Extraction and Systematic Synthesis of Textual Information by Contextualization and Enrichment : Applications of Automated Reasoning in the Cognitive Sciences

John Bagiliko[a,c,*,1,3], Kaninda MUSUMBU[b,d] and Saliou Diouf[a,c,**,1,3]

[a]AIMS, Km2 route de Joal (Centre IRD), B.P. 1418, Mbour - Senegal
[b]Labri - Université Bordeaux 1 351, cours de la Libération 33405 TALENCE Cedex
[c]AIMS, Km2 route de Joal (Centre IRD), B.P. 1418, Mbour - Senegal

ARTICLE INFO

Keywords:
Artificial Intelligence
Commonsense Reasoning
Automated Reasoning
Language Models
Wingograd Schema

ABSTRACT

One of the major problems in AI and Deep Learning is Commonsense Reasoning. As easy as it is for humans to perform certain tasks without having to think and waste much time, machines have difficulties in performing those tasks without necessarily been programmed.The Winograd schema is one of the recommended ways for testing the Commonsense reasoning ability of machines. It is difficult for machines to answer this Winograd Schema. In this work, we propose the use of neural network based on Language Models in tackling this problem. Our network takes in only word inputs for the training on large vocabulary size. This model attains an accuracy of 54.58 percent when ran on the Winograd Schema Challenge.

1. Introduction

Throughout human history, people have used technology to model themselves. There is evidence of this from ancient China, Egypt, and Greece that bears witness to the universality of this activity [1]. According to [2], Artificial Intelligence (AI) is the art, science and engineering of making intelligent agents, machines and programs. The field aims at providing solutions for one simple yet extremely tough objective, 'Can machines think and reason like human beings?' The central goal of AI is to understand the principles that make intelligent behavior possible in natural or artificial systems [1]. Some objectives of AI include emulation of cognitive learning, semantics, and knowledge representation, learning, reasoning, problem solving, planning and natural language processing [3]. Some researchers say that the problem of automating human reasoning is the lack of commonsense knowledge or commonsense reasoning. In the paper [4], this was explicitly stated when the authors said that current computer programs intended for tackling language tasks succeed only by manipulating individual words or short phrases. Common sense is usually evaded in order to just focus on short-term results. Unfortunately, it is hard to see how human level understanding can be achieved without greater attention to common sense. This can also be seen in the book [5] when the author pointed out the common difficulty of creating a thinking machine is the result of the lack of common sense knowledge in AI programs. Commonsense knowledge as opposed to specialized knowledge refers to the things that most people can do, often without conscious thought. In [6], the authors stated that an important research problem in AI is the logical formalization of common sense reasoning. [7] presents three reasons why AI should emphasize on commonsense reasoning rather than scientific knowledge. Several researchers [5] have tackled the problem of understanding commonsense reasoning.

[8] described a simple neural language model that relies only on character-level inputs. Their model employs a convolutional neural network (CNN) and a highway network over characters, whose output is given to a long short-term memory (LSTM) recurrent neural network language model (RNN-LM). On the English Penn Treebank [11], the model is on par with the existing state-of-the-art despite having sixty percent fewer parameters. On languages with rich morphology (Arabic, Czech, French, German, Spanish, Russian), the model out-performs word-level/morpheme-level LSTM baselines, again with fewer parameters. The results suggest that on many languages, character inputs are

john.bagiliko@aims-senegal.org (J. Bagiliko); musumbu@labri.u-bordeaux.fr (K. MUSUMBU); saliou_diouf@aims-senegal.org (S. Diouf)
https://john-bagiliko.github.io (J. Bagiliko); http://www.labri.fr/perso/musumbu/ (K. MUSUMBU); https://www.researchgate.net/profile/Saliou_Diouf3 (S. Diouf)
ORCID(s):

sufficient for language modeling. Analysis of word representations obtained from the character composition part of the model reveals that the model is able to encode, from characters only, both semantic and orthographic information.

