Comparison of results based on window size changes

UIUC | CS410 Text Information Systems | Fall 2022
Eunbi Go (eunbigo2@illinois.edu)

Introduction

The purpose of this paper is to compare the results according to the changes in window size to determine whether the window size affects the results and to what extent the impact affects the results. For the comparison, 3 models will be trained with the different window sizes(2, 5 and 10) using Gensim Word2Vec. This paper will cover a step-by-step description of each process, outcome and conclusion. The process will include downloading a corpus(a collection of documents), preprocessing and training Word2Vec model with the preprocessed text data.

Process

1. Import packages needed

```
import re
import urllib.request
import zipfile
from lxml import etree
from nltk.tokenize import word_tokenize, sent_tokenize
import nltk
nltk.download('punkt')
```

2. Download and preprocess the training data

   I. Download

```
urllib.request.urlretrieve("https://raw.githubusercontent.com/
ukairia777/tensorflow-nlp-tutorial/main/09.%20Word%20Embedding/
dataset/ted_en-20160408.xml", filename="ted_en-20160408.xml")
```

   II. Preprocess

```
targetXML = open('ted_en-20160408.xml', 'r', encoding='UTF8')
target_text = etree.parse(targetXML)

# Extract contents between <content> </content>
parse_text = '\n'.join(target_text.xpath('//content/text()'))

# Remove parenthesis and contents in between
content_text = re.sub(r'\([^)]*\)', '', parse_text)

# Sentence Tokenization using nltk
sent_text = sent_tokenize(content_text)

normalized_text = []
for string in sent_text:
     tokens = re.sub(r"[^a-z0-9]+", " ", string.lower())
     normalized_text.append(tokens)

# Word Tokenization using nltk
result = [word_tokenize(sentence) for sentence in
normalized_text]
```

III. Verify that tokenization was successful

```
print('Sample size: {}'.format(len(result)))
for line in result[:3]:
    print(line)
```

Result:

```
Sample size: 273380
['here', 'are', 'two', 'reasons', 'companies', 'fail', 'they',
'only', 'do', 'more', 'of', 'the', 'same', 'or', 'they', 'only',
'do', 'what', 's', 'new']
['to', 'me', 'the', 'real', 'real', 'solution', 'to', 'quality',
'growth', 'is', 'figuring', 'out', 'the', 'balance', 'between',
'two', 'activities', 'exploration', 'and', 'exploitation']
['both', 'are', 'necessary', 'but', 'it', 'can', 'be', 'too',
'much', 'of', 'a', 'good', 'thing']
```

3. Training Word2Vec model

```
from gensim.models import Word2Vec
from gensim.models import KeyedVectors

# Adjust window size
model = Word2Vec(sentences=result, size=100, window=5, min_count=5,
workers=4, sg=0)
```

Outcome

After training the dataset, I was able to use gensim.models.Word2Vec.most_similar to find the top-10 most similar words and gensim.models.Word2Vec.similarity to compute cosine similarity between two words. The following is the outcome for each word.

- gensim.models.Word2Vec.most_similar: Computes cosine similarity between a simple mean of the projection weight vectors of the given words and the vectors for each word in the model. The method corresponds to the *word-analogy* and *distance* scripts in the original word2vec implementation.[1]

1. dog

Window Size	Outcome
2	[('doctor', 0.7880403995513916), ('chair', 0.7833065390586853), ('cat', 0.7829782366752625), ('leg', 0.7778382301330566), ('grandmother', 0.7714995741844177), ('mom', 0.7631133794784546), ('husband', 0.7620934844017029), ('wife', 0.7560390830039978), ('neighbor', 0.7516508102416992), ('uncle', 0.748279333114624)]
5	[('chair', 0.8188003301620483), ('cat', 0.8076136112213135), ('seat', 0.7683653831481934), ('nose', 0.7569484114646912), ('leg', 0.7563033103942871), ('uncle', 0.7498229742050171), ('mom', 0.7483735084533691), ('friend', 0.7347028255462646), ('wedding', 0.7331169843673706), ('doctor', 0.731247067451477)]
10	[('cat', 0.8208074569702148), ('leg', 0.8125799298286438), ('chair', 0.7728151082992554), ('nose', 0.7474145889282227), ('pants', 0.7473082542419434), ('seat', 0.7409422993659973), ('mouth', 0.7289161682128906), ('hair', 0.7244070768356323), ('bow', 0.7226544618606567), ('chest', 0.7169888019561768)]

