MC500. Adding a Grammar Test to form the MC500.

In addition to the details described above, MC160 workers were given a target elementary grade school level (1-4) and a sample story matching that grade level5.

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.

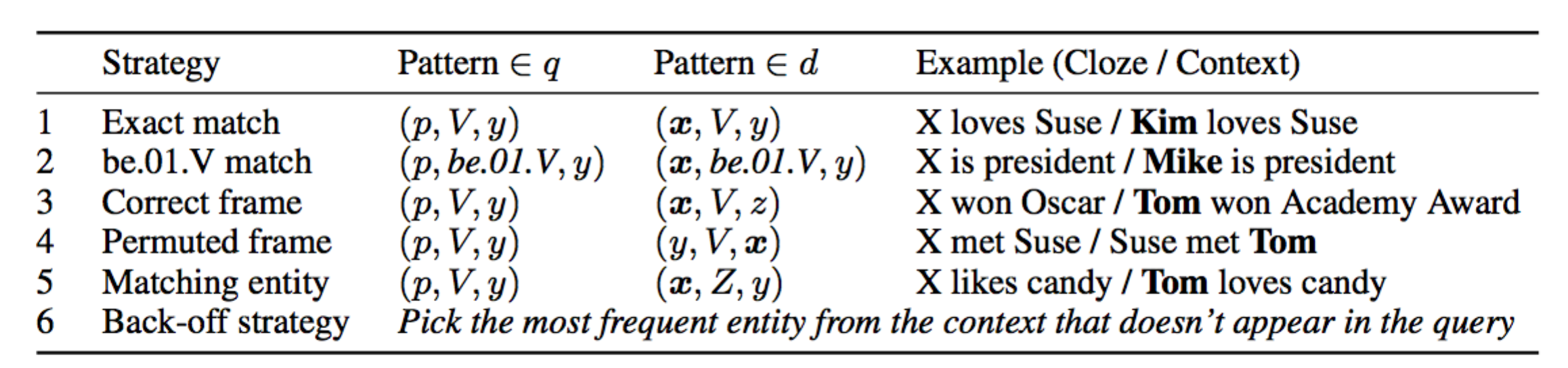
Exclusive majority (exclusive frequency) chooses the entity most frequently observed in the context but not observed in the query. The idea behind this exclusion is that the placeholder is unlikely to be mentioned twice in a single Cloze form query[2].

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:

talk about RNN and neural network, LSTM is a special one, according to the google paper deep mind paper, LSTM advantage.

In [machine learning](https://en.wikipedia.org/wiki/Machine_learning) and [cognitive science](https://en.wikipedia.org/wiki/Cognitive_science), artificial neural networks (ANNs) are a family of models inspired by [biological neural networks](https://en.wikipedia.org/wiki/Biological_neural_network) (the [central nervous systems](https://en.wikipedia.org/wiki/Central_nervous_system) of animals, in particular the [brain](https://en.wikipedia.org/wiki/Brain)) and are used to estimate or [approximate](https://en.wikipedia.org/wiki/Universal_approximation_theorem) [functions](https://en.wikipedia.org/wiki/Function_(mathematics)) that can depend on a large number of [inputs](https://en.wikipedia.org/wiki/Argument_of_a_function) and are generally unknown. Artificial neural networks are generally presented as systems of interconnected "[neurons](https://en.wikipedia.org/wiki/Artificial_neuron)" which exchange messages between each other.

An ANN is typically defined by three types of parameters:

* The interconnection pattern between the different layers of neurons
* The learning process for updating the weights of the interconnections
* The activation function that converts a neuron's weighted input to its output activation.

Contrary to feedforward networks, [recurrent neural networks](https://en.wikipedia.org/wiki/Recurrent_neural_network) (RNNs) are models with bi-directional data flow. While a feedforward network propagates data linearly from input to output, RNNs also propagate data from later processing stages to earlier stages. RNNs can be used as general sequence processors.

For RNN model, there are several types, like Fully recurrent network, Hopfield network, Elman networks and Jordan networks, Echo state network and Long short term memory network,etc.

The [Long short term memory](https://en.wikipedia.org/wiki/Long_short_term_memory) (LSTM) network, developed by [Hochreiter](https://en.wikipedia.org/wiki/Sepp_Hochreiter) & [Schmidhuber](https://en.wikipedia.org/wiki/J%C3%BCrgen_Schmidhuber) in 1997,[7] is an artificial neural network structure that unlike traditional RNNs doesn't have the [vanishing gradient problem](https://en.wikipedia.org/wiki/Vanishing_gradient_problem) (compare the section on training algorithms below). It works even when there are long delays, and it can handle signals that have a mix of low and high frequency components. LSTM RNN outperformed other methods in numerous applications such as language learning[8] and connected handwriting recognition.[9]

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.

