Assignment 5 Tutorial

CSC485, Fall 2015

Krish Perumal
krish@cs.toronto.edu

November 26, 2015

Part 1

- Use Word2Vec to generate embeddings (vector representations) of words in Stanford Sentiment Treebank corpus
- Experiment with training parameters and do some qualitative and quantitative analysis

Part 1

- Use gensim (Implementation of Word2Vec in Python)
- Already downloaded on CDF; just set PYTHONPATH to access it:

```
setenv PYTHONPATH /u/frank/csc485/A5/modules/
```

- Extract sentences from Stanford Sentiment Treebank corpus using data_utils.py (code specified in handout)

Part 1

- Run Word2Vec on all sentences:

```
word2vec.Word2Vec(sentences, sg=sg, size=size,
window=window, min_count=min_count, iter=iter)
```

Parameter	Description
sg [1,0]	1 = skip-gram; 0 = continuous bag-of-words
size [2,50,200]	Number of vector dimensions
window [1,2,5]	Window length of surrounding words to predict given current word
min_count [1,5]	Minimum number of times a word must occur (else, it's not part of the model)
iter [1,5]	Number of training iterations

- Iterate through all possible parameter value combinations (2 X 3 X 3 X 2 X 2 = 72)

Part 1: Qualitative Analysis

- For each model, find (upto) 10 words most similar to: love, laugh, enjoy, yawn, hate, uncomfortable
- Write to a csv file
- Write a report (100-200 words):
  - What are the major trends with respect to parameter values and the kind of similar words?
  - Are there differences between using skip-gram and cbow? Explain.

Part 1: (Pseudo) Quantitative Analysis

- Use hand-curated set of analogies in:

`/u/frank/csc485/A5/data/questions-words.txt`

- This file contains analogous word-pairs:
  – dog dogs octopus octopodes
- Accuracy is determined by whether the word embeddings have similar distances between analogous word-pairings
  – vector(dog) – vector(dogs) + vector(octopus) ~= vector(octopodes)?

Part 1: (Pseudo) Quantitative Analysis

- Simply use:

```
acc = model.accuracy('/u/frank/csc485/A5/data/
questions-words.txt')
```

- This will return a list of dictionaries; example:

```
acc[i] {
  'incorrect': [(u'son', u'daughter',  u'king',
 u'queen')],
  'section': u'family',
  'correct': [(u'boy', u'girl', u'he', u'she')]
 }
```

- You only need care about list elements from indices 4 to 13 (14 is for total)

Part 2: Sentiment prediction

- Predict sentiments of words in:

`/u/frank/csc485/A5/data/stanfordSentimentTreebank/dictionary.txt`

- Format -- each line contains:
  - <word>|<word_id>
  - (Example: `happy|3342`)

- True sentiment values in:

`/u/frank/csc485/A5/data/stanfordSentimentTreebank/sentiment_labels.txt`

- Format -- each line contains:
  - <word_id>|<sentiment_value>
  - (Example: `3342|0.9`)

Part 2: Sentiment prediction

- Caveat!
  - The dictionary and sentiment_labels contain phrases too! Ignore them.

- Write all word vectors and their sentiment class labels into arff

- Predict sentiment using Weka (machine learning toolkit implementation in Java)

- You will predict only 5 classes of sentiments:
  [0,0.2] = 0, (0.2,0.4] = 1, (0.4,0.6] = 2, (0.6,0.8] = 3, (0.8,1] = 4

Part 2: Sentiment prediction

- Example arff:

@relation weather
@attribute outlook {sunny, overcast, rainy}
@attribute temperature numeric
@attribute humidity numeric
@attribute windy {TRUE, FALSE}
@attribute play {yes, no}
@data
sunny,85,85,FALSE,nosunny,80,90,TRUE,noovercast,83,86,FALSE,yesrainy,
70,96,FALSE,yesrainy,68,80,FALSE,yesrainy,65,70,TRUE,noovercast,
64,65,TRUE,yessunny,72,95,FALSE,nosunny,69,70,FALSE,yesrainy,
75,80,FALSE,yessunny,75,70,TRUE,yesovercast,72,90,TRUE,yesovercast,
81,75,FALSE,yesrainy,71,91,TRUE,no

Part 2: Sentiment prediction

- Running Weka:

```
java -cp /u/frank/csc485/A5/WEKA/weka.jar
weka.classifiers.functions.SMO *classifier options*
```

- Just add arff filename and the number of folds of cross-validation in classifier options and you're good to go!

Part 3: Smoothed soft-max

- Implement a function to compute conditional probability of context word (w_o) given current word (w_i) using soft-max

$$P(w_o \mid w_i) = \frac{\exp(V_{w_o}^{\top} v_{w_i})}{\sum_{w=1}^{||V||} \exp(V_{w}^{\top} v_{w_i})}$$

- Accommodate smoothing parameter, δ
- Input: Matrix θ (slide 26)
- Output: |V| X |V| matrix with all probability values
  - Ordering of elements -- choose one and specify it in comments)