[10] considered two recurrent Language Models (LMs) where one processes word inputs while the other processes character inputs. While the input layers takes both word and character inputs, the output layer was designed to give only word output. Their base model consists of two Long Short-Term Memory (LSTM) with 8192 hidden units. The output gate of each LSTM uses peepholes and a projection layer to reduce its output dimensionality to 1024. They performed drop-out on LSTM's outputs with probability 0.25. For word inputs, they used an embedding lookup of 800000 words, each with dimension 1024. For character inputs, they use an embedding lookup of 256 characters, each with dimension 16. They concatenated all characters in each word into a tensor of shape (word length, 16) and add to its two ends the $< beginofword >$ and $< endofword >$ tokens. The resulting concatenation was zero-padded to produce a fixed size tensor of shape (50, 16). This tensor was then processed by eight different 1-D convolution (Conv) kernels of different sizes and number of output channels, each followed by a Rectified Linear Units (ReLU) activation.

We follow from the works of [8] and [10], that is we use the Neural Network approach but we use only word inputs in training our model. It should be noted that in [8], only character inputs were used while both character and word inputs were used in [10]. We choose to use only word inputs since the use of only characters and or both character and word inputs in training a neural network model may lead to the creation of a supper bot and not a thinking machine. Since a reasoning machine is desired, we think it is reasonable and intuitive to allow the machine to interact with only the words in a language (English for example). The previous works as have already been mentioned involved the use of convolutional neural network but our do not implement that in our work.

2. Language Models (LMs)

Language modeling [12] is the task of assigning a probability to sentences in a language ('what is the probability of seeing the sentence the lazy dog barked loudly ?'). Besides assigning a probability to each sequence of words, the language models also assigns a probability for the likelihood of a given word (or a sequence of words) to follow a sequence of words ('what is the probability of seeing the word barked after the seeing sequence the lazy dog ?'). The probability [10] is able to be assigned to any given text based on what the LMs have learned from training data.

Formally, the task of language modeling is to assign a probability to any sequence of words $w_{1:n}$, i.e. to estimate $P(w_{1:n})$. Using the chain-rule of probability, this can be rewritten as:

$$P(w_{1:n}) = P(w_1)P(w_2 \mid w_1)P(w_3 \mid w_{1:2})P(w_4 \mid w_{1:3}) \dots P(w_n \mid w_{1:n-1}) \tag{1}$$

That is, a sequence of word-prediction tasks where each word is predicted conditioned on the preceding words. Even though it seems manageable to model the probability of the occurrence of a single word based on its left context than assigning a probability score to an entire sentence, the last term in the equation, still requires conditioning on $n-1$ words, which is as hard as modeling an entire sentence. For this reason, language models make use of the markov-assumption , stating that the future is independent of the past given the present. More formally, a kth order markov-assumption assumes that the next word in a sequence depends only on the last k words:

$$P(w_{i+1} \mid w_{1:i}) \approx P(w_{i+1} \mid w_{i-k:i}).$$

Estimating the probability of the sentence then becomes

$$P(w_{i:n}) \approx \prod_{i=1}^{n} P(w_i \mid w_{i-k:i-1}) \tag{2}$$

where $w_{-k+1}, \dots w_0$ are defined to be special padding symbols. The task is then to accurately estimate $P(w_{i+1} \mid w_{i-k:i})$ given large amounts of text.

LMs [10] are trained on text corpora, which encodes human knowledge in the form of natural language.

3. Data and Model Formulation

We used the Brown Corpus that comes with the Natural Language Toolkit (NLTK) in Python. In our neural network model, we take only word inputs unlike that in the current state-of-the-art where both word inputs and character inputs

are taken. This model only requires that the vocabulary size be large. Intuitively, humans know what they know as a result of their interactions with the things around them. It is, therefore, logical that the vocabulary size for training be large. This means that the machine should have seen sentences or words similar to those in the Winograd Schema. In the current state-of-the-art, an embedding look-up of 800000 words was used. The data and the saved model files of the current-state-of-the-art is about 280 Gigabytes. In our case, we use about 2300 sentences, approximately 371350 words in embedding look-up layer. We did not perform any drop-outs since our data is really small. Brown corpus, from which our data was extracted, contains about 57340 sentences which is about 1318820 words.