Paper reading

Blocks and fuel[10]:

Two Python frameworks to train neural networks on large datasets: Blocks and Fuel. Blocks is based on Theano, a linear algebra compiler with CUDA-support (Bastien et al., 2012; Bergstra et al., 2010). It facilitates the training of complex neural network models by providing parametrized Theano operations, attaching metadata to Theano’s symbolic com- putational graph, and providing an extensive set of utilities to assist training the networks, e.g. training algorithms, logging, monitoring, visualization, and serialization. Fuel provides a standard format for machine learning datasets. It allows the user to easily iterate over large datasets, performing many types of pre-processing on the fly. [10]

Blocks and Fuel are being developed by the Montreal Institute of Learning Algorithms (MILA) at the University of Montreal. Their focus lies on quick prototyping of complex neural network models.

Instead of introducing new abstract objects representing ‘models’ or ‘layers’, Blocks annotates the Theano computational graph, maintaining the flexibility of Theano while making large models manageable.

Data processing is an integral part of training neural networks, which is not addressed by many of the aforementioned frameworks. Fuel aims to fill this gap. It provides tools to download datasets and iterate/preprocess them efficiently.

Theano is a popular choice for the implementation of neural networks (see e.g. Goodfellow et al. (2013b); Pascanu et al. (2013)). Blocks and many other libraries, such as Pylearn2 (Goodfellow et al., 2013a), build on Theano by providing reusable components that are common in neural networks, such as linear transformations followed by non-linear activations, or more com- plicated components such as LSTM units. In Blocks these components are referred to as bricks or “parametrized Theano operations”.

Blocks comes with a large number of ‘bricks’.

The gradient descent training algorithm in Blocks is composed of different ‘step rules’ that modify the descent direction (learning rate scaling, momentum, gradient clipping, weight norm clipping, etc.). A variety of algorithms such as AdaGrad, ADADELTA, Adam, RMSProp are available as step rules. But in our project, we use gradient descent algorithm.

Fuel’s goal is to provide a common interface to a variety of data formats and published datasets such as MNIST, CIFAR-10, ImageNet, etc. while making it easy for users to write an interface to new datasets.

Blocks relies on Fuel for its data interface, but Fuel can easily be used by other machine learning frameworks that interface with datasets.

Fuel offers built-in scripts that automate the task of downloading datasets, (similar to e.g. skdata1) and converting them to Fuel’s HDF5 specification.

The fuel download script is used to download raw data files. Downloading the raw MNIST data files is as easy as typing fuel download mnist. The fuel convert script is used to convert raw data files into HDF5-format.

Reproducibility being an important feature of both Fuel and Blocks, the fuel convert script automatically tags all files it creates with relevant module and interface versions and the exact command that was used to generate these files. Inspection of this metadata is done with the fuel info script.

Theano is a compiler for mathematical expressions in Python that combines the convenience of NumPy’s syntax with the speed of optimized native machine language. The user composes mathematical expressions in a high-level description that mimics NumPy’s syntax and semantics, while being statically typed and functional (as opposed to imperative). These expressions allow Theano to provide symbolic differentiation. [11]

Theano was introduced to the machine learning community by Bergstra et al. (2010) as a CPU and GPU mathematical compiler, demonstrating how it can be used to symbolically define mathematical functions, automatically derive gradient expressions, and compile these expressions into executable functions that outperform implementations using other existing tools.[12]

How deep attentive structures as mentioned in google paper, use the parameters, and different layers, how to calculate layer (compare with the paper and us), how to calculate the right answer in each layer, slide for calculate for a new paragraph.

Here is compare of original method and the method we used

The Attentive Reader can be viewed as a generalisation of the application of Memory Networks to question answering [3]. That model employs an attention mechanism at the sentence level where each sentence is represented by a bag of embeddings. The Attentive Reader employs a finer grained token level attention mechanism where the tokens are embedded given their entire future and past context in the input document.

The representation r of the document d is formed by a weighted sum of these output vectors. These weights are interpreted as the degree to which the network attends to a particular token in the document. Given this attention score the embedding of the document r is computed as the weighted sum of the token embeddings.