2. texas

[1] gensim.models.Word2Vec.most_similar, https://tedboy.github.io/nlps/generated/generated/gensim.models.Word2Vec.most_similar.html

Window Size	Outcome
2	[('florida', 0.8814058303833008), ('beijing', 0.8679476976394653), ('oxford', 0.863100528717041), ('philadelphia', 0.8630014657974243), ('virginia', 0.8623799085617065), ('boston', 0.8594883680343628), ('seattle', 0.8581527471542358), ('turkey', 0.8546115159988403), ('paris', 0.8544856905937195), ('norway', 0.8523437976837158)]
5	[('florida', 0.8717864751815796), ('california', 0.8636471033096313), ('diego', 0.8527736067771912), ('philadelphia', 0.8503570556640625), ('spain', 0.8491007089614868), ('london', 0.8457138538360596), ('beijing', 0.8442980051040649), ('boston', 0.8374168872833252), ('chicago', 0.8354970812797546), ('france', 0.8353651762008667)]
10	[('florida', 0.8579306602478027), ('virginia', 0.8464271426200867), ('massachusetts', 0.8459047079086304), ('spain', 0.8444886207580566), ('boston', 0.8416600823402405), ('county', 0.8335837125778198), ('san', 0.8266251087188721), ('jordan', 0.8208940625190735), ('france', 0.8154868483543396), ('lagos', 0.8148953318595886)]

3. beer

Window Size	Outcome
2	[('microphone', 0.7955694198608398), ('cookie', 0.7814821600914001), ('jar', 0.7813495397567749), ('candy', 0.7787782549858093), ('knife', 0.7776947021484375), ('statue', 0.7766567468643188), ('dome', 0.7678611278533936), ('rug', 0.7649728059768677), ('powder', 0.7579901218414307), ('wine', 0.756824791431427)]
5	[('blanket', 0.7774106860160828), ('toilet', 0.7746675610542297), ('goat', 0.7713389992713928), ('knife', 0.7545533180236816), ('chicken', 0.7511787414550781), ('coat', 0.7456350326538086), ('chocolate', 0.7418212890625), ('needle', 0.7386212348937988), ('sandwich', 0.7371845245361328), ('silver', 0.7348338961601257)]
10	[('ginger', 0.7599152326583862), ('sweet', 0.7359527349472046), ('hat', 0.7284958362579346), ('chicken', 0.7262558937072754), ('drunk', 0.7034957408905029), ('sandwich', 0.698309600353241), ('juice', 0.6979900598526001), ('cheese', 0.6966890692710876), ('coat', 0.6964220404624939), ('candy', 0.6918215751647949)]

- gensim.models.Word2Vec.similarity

1. woman + queen / woman + desk

Window Size	Outcome woman + queen	Outcome woman + desk
2	0.59681004	0.28426167
5	0.658015	0.284533

10	0.66555214	0.27088872

2. beer + wine / beer + unicorn

Window Size	Outcome beer + wine	Outcome beer + unicorn
2	0.75682473	0.55874246
5	0.7318056	0.60330164
10	0.6812348	0.6158479

3. eye + noes / eye + phone

Window Size	Outcome eye + nose	Outcome eye + phone
2	0.6620888	0.49552277
5	0.70261985	0.49226564
10	0.72461283	0.44583383

Conclusion

Through these processes, I was able to confirm that the changes in window size does affect the results. However, by looking at the outcome of gensim.models.Word2Vec.most_similar, it looked like there was not a meaningful change. It was very difficult to conclude that the similarity is improving or deteriorating. But if we look at the outcome of gensim.models.Word2Vec.similarity, it becomes more clear that the window size can be task specific. Although I could not see a clear improvement or deterioration of the outcome, I could confirm that the results would not change significantly with the change of one hyper-parameter.

The Jupyter notebook file built for this paper is uploaded on github together.