4. Results

Our model returns the probability of the occurrence of a sentence using language models. Based on this, we wrote a python function that takes in two sentences, the Winograd pair for example, and returns the sentence with the highest probability as the the correct answer as in Figure 2.

```
sent1 = "The trophy doesn't fit into the brown suitcase because the trophy is too large."
sent2 = "The trophy doesn't fit into the brown suitcase because the suitcase is too large."
answer_winograd(sent1, sent2)
```

```
Finding probability of sentence 2 ================================================
P(w=the|h=)=0.2537929117679596
P(w=trophy|h=the)=0.01192901935428381
P(w=doesn't|h=the trophy)=0.7991780638694763
P(w=fit|h=the trophy doesn't)=0.9127412438392639
P(w=into|h=the trophy doesn't fit)=0.945506751537323
P(w=the|h=the trophy doesn't fit into)=0.9999738931655884
P(w=brown|h=the trophy doesn't fit into the)=0.8304048776626587
P(w=suitcase|h=the trophy doesn't fit into the brown)=0.7648259997367859
P(w=because|h=the trophy doesn't fit into the brown suitcase)=0.9966496825218201
P(w=the|h=the trophy doesn't fit into the brown suitcase because)=0.9954020977020264
P(w=trophy|h=the trophy doesn't fit into the brown suitcase because the)=0.08635813742876053
P(w=is|h=the trophy doesn't fit into the brown suitcase because the trophy)=0.9718054533004761
P(w=too|h=the trophy doesn't fit into the brown suitcase because the trophy is)=0.9667530655860901
P(w=large|h=the trophy doesn't fit into the brown suitcase because the trophy is too)=0.7807540893554688
Probability of sentence1: 8.333643772195412e-05


Finding probability of sentence 2 ================================================
P(w=the|h=)=0.001070535508915782
P(w=trophy|h=the)=0.0010647987946867943
P(w=doesn't|h=the trophy)=0.0010586410062387586
P(w=fit|h=the trophy doesn't)=0.0010616992367431521
P(w=into|h=the trophy doesn't fit)=0.0010658823885023594
P(w=the|h=the trophy doesn't fit into)=0.0010673125507310033
P(w=brown|h=the trophy doesn't fit into the)=0.0010619070380926132
P(w=suitcase|h=the trophy doesn't fit into the brown)=0.0010609260061755776
P(w=because|h=the trophy doesn't fit into the brown suitcase)=0.0010676911333575845
P(w=the|h=the trophy doesn't fit into the brown suitcase because)=0.001062823343090713
```

Figure 1: Function takes in two sentences and selects the correct one

```
sent1 = "The trophy doesn't fit into the brown suitcase because the trophy is too large."
sent2 = "The trophy doesn't fit into the brown suitcase because the suitcase is too large."
answer_winograd(sent1, sent2)
```

```
Prob. sentence1: 8.333643772195412e-05
Prob. sentence2: 2.373675446089101e-42
sentence1 is correct
```

