CSE 575 Statistical Machine Learning

Commonsense Reasoning, Coreference resolution (WinoGrad Schema Challenge)

Group 2 - Team Members

1. **Prashant Garg** - 1207978391 - pgarg10@asu.edu [18%]

Project Management, Project Idea, Regex engine , Finding various resource and datasets, java Development of regex engine.

2. **Sagar Patni** - 1213217718 - shpatni@asu.edu [17%]

Initial reading of papers and future model proposal, LSTM implementation, converting Input into regex using nltk.

3. **Shiksha Patel** -1211045638 - spatel55@asu.edu [17%]

Initial reading of papers and future model proposal,coreference resolution ,LSTM implementation, filter questions from statements program using Stanford CoreNLP

4. **Vishal** - 1211138809 - vvaidhya@asu.edu [17%]

Filter questions from statements using Stanford CoreNLP, Use Neural network parser to get sentence structure for above filtering, Coreference resolution

5. **Aravind Manoharan** -1213332547 - amanoh10@asu.edu [16%]

coreference resolution , Database management,LSTM

6. **Kishore** - 1212457608, kmadhav1@asu.edu [15%]

Initial reading of papers and future model proposal,Coreference resolution,Database management.

INDEX

Introduction:

Natural Language Processing began in 1950 with the emergence of Turing Test. A computer is considered intelligent according to turing test if it could carry on a conversation with a human without the human realizing that the other person is a machine. NLP strives towards this goal of achieving human computer interaction and it will play a vital role in bridging the gap between human communication and digital data. NLP shall heavily influence research areas like machine translation, fighting spam, information extraction, summarization and question answering. In this project we endeavor to take on the challenge of question answering.

Problem Statement:

Word2Vec is a famous model to find patterns and relations between words. But there is no technique in which semantic meaning relationship of sentences can be used to differentiate between the classification of whether a statement is a question or not followed by whether this statement is a valid statement in accordance with the knowledge we are looking for. Last step being how similar are two different types of sentences in extracting same type of knowledge.

Challenge is:

1. Sam is not able to lift his son because he is so heavy.

2. Sam is not able to lift his son because he is so weak.

Who is he in both sentences ?

Challenge Overview:

Winograd Schema questions require the resolution of anaphora, that is the machine must identify the antecedent of an ambiguous pronoun in a statement. Example:

*1.The trophy wouldn't fit in the suitcase because **it** was too big(or small).*

2. What was too big(or small)

Answer?

1. Trophy
2. Suitcase

We are not solving the whole winograd challenge [link] here but trying to resolve the part of it and one of the main infrastructure in which two similar sentences can be used to extract knowledge

Current Research:

From Last few years a lot of people are trying to resolve the winograd challenge schema. Every year, in famous Natural language Processing conferences like ACL, EMNLP. Even a lot of work is going in Arizona State university labs regarding the same. The project idea is inspired by the work of Arpit Sharma, a phd candidate at Arizona state university[8].

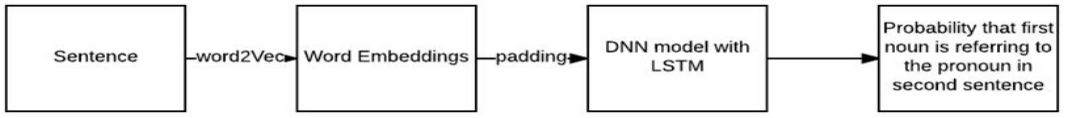
Here we present an approach designed by Quin Liu[9] et al for solving the Winograd schema problems.This approach is to learn knowledge enhanced embeddings based on the common sense knowledge used to generate constraints.

Based on the generated constraints, the following two methods are proposed 1. Unsupervised Semantic Similarity Method. 2. Neural Knowledge Activated Method. The first method is an unsupervised method and uses all the candidate mentions from the pre-trained embeddings and the second method is a supervised method which differs from the first .The former uses uses deep neural networks having composed embedding vectors as input features. The overall accuracy comes 50 percent on the corpus "Wiki-Large".

Our Approach

We have considered 2 different approaches for our system.

1. Logic programing using Regex.
    a.Convert the sentence into regex.
    b.Check for the answer in KB.
    c.If not in Kb, then search for similar regex in various data sources(in our case, search engines).
    d. Then extract the data from these resources filter it out.
    e.Use Machine Learning to differentiate, the data into sentence and question.
    f. Use sentences to extract knowledge related to the asked Question.
2. Neural Network + NLP



In approach 2, we examine deep learning approaches to the problem

Language Processing:

A. NLTK

NLTK is a leading platform to work with human language data. It can be used as an interface to many resources such as tokenization, tagging words and parsing the data.

