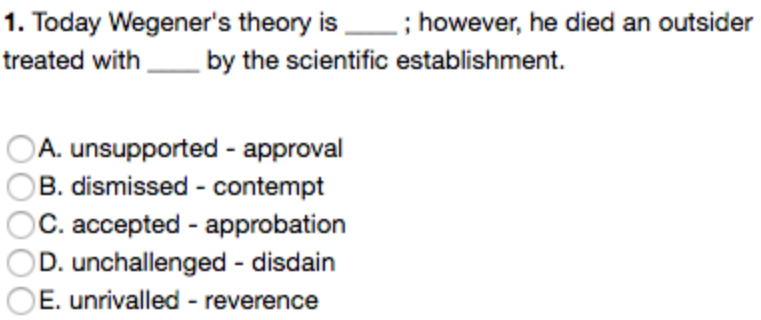
Project Proposal

Brian Xu, bwxu@mit.edu
Heeyoon Kim, heeyoon@mit.edu
Michelle Huang, mkyhuang@mit.edu
Xiaoxue Liu, sarahliu@mit.edu

Abstract

SAT vocabulary questions involve selecting the best word or words out of the choices given to fill in a blank for a block of text.



Example SAT vocabulary question.

This project will solve these vocabulary questions by applying the ideas of n-gram models, parsing, and recurrent neural networks in order to correctly rank and identify the best solutions. In particular, we will score each option with our model and select the highest scoring answer as our solution.

Data/Corpora

To acquire data, we will create a database of past SAT vocabulary questions and solutions. We will also use a database of all SAT vocabulary words and their definitions for our model.

Baselines

We will use multiple baselines to compare our results, such as random answering and average SAT test scores. We will also compare it to previous similar classifiers.

Evaluation

To measure the performance of our system, we will analyze the accuracy of our model in answering easy, medium, and hard questions. We will also look at top 2 accuracy, where we check if the answer was one of our top two choices. Another consideration will be the speed of our algorithm and whether it works within a reasonable time span with a reasonable amount of resources.

Report link for project updates/ "lab notebook"

https://docs.google.com/document/d/1ATceKZNPIFECAnmQ0eaU2_6yU07pu4EiijlSCzAv0do/edit

Link to presentation:
https://docs.google.com/presentation/d/1f_LczTt9tygMW0Yd40zX4a9MN-Vjp0MNVC1BdAtZuIo/edit?usp=sharing

Github link for project (must be public and on github.mit.edu, not .com)

https://github.mit.edu/bwxu/6.864_project

Maybe Useful Links
1. https://cs.umd.edu/~miyyer/pubs/2014_qb_rnn.pdf - Using RNNs to do quizbowl questions
2. https://cs224d.stanford.edu/reports/StrohMathur.pdf - Using RNNs for question answering
3. https://www.aclweb.org/aclwiki/index.php?title=SAT_Analogy_Questions_(State_of_the_art) - SAT analogies database

Lab Notebook

Week 1

Everyone:

- TensorFlow tutorials
- Read https://www.microsoft.com/en-us/research/publication/computational-approaches-to-sentence-completion/
- https://www.microsoft.com/en-us/research/publication/a-challenge-set-for-advancing-language-modeling/ (dataset)
- http://www-nlp.stanford.edu/links/statnlp.html (more datasets)

Michelle:

- RNN

Sarah:

- RNN

Heeyoon:

- Unigram model

Brian:

- LSA model (maybe)
- Unigram model
- Add datasets into github

Datasets

Unigram Model

Week 2

Heeyoon and Michelle met with Yu to get recommendations on implementing the LSTM and bidirectional model. For the bidirectional model, we plan to train the data on the corpus, and then reverse all the sentences in the corpus and reverse

Links to read from Yu:

- http://nlp.stanford.edu/pubs/snli_paper.pdf
- https://arxiv.org/pdf/1512.08849.pdf

- https://arxiv.org/pdf/1603.07044v1.pdf
- https://arxiv.org/pdf/1509.06664v4.pdf

(Those links are mainly for the Question-Hypothesis model, not sure we're still using that anymore)

Tasks To Do:
Brian:

- Smoothing bigram model

Heeyoon, Sarah, Michelle:

- Continue working on RNN, implement reader for data
- Implement softmax of cosine similarity of question (word for word), and answer (input answer into sentence)
- Use bidirectional language model to train

Week 3 - 4

Brian:

- Made word embeddings model
- Accuracy of 30% on MSR data

Michelle, Heeyoon, and Sarah met up to create a different a version of the reader that would work with the WSJ set. Had to run through each

Week 5

Michelle, Heeyoon, Sarah coded the first version of the LSTM model with the readers, based on the code provided in the public Tensorflow github.
Takes a long time to run.
Heeyoon coded a version of the test reader that would fill in each question statement with each of the five answer choices, and output five sentences for each question.

Week 6

Started running things using MIT's OpenMind computing cluster, since it's easier to run GPU on OpenMind (as well as run multiple tasks).

Heeyoon started changing the model code so that it actually outputs a probability matrix after each epoch. Probability matrix represents the prob of each word occurring next.

Week 7/8

Heeyoon implemented a simple version of LSTM that actually outputted the most likely word to go in the blank. Currently only works one sentence at a time, since Tensorflow flattens all the sentences, so hard to know where blanks are (since we strip out blanks from test data).
Heeyoon and Sarah implemented a new version of test reader that stops at blank, and outputs a list of partial sentences as well as the word to id dictionaries. Used this in above code.

Heeyoon and Sarah ran by hand 20 sentences to test preliminary accuracy for one epoch one layer.

Made poster (mostly Michelle).

Brian enhanced the model so that it can run all 1040 sentences at once by creating a create_tensor function, so we can create 1040 tensors, one for each sentence. Ran the model for one epoch, two layers. Also implemented bidirectional model using the updated LSTM code.

Week 8

Running the bidirectional model and the LSTM model for 1040 sentences.
Writing paper.

Results

Bigram/Unigram Model

- Using Holmes Training Data
- Sentence Completion Challenge - 25%
- SAT vocab questions - 16%

Word Embeddings Model

- Uses Google word embeddings
- Sentence Completion Challenge - 30%
- SAT vocab questions - 27%

RNN Model

- Trained on WSJ
- Sentence Completion Challenge -
- SAT vocab questions -