Figure 2: Function takes in two sentences and selects the correct one

ISSN: 2455-9210

```
sentence = "Emma did not pass the ball to Janie although Emma saw that she was open."
tok = tokenizer.texts_to_sequences([sentence])[0]
if len(tok)==0:
    tok = [1]
x_test, y_test = prepare_sentence(tok, maxlen)
x_test = np.array(x_test)
y_test = np.array(y_test) - 1  # The word <PAD> does not have a class
p_pred = model.predict(x_test)
vocab_inv = {v: k for k, v in vocab.items()} #inverts the vocabulary (vocab)
log_p_sentence = 0
for i, prob in enumerate(p_pred):
    word = vocab_inv[y_test[i]+1]  # Index 0 from vocab is reserved to <PAD>
    history = ' '.join([vocab_inv[w] for w in x_test[i, :] if w != 0])
    prob_word = prob[y_test[i]]
    log_p_sentence += np.log(prob_word)
    print('P(w={}|h={})={}'.format(word, history, prob_word))
print('Prob. sentence: {}'.format(np.exp(log_p_sentence)))
```

```
P(w=emma|h=)=0.0013499594060704112
P(w=did|h=emma)=0.007280191406607628
P(w=not|h=emma did)=0.007922312244772911
P(w=pass|h=emma did not)=0.0007475197198800743
P(w=the|h=emma did not pass)=0.6769769191741943
P(w=ball|h=emma did not pass the)=0.005543738603591919
P(w=to|h=emma did not pass the ball)=0.645117998123169
P(w=janie|h=emma did not pass the ball to)=0.033649321645498276
P(w=although|h=emma did not pass the ball to janie)=0.004874133970588446
P(w=emma|h=emma did not pass the ball to janie although)=0.010838935151696205
P(w=saw|h=emma did not pass the ball to janie although emma)=0.015375128947198391
P(w=that|h=emma did not pass the ball to janie although emma saw)=0.09928759932518005
P(w=she|h=emma did not pass the ball to janie although emma saw that)=0.046464305371046066
P(w=was|h=emma did not pass the ball to janie although emma saw that she)=0.020092211663722992
P(w=open|h=emma did not pass the ball to janie although emma saw that she was)=0.0002191112143918872
Prob. sentence: 7.822398227916294e-29
```

Figure 3: Model predicts probability of correct sentence

ISSN: 2455-9210

```
sentence = "Emma did not pass the ball to Janie although Janie saw that she was open."
tok = tokenizer.texts_to_sequences([sentence])[0]
if len(tok)==0:
    tok = [1]
x_test, y_test = prepare_sentence(tok, maxlen)
x_test = np.array(x_test)
y_test = np.array(y_test) - 1  # The word <PAD> does not have a class
p_pred = model.predict(x_test)
vocab_inv = {v: k for k, v in vocab.items()} #inverts the vocabulary (vocab)
log_p_sentence = 0
for i, prob in enumerate(p_pred):
    word = vocab_inv[y_test[i]+1]  # Index 0 from vocab is reserved to <PAD>
    history = ' '.join([vocab_inv[w] for w in x_test[i, :] if w != 0])
    prob_word = prob[y_test[i]]
    log_p_sentence += np.log(prob_word)
    print('P(w={}|h={})={}'.format(word, history, prob_word))
print('Prob. sentence: {}'.format(np.exp(log_p_sentence)))
```

```
P(w=emma|h=)=0.0013499594060704112
P(w=did|h=emma)=0.007280191406607628
P(w=not|h=emma did)=0.007922312244772911
P(w=pass|h=emma did not)=0.0007475197198800743
P(w=the|h=emma did not pass)=0.6769769191741943
P(w=ball|h=emma did not pass the)=0.005543738603591919
P(w=to|h=emma did not pass the ball)=0.645117998123169
P(w=janie|h=emma did not pass the ball to)=0.033649321645498276
P(w=although|h=emma did not pass the ball to janie)=0.004874133970588446
P(w=janie|h=emma did not pass the ball to janie although)=0.007227479480206966
P(w=saw|h=emma did not pass the ball to janie although janie)=0.023134488612413406
P(w=that|h=emma did not pass the ball to janie although janie saw)=0.010628136806190014
P(w=she|h=emma did not pass the ball to janie although janie saw that)=0.0024075827095657587
P(w=was|h=emma did not pass the ball to janie although janie saw that she)=0.12163598090410233
P(w=open|h=emma did not pass the ball to janie although janie saw that she was)=0.0006617018370889127
Prob. sentence: 7.958609736504935e-30
```

Figure 4: Model predicts probability of wrong sentence

Figures 4 and 3 shows the conditional probabilities output by the model.