NLTK Tokenizer:

Tokenizer converts a strings into a string of substrings. For example, tokenizer converts a sentence into a series of words and punctuations.

For example, given a sentence

"*Sam can not lift his son because he is heavy*."

When we tokenize the above sentence, it converts into a series of words and punctuations like below.

['Sam', 'can', 'not', 'lift', 'his', 'son', 'because', 'he', 'is', 'heavy', '.']

NLTK Tagger:

NLTK Tagger processes the series of words which is given by the tokenizer and attaches a part of speech tag to each word.

For example, given a tokenized words

['And', 'now', 'for', 'something', 'completely', 'different']

When we tag these words, we get the following results

[('And','CC'), ('now','RB'), ('for','IN'), ('something', 'NN'), ('completely','RB'), ('different','JJ')]

B. Stanford CoreNLP

Stanford CoreNLP is another leading platform for working with human language data. Similar to NLTK, it can also give base form of words like tokenizing, their parts of speech, normalizing dates and time, indicate sentiment and differentiating questions from statements.

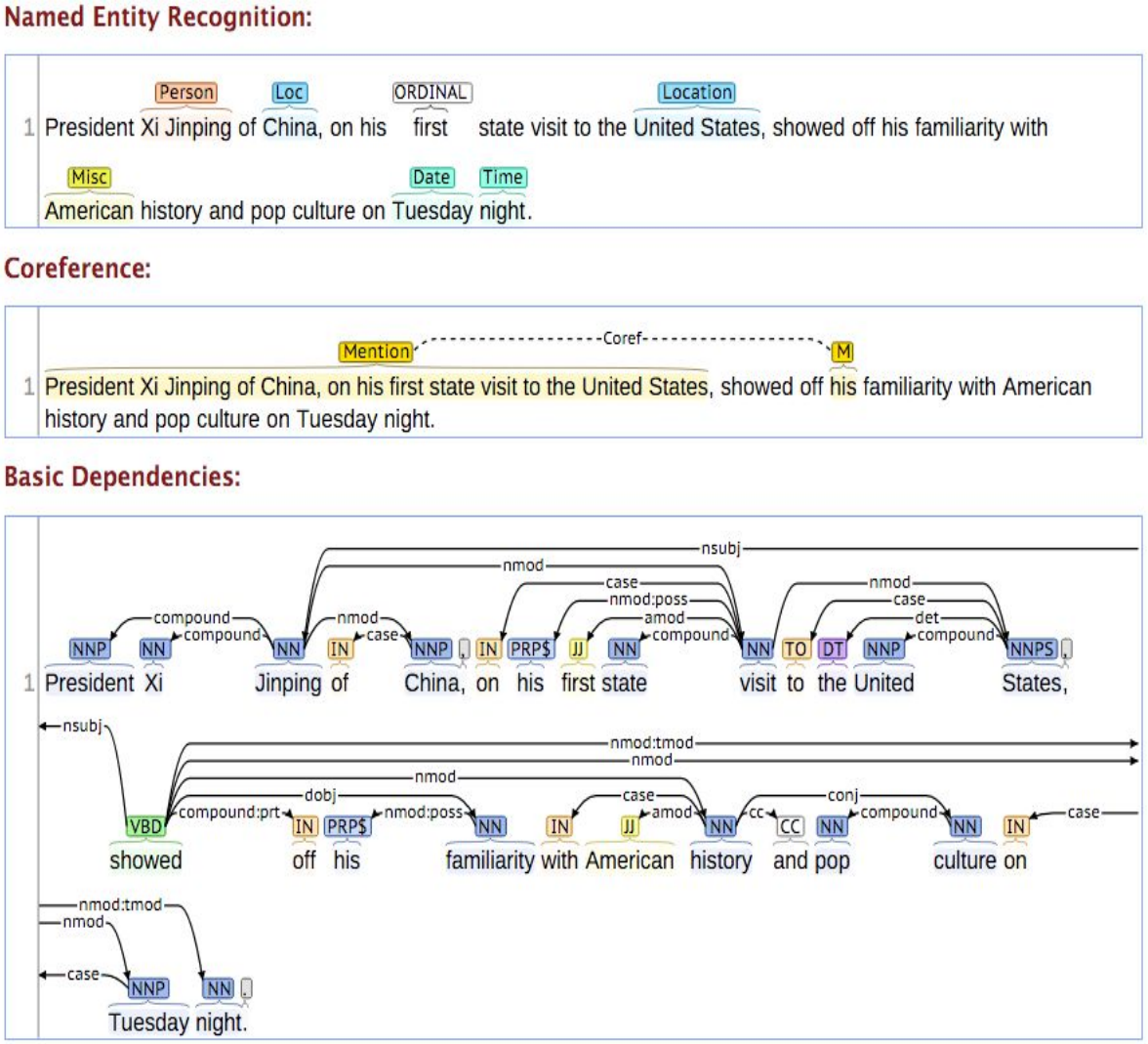


Fig. 1 Constituency parsing using Stanford CoreNLP
(https://stanfordnlp.github.io/CoreNLP/)

