Machine Learning - deep and structured Hw3

B00902006 何主恩
B00902053 馮　硯
B00902064 宋昊恩
R03922163 姜佳昀

May 29, 2015

1 Implementation

We utilize package *eigen* to execute our *c++* code in parallel. It is easy to use this package, just add the eigen package folder path into the compilation include path. To use this package, just download it and decompress it.
We also utilize several toolkits for parsing purposes, for example, package *nltk* in python. With this package, I can tokenize my sentences and works much easier. To use this package, just type *pip install –user nltk; python*, then *import nltk* and *nltk.download('all')*.
What's more, we use *word2vec* to create word vectors. This toolkit is really useful for our later experiments. To use this package, just download it and *make install* it.

2 Preprocessing

First, we remove noises like the headers, punctuations, and other special characters in the data. Then make all training files and the testing file as the corpus to learn the vector of every word by *word2vec* (*parameters:-cbow 1 -size 200 -window 8 -min-count 5 -negative 25 -hs 0 -sample 1e-4 -threads 20 -binary 0 -iter 15*). For *Recurrent Neural Network Language Model (RNNLM)*, we set a *sliding-window* of size N to make every sequence of N words in a sentence to be a training sample and concatenate *word2vec* vectors of the N words to be the feature vector.

3 Proposed Solution

To solve this Cloze test with multiple choice, we propose three solutions. They are *Random Guess*, *Cosine Similarity Wordwise Evaluation*, *Recurrent Neural Network Language Model (RNNLM)*.

3.1 Random Guess

3.1.1 Algorithms

This is a naive baseline. We simply guess (c) for all the questions.

3.1.2 Experiments Results

Onboard result:

- 0.20288

3.2 Cosine Similarity Wordwise Evaluation

3.2.1 Algorithms

In this approach, there are two steps to gain the proper result. First, use *word2vec* tool to transform each word into a fix-length vector. It is garanteed that the lower the cosine distance between two vectors, the higher similarities they share. With this property, we derive our second evaluation step: calculate the distance between word vector and sentence vector.
There are many feasible ways to define the distance between word vector and sentence vector. For example, calculate the maximum cosine similarity between every word in sentence and the proposed answer. To be clear, for a sentence of length l, we can formulate the sentence as $w_1, w_2, ..., w_{k-1}, w_{k+1}, ..., w_l$, where w_{k_i} is the ith choice for Cloze questions. Then, we need to find out $\mathbf{argmax}_i\, cos(w_{k_i}, w_j$ for j = 1:l, $j \neq k$. It is similar to define other similar evaluation matrices.

3.2.2 Experiments Results

Onboard result:

- old + max: 0.43750
- new + max: 0.44327
- new + sum: 0.44712

where

- *old* means to split all non alphanumeric characters
- *new* means to split all non alphanumeric characters, except *apostrophe* and *hyphen*
- *max* means to calculate the maximum cosine similarity within a sentence
- *sum* means to calculate the summation cosine similarity within a sentence

3.3 Recurrent Neural Network Language Model (RNNLM)

3.3.1 Algorithms

The implementation is just follow the steps describe in *BackPropagation Through Time*.
The only difference is that we further define the objective function as: the norm-2 distance between predicted word vector and the real subsequent word.

3.3.2 Experiments Results

4 Division of Teamwork

- 何主恩：RNNLM implementation, experiment
- 宋昊恩：preprocessing, consine similarity implementation,experiment
- 姜佳昀：parameters tuning, experiment