We visualize these probabilities in another test pair of correct sentence and wrong sentence in 5 and 6 respectively

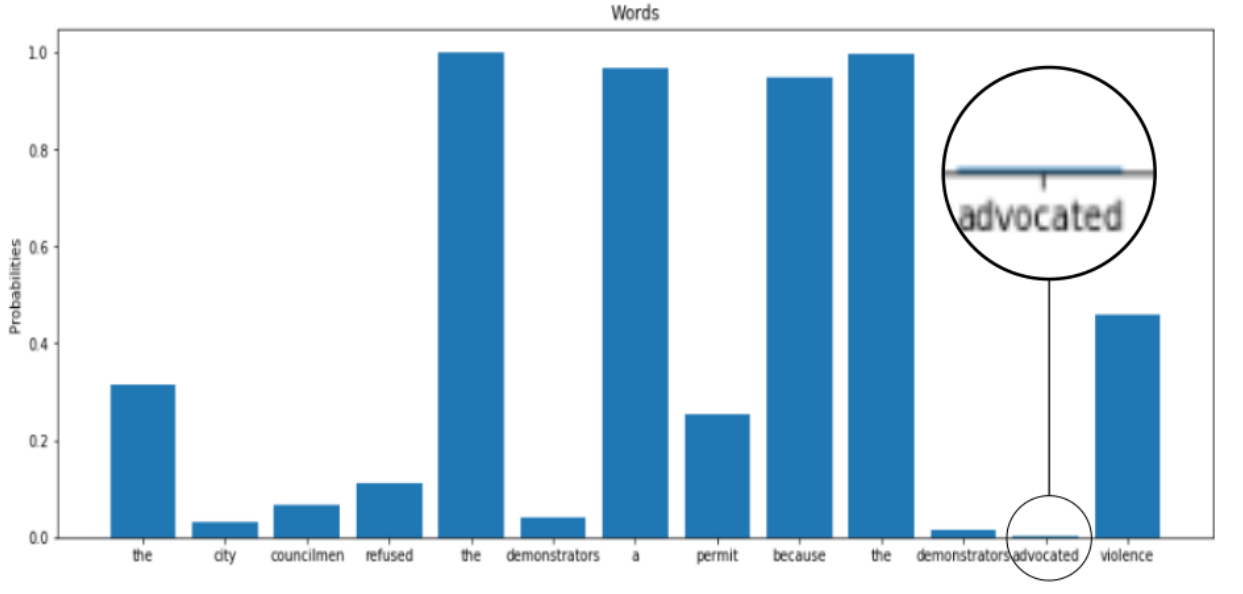

Figure 5: Conditional probabilities of words in a correct sentence

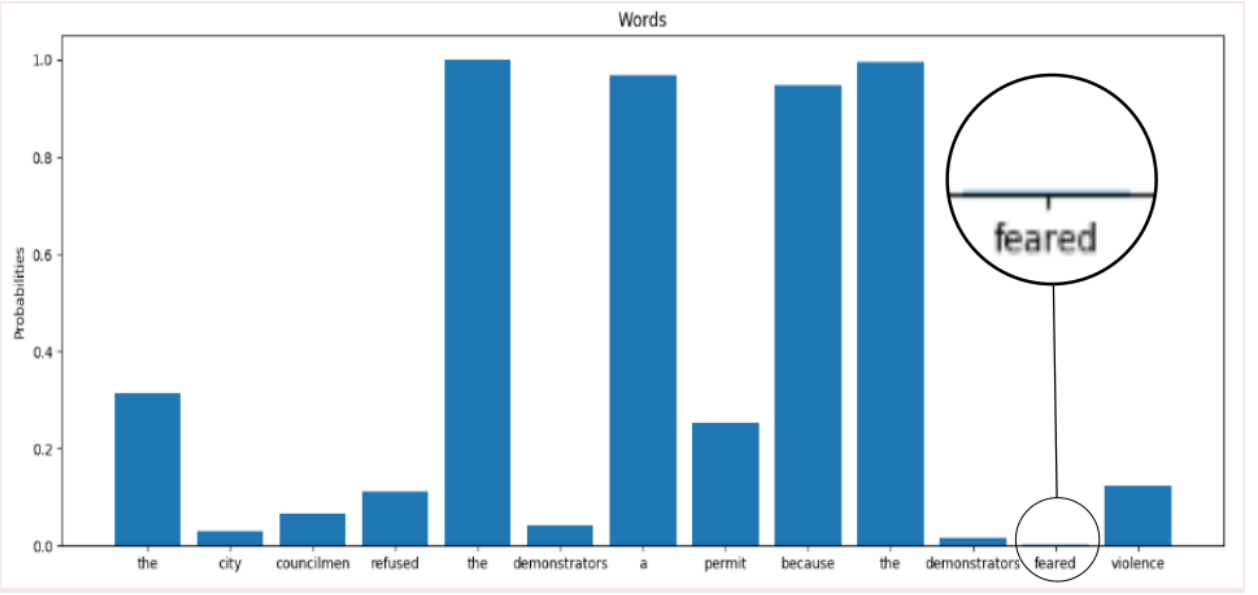

Figure 6: Conditional probabilities of words in a wrong sentencey

It can be seen in figures 5 and 6 that the obvious difference between the two sentences starts from the columns labeled "advocated" and "feared" respectively. The probability value for "feared" is so small and close to 0. On the contrary, the probability of "advocated" in figure 5 is seen a bit. This is the condition that the word "advocated" appears in the sentence on condition that the other words to its left have already appeared.

5. Testing Model on the Winograd Schema

As can be seen from Figure 7 below, our model as tested on the Winograd Schema Challenge attains an accuracy of about 54.58 percent, thus it answers over 140 questions correctly out of the 273 questions. It is unable to make a decision on about 60 questions; and it answers about 60 questions wrongly. This is as a result of insufficient training data.

```
Accuracy of Neural LM model on Winograd:  54.57875457875458 %
==== Finished prediction successfully! ====
```