Word2Vec:

Word2Vec procedure is used for converting words in vectors of n dimensions and it retains relationship among the context word

There are two approaches for converting words to vectors
1. CBOW
2. skip-gram

The CBOW approach tries to predict the word given the context words and skip-gram approach tries to predict the context words given a word.

In this project we are using skip gram approach, it works as illustrated below

1. Define the vocabulary size
2. Define the vector length.
3. Feed forward neural network with input size and output size as length of vocabulary and hidden layer size as vector length.
4. Define a window size to capture context words.
5. Generate word pairs as (word, context words) by scanning through the document using the defined window size.
6. Context the vocabulary into continuous bag of words.
7. Train the Neural network to predict the context word given a word.
8. After training discard the output layer and weight of the hidden layer represents the word vectors.

Diagram:

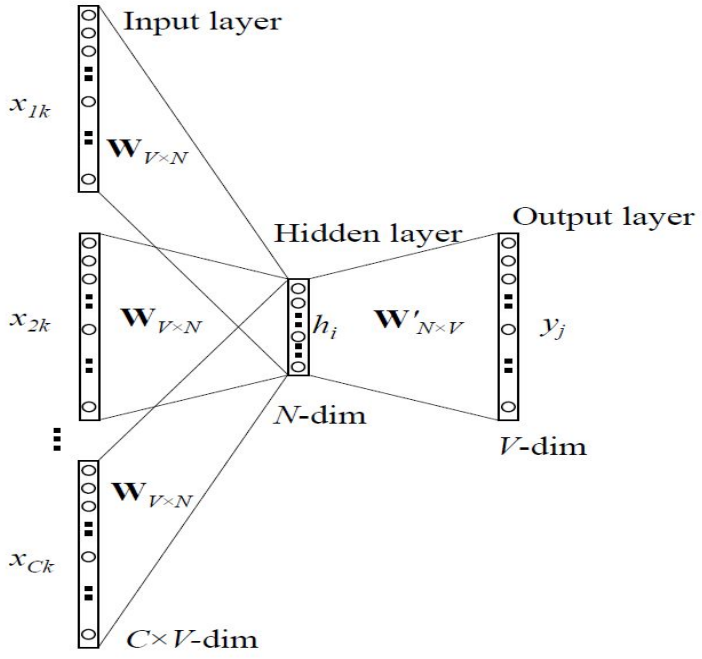


Fig. 2 Neural Network structure (Source: https://i.stack.imgur.com/fYxO9.png)

Neural Network

A. Feed Forward Neural Network.

These are simplest form of neural network which makes prediction by combining information information from previous layers. The idea behind the learning is to understand If we perturb the input by small amount how the output is changed. The Feed forward neural Network learns by propagating the error from output layer and adjusting the weights and biases at each layer.

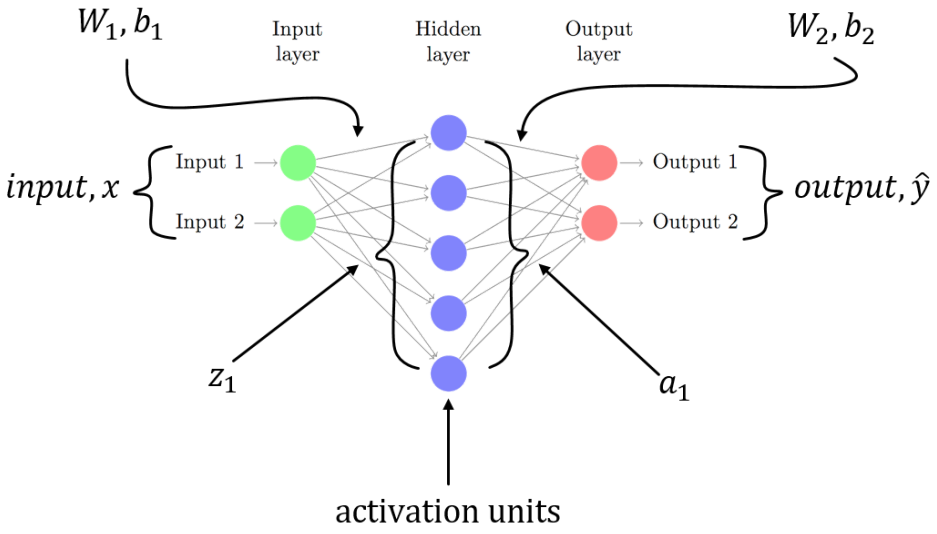


Fig. 3 Feedforward Architecture (Source: https://i.stack.imgur.com/iHDtO.png)

The algorithm performs below taks

1. Forward pass
2. Calculate the error in the prediction.
3. Backpropagation

In step 1, The data and the weights of the network are used to compute a prediction (y).