Below is what the original paper said.

For the Deep LSTM Reader, we consider hidden layer sizes [64, 128, 256], depths [1, 2, 4], initial learning rates [1E3, 5E4, 1E4, 5E5], batch sizes [16, 32] and dropout [0.0, 0.1, 0.2]. We evaluate two types of feeds. In the *cqa* setup we feed first the context document and subsequently the question into the encoder, while the *qca* model starts by feeding in the question followed by the context document. We report results on the best model (underlined hyperparameters, *qca* setup). For the attention models we consider hidden layer sizes [64, 128, 256], single layer, initial learning rates [1E4, 5E5, 2.5E5, 1E5], batch sizes [8, 16, 32] and dropout [0, 0.1, 0.2, 0.5]. For all models we used asynchronous RmsProp [20] with a momentum of 0.9 and a decay of 0.95.

What we did is using a new method: (here should insert the function in PPT)

Define a new method to determine which possible answer is the correct one. (function) That is, the correct answer should be the most similar to the most highly attended words in the story. We use the similarity metric (function) for a multiple choice answer of length n words. Take the weighted average of the attention each word from the multiple choice answer has within the story. If a word in the answer is not in the story, we then chooce to use word distance from the closest attended word as apenalty.

New graph for the run time, google highlighten entity method, bofore our method.

As for the run time, and computational complexity. **Original Attentive Reader:** trained for 2-4 days using two NVIDIA Titan Black GPUs (PassMark score 8,667 each)

**Our setup:** one NVIDIA GTX 660 GPU (PassMark score 4,120) inside older machine: Core 2 Quad Q8300, 8GB DDR2 PC6400 RAM. And the training time is around one day.

For result, we first calculate weight matrix multiply vector embedding matrix for each word, and choose the highest possibility. The higher the possibility it is, the more colorful the words could be.

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. We deal with deleting punctuation like “,” “.”,etc. Later we also extract questions pattern like “when”, “how”, etc. we change numbers into English words by dictionary. We also do the following step for text preprocessing.

1) using stop words library to remove stop words

2) using regular expression

3) def a function and by recursion to convert a number into a verb.

4) using library to deal with stem words

5) using lemmatizatio: Returns the input word unchanged if it cannot be found in WordNet.

6) dealing with rawfile and answer file input. To become like…

7) getting the entities.

8) preprocessing both questions and answers

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.

[7] [Hochreiter, Sepp](https://en.wikipedia.org/wiki/Sepp_Hochreiter); and [Schmidhuber, Jürgen](https://en.wikipedia.org/wiki/J%C3%BCrgen_Schmidhuber); Long Short-Term Memory, Neural Computation, 9(8):1735–1780, 1997

[8] Gers, Felix A.; and Schmidhuber, Jürgen; LSTM Recurrent Networks Learn Simple Context Free and Context Sensitive Languages, IEEE Transactions on Neural Networks, 12(6):1333–1340, 2001

[9] Graves, Alex; and Schmidhuber, Jürgen; Offline Handwriting Recognition with Multidimensional Recurrent Neural Networks, in Bengio, Yoshua; Schuurmans, Dale; Lafferty, John; Williams, Chris K. I.; and Culotta, Aron (eds.), Advances in Neural Information Processing Systems 22 (NIPS'22), December 7th–10th, 2009, Vancouver, BC, Neural Information Processing Systems (NIPS) Foundation, 2009, pp. 545–552

[10] Blocks and Fuel: Frameworks for deep learning

[11] Theano: A CPU and GPU Math Compiler in Python

James Bergstra, Olivier Breuleux, Frédéric Bastien, Pascal Lamblin, Razvan Pascanu, Guillaume Desjardins, Joseph Turian, David Warde-Farley, Yoshua Bengio

[12] Theano: new features and speed improvements

Fre ́de ́ric Bastien, nouiz@nouiz.org Pascal Lamblin, lamblinp@iro.umontreal.ca Razvan Pascanu, r.pascanu@gmail.com James Bergstra, james.bergstra@gmail.com Ian Goodfellow, goodfeli@iro.umontreal.ca Arnaud Bergeron, bergearn@iro.umontreal.ca Nicolas Bouchard, nicolas.bouchard.1@gmail.com David Warde-Farley, wardefar@iro.umontreal.ca Yoshua Bengio, yoshua.bengio@umontreal.ca Dept. IRO, Universite ́ de Montre ́al, Montre ́al (QC), H3C 3J7, Canada