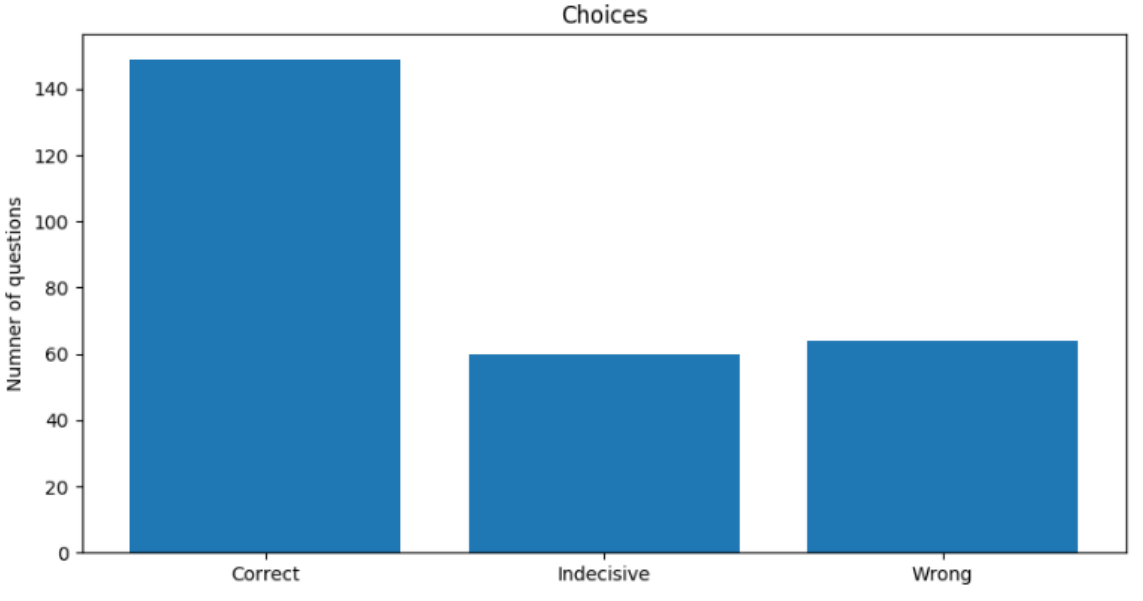

Figure 7: Model answers over 140 questions correctly

6. Discussion

It should be noted that our contribution in this work is the use of only word input in our neural network model while both character and word inputs were used in the current state of the art. In figure 1 is the output of our python function which calls our model and runs it on two sentences (one correct sentence and one wrong sentence). We see the probabilities of each word displayed. In Figure 2, it is clear that our model is able to compute the probabilities of each of the two sentences. Figure 3 displays the probabilities of words in a correct sentence while figure 4 displays probabilities of words in a wrong sentence. Figure 5 and Figure 6 visualize the probabilities of words in a correct sentence and a wrong sentence respectively. The difference between the two bar graphs can be seen by considering the last two bars in each. The last but one bar in the correct sentence (figure 5) is higher than the one in the wrong sentence (Figure 6). This indicates that the probability of the word "advocated" appearing in Figure 5 to make the sentence correct is higher than the probability of the word "feared" appearing in that same position of the sentence as in Figure 6. This is further seen in the last bars in both sentences. The last bar in Figure 5 is far higher than that in Figure 6 for the same reason. It could be concluded that our model is able to detect some special words that either make the sentence correct or wrong. Better still, all the bars in the two figures are similar except the last two bars. The special word is detected as soon as it makes one of the sentences wrong while another word, appearing in that same position, makes the sentence correct. The difference in the probabilities is seen clearly in the two figures. Finally, Figure 7 displays the results of our model when tested on the Winograd schema dataset (it is made up of 273 questions). The figure indicates that our model is able to answer about 150 questions correctly; it is unable to make a decision on about 60 questions and answers about 60 wrongly. The fact that it could not make a decision on some questions while answering some wrongly may be due to the limited size of our training dataset. It may be realized that our