In step 2, the error is computed based the true labels provided by user and prediction computed in step 1.

In step 3, the weights in each layer are adjusted based on error computed in step 2. The derivative of sigmoid at output layer is calculated to determine the value of slope and that is multiplied by the error. This give the magnitude of the error and the direction in which the weights needs to be changed.

B. Recurrent Neural Network

The idea behind RNN is to make use of the information sequentially. The RNN takes sequence of data as an input and tries to predict the future steps. Along with the current input data RNN takes input from previous hidden layers.

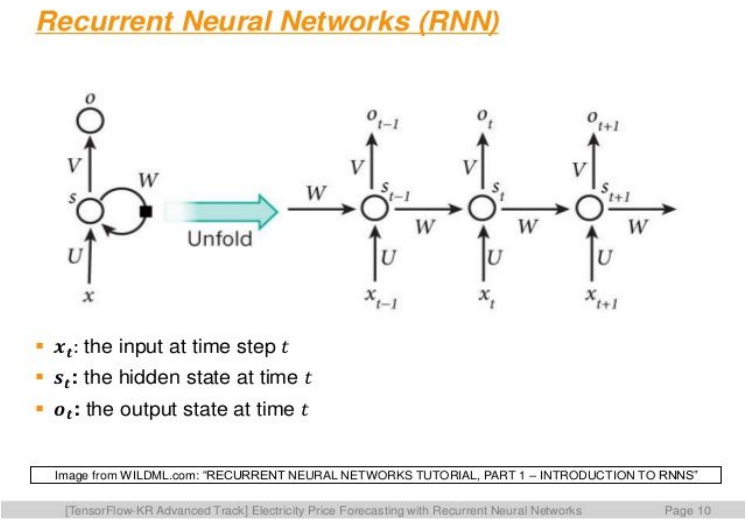


Fig. 5 Recurrent Neural Networks Structure
(Source:https://image.slidesharecdn.com/electricitypriceforecasting-160617195753/95/electricity-price-forecasting-with-recurrent-neural-networks-10-638.jpg?cb=1466221932)

C. LSTM

The LSTM addresses the vanishing gradient problem seen in vanilla recurrent neural networks. The LSTM is controlled by 4 gates viz 1. Selection 2. Ignore 3. Foregt 4. Memory.

Each gate is a neural network which learns the functionality with combination of inputs and neural network training. Memory unit remembers many steps, Selection gate learn what prediction to select, Forget gate learns what information to forget and Ignore gate learns to ignore data from previous layers.

LSTM diag.

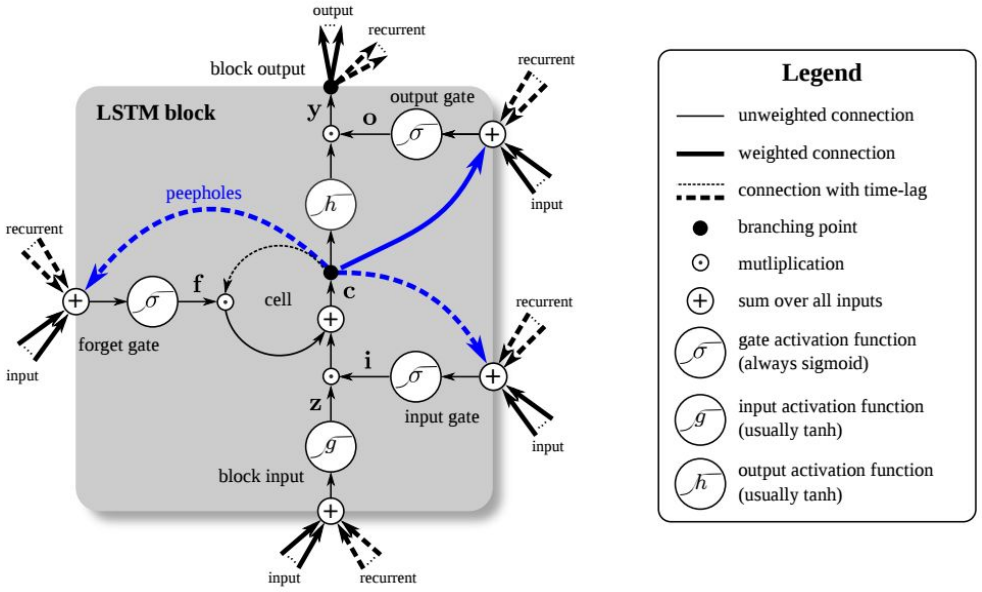


Fig. 6 LSTM Structure
(Source: https://devblogs.nvidia.com/parallelforall/wp-content/uploads/2016/03/LSTM.png)

D. Recurrent Neural Network learning BPTT

The learning in RNN is similar to that of simple feed forward neural network. In RNN, the hidden layer of previous layer is given as a input to next layer so we have to consider to learn weights from previous layers as well i.e. we have to propagate error in layer n back to layer 0. This approach is called as BackPropagation Through Time. The Recurrent Neural Network has difficulties in remembering long term dependencies and it suffers from vanishing gradient problem of neural network learning. Which is addressed by Long Short Term Memory Units illustrated above.

Our Approaches to the problem

Flowchart

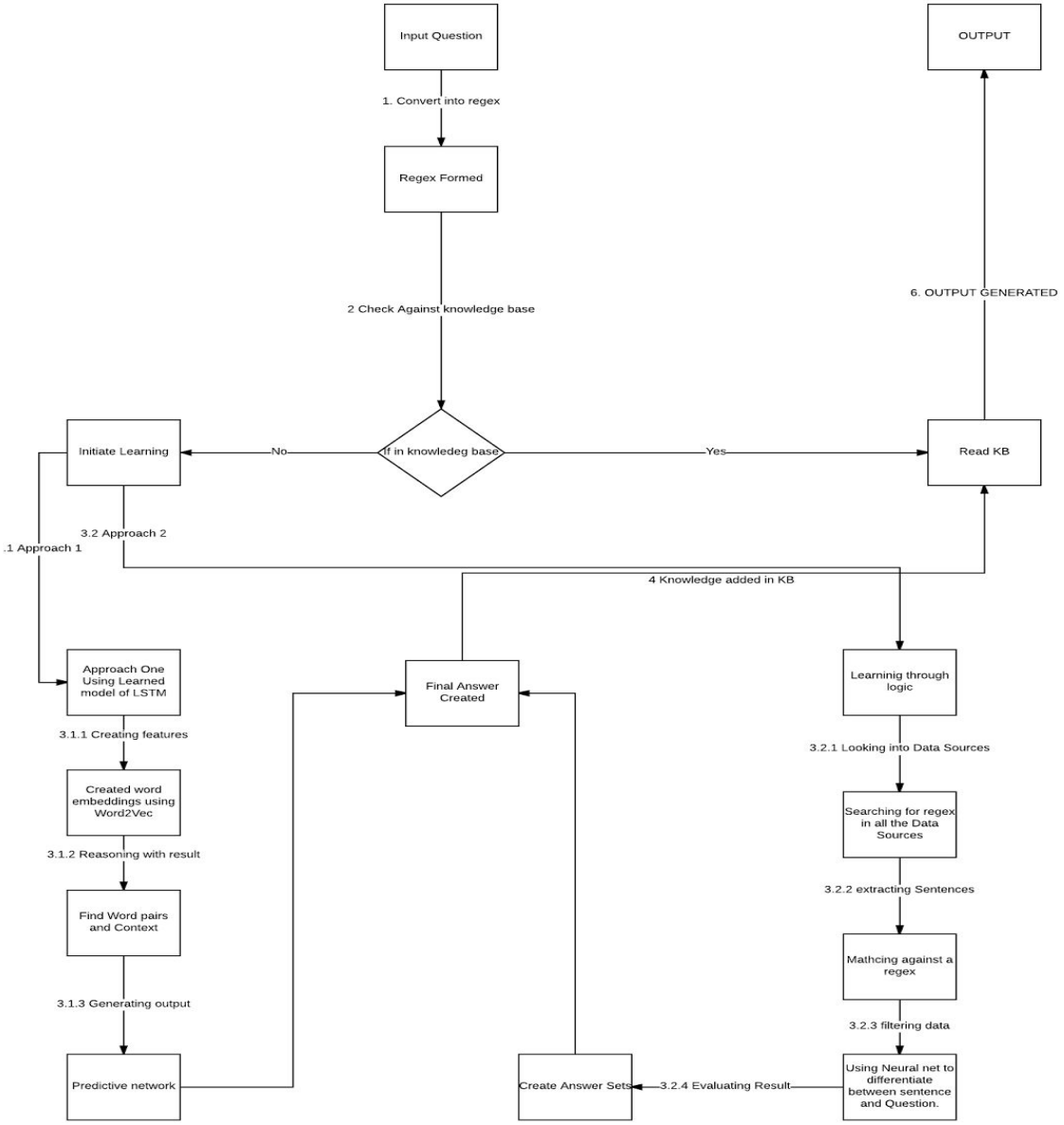


Fig. 7 Flowchart of our system

Approach 1

Approach Description

Step 1: Converting input to regex

In this step we convert input sentence into regular expression that can be passed to next step to generate a search query. In this step the sentence is tokenized into words using nltk word tokenize and language tags are assigned to each word using nltk.pos_tag and Finally we generate the regular expression by ignoring the tags “DT”, “PDT”, “POS”, “PRP”, “PRP$”, “NN”, “NNS”, “NNP” and “NNPS” which are noun, pronoun, predeterminer, determiner.

Sentence: Sam can not lift his son because he is heavy.

Regular Expression: .*can not lift.*because.*is heavy

The knowledge base we are creating has the form “A.*Verb.*B.*cause.*A.*C.*”, “A” and “A.*Verb.*B.*cause.*B.*C.*”, “B”.

So input question is converted into form which is independent of the Noun. The Reason we are doing this is for finding answer we do not care about the Noun but relationship between Noun and verb.

Example:-

Sentence: Sam can not lift his son because he is heavy.

*Regular Expression: .*can not lift.*because.*is heavy*

Step 2: Check for the answer in KB.

If data is in Kb, then return answer.If not in Kb, then search for similar regex in various data sources(in our case, search engines).

In this project we are searching the regex in www.ask.com and crawling the data. We are looking for first 100 links returned by search engine.

Then we are Comparing the test data obtained from these resources and parsing them using the regex which is built using various datasets like Wordnet and propbank.

Example :- *Input: .*can not lift.*because.*is heavy*