results is not bad as compared to that in the current state-of-the-art even though our training dataset was very small. This is due to fact that our training dataset and majority of their training datasets come from the same source and may have similar statistics. Also, it could be due to the fact that some words in our vocabulary have repeated themselves multiple times in their training dataset since theirs is very large.

7. Conclusions

At the end of the work, we have been able to perform the following pertaining to our objectives:

- extract and synthesize textual data (corpora) for analysis such as the Brown's corpus from the NLTK package and also the Winograd Schema questions from [10].
- train a neural model (neural network) on the training data (corpus) based on language models
- test this model on some of the Winograd Schema Challenge questions and the model answers questions with an accuracy of 54.58 percent.

References

[1] D. L. Poole and A. K. Mackworth, *Artificial Intelligence: Foundations of Computational Agents* CAMBRIDGE UNIVERSITY PRESS, NY, USA, 2010.
[2] D. Sarkar, R. Bali and T. Sharma, *Practical Machine Learning with Python: A Problem-Solvers Guide to Building Real-World Intelligent Systems* Springer Science, New York, USA, 2018.
[3] D. Sarkar, *Text Analytics with Python: A Practical Real-World Approach to Gaining Actionable Insights from your Data* Apress, Berkeley, CA, 2016.
[4] E. Davis and G. Marcus, COMMUNICATIONS OF THE ACM **58**, 92 (2015).
[5] R. Nakatsu, *DIAGRAMMATIC REASONING IN AI* John Wiley and Sons, Inc., Hoboken, New Jersey, 2010.
[6] M. P. Bonacina and A. Martelli, Intelligenza Artificiale **3**, 14 (2006).
[7] J. McCarthy, *Artificial Intelligence, Logic and Formalizing Common Sense* (In: Thomason R.H. (eds) Philosophical Logic and Artificial Intelligence. Springer, Dordrecht, 1989).
[8] Y. Jernite, D. Sontag, and A. M. Rus, CoRR **abs/1508.06615**,(2018).
[9] S. Ye, T. Chua, M. Kan, and L. Qiu, Information Processing and Management **43**, 1463 (2007).
[10] T. H. Trinh and Q. V. Le, *A Simple Method for Commonsense Reasoning* (2018).
[11] Bies, Ann, J. Mott and C. Warner, *English News Text Treebank: Penn Treebank* Web Download. Philadelphia: Linguistic Data Consortium, 2015.
[12] G. Yoav *Neural Network Methods for Natural Language Processing* , 2017.