Output regex:

((.\s((can|can|could)\s+not\s+(lift|lifted)).*?\s(because|after|to).*+?\s(()\s+(be|be|was|were|been|bing)\s+heavy))|(.*\s(()\s+(be|be|was|were|been|bing)\s+heavy).*?\s(so|in order to | but | that | then| , | when) .*+?\s((can|can|could)\s+not\s+(lift|lifted))))*

The above regex is looking for 3 things in particular :-

1. Taking care of Active and passive structure of the system.
2. Take care of the various forms of verb.
3. Take care of singular and plural form.

Step 3: Filtering statements Questions

A. Initial Approach:

The first approach we used was to converting each sentence into its word vectors and to see if we can differentiate between questions and statements. We used word2vec to get vector value of a word and then add it up get the vector value of a sentence. We used a neural net to train using this approach and learn from it. After the model has finished learning, testing it gave us a poor accuracy of 55% as two sentences like the ones below

Sentence 1: It is a tree.
Sentence 2: Is it a tree?

Both the above sentences have the same vector value. Hence this approach won't give the best results.

B. Stanford CoreNLP approach:

Differentiating between a question and statement can be quite hard especially without punctuation. We have to filter questions out because our goal is to find a specific knowledge in a given sentence and it needs to be a statement as knowledge of question in most cases differ from the knowledge that we intend to figure out. Hence filtering questions from a set of given sentences becomes critical.

We used Stanford CoreNLP to parse the sentence into its grammatical constituents. Their output returns a tree structure containing various levels of detail. But we only need the highest level because that would tell us if it's a question or statement. Constituency parsing at its highest level gives various attributes depending on sentence structure. The Highest level would contain the word SBARQ or SQ which are two types of questions. Hence when we use stanford CoreNLP's constituency parser, we can convert it's output to string and check for the string 'SBARQ' and 'SQ' to find out if it's a question.

Using this approach, we got an accuracy of 93.75% on the dataset 'QA_dataset.csv' submitted along with the report.

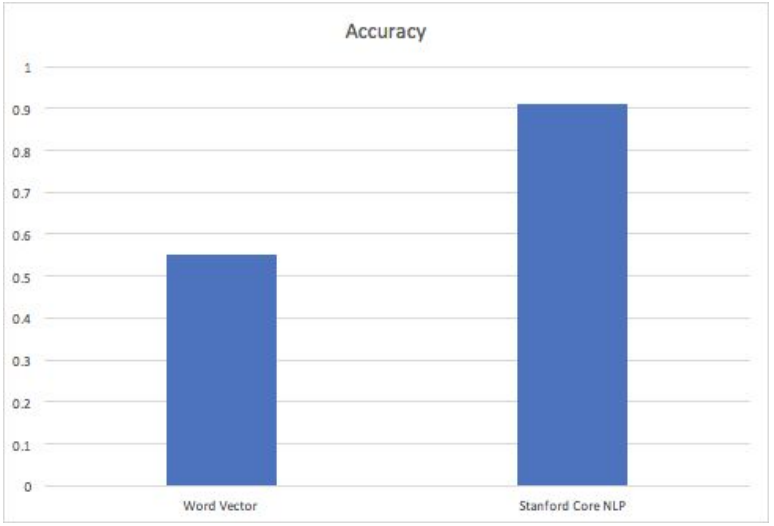


Fig. 8 Accuracy comparison

The one thing to note down is , the system is designed to work with the not only text as input , it is also designed to work with the Voice interfaces. So raw data is also designed when being filtered using the our system , it's not looking for "?" to differentiate for the question.

Then finally we are looking for the sample sentences which will meet our requirement and then we are returning the answer to Kb.

Approach 2

Approach Description

We (human beings) understand the language sequentially, we reason about each word we read and try to assign it to previous context we read. Oftentime we use context of word to determine the relationship between different words and understand its meaning.

In this approach we try to take this idea forward to work with the problem of coreference resolution. We can leverage the LSTM neural network's ability to predict the information sequentially. The word2vec model gives us contextual relationships how the words are related to each other. This can be helpful for predicting context similar to below sentence

The storm delayed the flight as it was very dangerous.

Here storm is contextually more similar to dangerous driver than the flight.

Method 1:

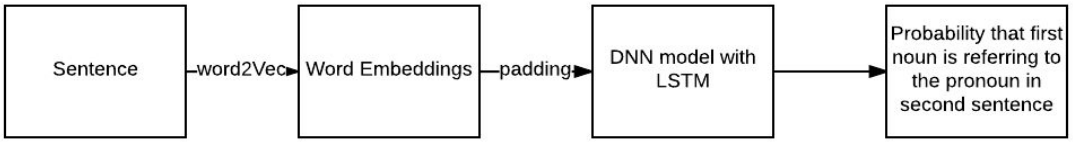


Fig. 9 Sen2Vec using DNN(4 hidden layer) and LSTM

- Convert sentence into the tokens using nltk
- Tokenize and assign a part of speech tag to each sentence using corenlp
- Tag Noun1 and Noun2
- Create word vectors from these tokens using word2vec model (length 300)
- Add zero padding to each sentence to make length of sentence same in each input
- Using this word tokens with padding and a word vector for the result, we train a vanilla LSTM (Keras) with output as Noun1 or Noun2.
- Then after collecting the results from lstm, we trained a neural network with 900 input units as the word vector output generated by lstm and two-word vectors for nouns with a binary result
- 1 in case pronoun in the second sentence refers to noun 1, 0 in case pronoun in the second sentence refers to noun 2

With vanilla LSTM with word2vec of sentences as input, we found that the predicted words were completely random, unrelated and out of context what we expected as per our requirement.

Therefore, we decided to try new model with some preprocessing in sentences for the contexts before feeding to LSTM. We should have word-pair context cluster.

Method 2:

Algorithm for Preprocessing (during training):

- Find part of speech tags in an input sentence so that we know noun1, noun2, pronoun1 and pronoun2 in the input sentence using corenlp part of speech tagging api.
- Find action word and related word-pair from the sentence. Also check if action word
- Map word pair to the word pair cluster (positive context, negative context).
- Break sentence in the following manner and then feed it to the LSTM.

Example:

Sentence1: Sam cannot lift his son because he is heavy.

Word-pair is lift-heavy and it should match to positive cluster (represents positive number).

Cannot – negative (represents negative number)

Final break down:

noun1 negative positive pronoun1 noun2 because pronoun2 noun1

Here noun1 refers to son.

In this sentence, we know that pronoun2 is for noun1 and during training, weights will be adjusted for each stage in LSTM architecture (selection, forgetting, ignoring, input).

Sentence 2: Sam cannot lift his son because he is weak.

Word-pair is lift-weak and it should match to negative cluster (represents negative number).

Cannot – negative (represents negative)

Final break down:

noun1 negative negative pronoun1 noun2 because pronoun2 noun2

Pronoun2 refers to noun2.

In this sentence, we know that pronoun2 is for noun2 and during training, weights will be adjusted for each stage in LSTM architecture (selection, forgetting, ignoring, input).

During testing, our LSTM network is to predict what should come after pronoun2 based on the weights decided for each stage earlier. In our basic implementation we selected some positive number and negative numbers between +1 and -1.

Below is the flow diagram of LSTM stages:

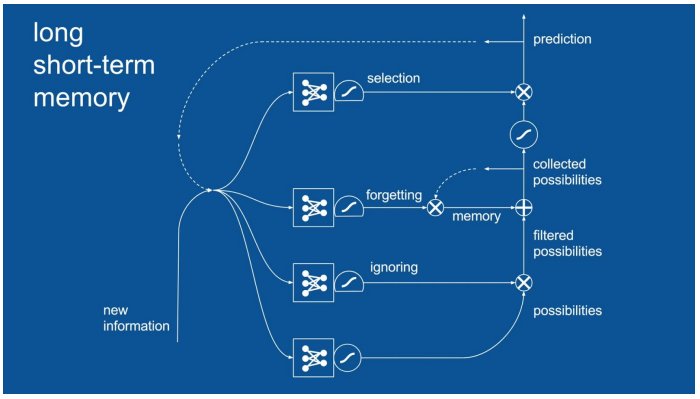


Fig. 10 Stages in LSTM (https://youtu.be/WCUNPb-5EYI?t=17m21s)

We implemented above model but got prediction only for very few sentences. We figured out that each conjunction also puts a big impact on over all context in the sentence. Therefore, there should to be some positive/negative representation of conjunctions as well. We could not come up with some good representation for conjunction though. Another reason, could be selecting integer representation for positive/negative labels which should driver weights properly. Because of lack of time, we could not test it and work more on it.

Method 3:

We first tried Coreference Resolution[10] method and used api provided by corenlp.

We found that it could figure out pronoun in simple sentences only. It was unable to provide correct prediction for complex sentences (as per our project needs).

We then tried the method and code implemented in [11]. It is based on Coreference Resolution[12] but using DNN.

We found that it's accuracy was 66% . Since We were directly using the weights which they provided (code and actual implementation for training is not available with https://github.com/huggingface/neuralcoref).

Therefore, we were not sure about what parameter were actually deciding the final weights during training and how the model was trained.

A thing to note is these two approaches are being runned in parallel.

Challenges:

Approach 1:

- Lack of dataset
- Takes time to answer a single query
- Difficult to categorize different regex types
- Requires search engines scrolling which are not available for free

Approach 2:

- Lack of context
- Lack of method which can generalize the solution.

Future Work

Approach 1:

- Training a model which can recognise various form of regular expression based on noun verb relation
- Better generalisation of the verb so that the search domain can be increased to answer the same query.

Approach 2:

- The language structure is tree based and it's difficult to perform pronoun disambiguation using sequential LSTM. The tree based LSTM which is recently proposed by "deep architecture for coreference resolution"[4] structure can be used.
- The dataset used in this project was smaller, the Approach 1 can be extended to collect similar sentence from web and improve quality of model.

Conclusion:

Semantic awareness is very important when we deal with applications in natural language processing. Only logic programming is not possible as the domain to search for the knowledge is very large , only statistical approach is not possible as nearest neighbors without keeping semantic in feature lies in oppposite set. We need a approach which uses both. The system we implemented if backed up with paid search engine queries can bring results of any query with optimization. Approach one is data dependent and Approach 2 is limited in the context of not able to carry logic inside it. This was a good attempt in making a system which uses state-of-the art models along with historical learning using regex to come up with the answer.

Dataset & Libraries:

1. Stanford Core NLP[3]
2. WordNet[1]
3. Propbank[2]
4. K-parser
5. Word2vec[6]
6. Keras
7. Training data set:- "http://www.hlt.utdallas.edu/~vince/data/emnlp12/train.c.txt"
8. Testing data set:- "http://www.hlt.utdallas.edu/~vince/data/emnlp12/test.c.txt"

References:

1. https://wordnet.princeton.edu
2. https://verbs.colorado.edu/propbank/framesets-english/
3. https://stanfordnlp.github.io/CoreNLP/
4. https://cs224d.stanford.edu/reports/ChengXioa.pdf
5. https://arxiv.org/pdf/1503.00075.pdf
6. https://code.google.com/p/word2vec/
7. https://pdfs.semanticscholar.org/1154/0131eae85b2e11d53df7f1360eeb6476e7f4.pdf
8. Sharma,Arpit, etal. "Identifyingvariouskinds ofevent mentionsink-parser Proceedingsofthe3rdWorkshoponEVENTSattheNAACL-HLT.2015.
9. https://arxiv.org/abs/1611.04146
10. https://nlp.stanford.edu/projects/coref.shtml
11. https://medium.com/huggingface/state-of-the-art-neural-coreference-resolution-for-chatbots-3302365dcf30
12. https://en.wikipedia.org/wiki/Coreference