Kadyrov_Daniel_Project_1

December 18, 2020

Daniel Kadyrov

Stevens ID: 10455680

CS557 - Natural Language Processing

Group 32 - Daniel Kadyrov

1 Part 1

Respond to any one of Chapter 5, 6, 7, or 8 Exercises of BKL. Report on the packages/programs used, to what it/they were applied, the results, and your interpretation.

1.1 Exercise 6.10

Suppose you wanted to automatically generate a prose description of a scene, and already had a word to uniquely describe each entity, such as the book, and simply wanted to decide whether to use in or on in relating various items, e.g., the book is in the cupboard versus the book is on the shelf. Explore this issue by looking at corpus data and writing programs as needed. Consider the following examples:

a. in the car versus on the train

b. in town versus on campus

c. in the picture versus on the screen

d. in Macbeth versus on Letterman

This was a problem that spoke to me as a New Yorker because of the amount of "on vs. in" arguements I have with friends from neighboring locations. Are you in line or on line? are you in NYC or on Long Island? My understanding of this problem, based on this chapter, is to generate a database of uses for "on" vs "in". I used the NLTK Brown corpus.

[23]:
```
from nltk.corpus import brown
from nltk.tokenize import word_tokenize

corpus = brown.tagged_sents()

sents = [
    "in the car",
    "on the train",
```

```
    "in town",
    "on campus",
    "in the picture",
    "on the screen",
    "in Macbeth",
    "on Letterman"
]

db = {}
for sent in sents:
    db[sent] = []
    flag = False

    sent_tok = word_tokenize(sent)

    length = len(sent_tok)
    for s in corpus:
        for i in range(len(s) - length):
            for j in range(length):
                if s[i+j][0] == sent_tok[j]:
                    flag = True
                else:
                    flag = False
                    break
            if flag:
                db[sent].append(s)

for key in db.keys():
    print("{}: {}".format(key, len(db[key])))
```

```
in the car: 9
on the train: 8
in town: 21
on campus: 1
in the picture: 5
on the screen: 6
in Macbeth: 1
on Letterman: 0
```

2 Part 2

2.1 Exercise 42

Use WordNet to create a semantic index for a text collection. Extend the concordance search program in Example 3-1, indexing each word using the offset of its first synset, e.g., wn.synsets('dog')[0].offset (and optionally the offset of some of its ancestors in the hypernym hierarchy).

[10]:
```
import nltk
from nltk.corpus import stopwords, wordnet

def semantic_index(text, ancestors=False):
    stopword = stopwords.words("english")
    lemmatizer = nltk.WordNetLemmatizer()

    array = []

    for t in text.split():
        t_lemma = lemmatizer.lemmatize(t.lower())

        if t_lemma not in stopword:
            try:
                index = str(wordnet.synsets(t_lemma)[0].offset())
                index += wordnet.synsets(t_lemma)[0].pos()

                if ancestors == True:
                    hp = str(wordnet.synsets(t_lemma)[0].hypernyms()[0].
 ↪offset())
                    hp += wordnet.synsets(t_lemma)[0].hypernyms()[0].pos()
                    index = " || ".join((index,hp))

                array.append(" || ".join((t, index)))
            except:
                array.append(t)
        else:
            array.append(t)

    return " ".join(array)
```

[11]:
```
from nltk.corpus import inaugural
from nltk.tokenize import sent_tokenize

sents = sent_tokenize(inaugural.raw())
max_sent = max(sents, key=len)

semantic_index(max_sent)
```

[11]:
```
'On this subject || 6599788n it might || 5030680n become || 149583v me better ||
5143558n to be silent || 1919428s or to speak || 941990v with diffidence; but as
something may || 15211484n be expected, the occasion, I hope, will be admitted
|| 817311v as an apology || 6633363n if I venture || 797878n to say || 14485526n
that if a preference, upon principle, of a free || 7947958n republican ||
10522633n government, formed || 2448185v upon long || 1828405v and serious ||
2118379a reflection, after a diligent || 1736122s and impartial || 1723308a
inquiry || 5797597n after truth; if an attachment || 7545161n to the
```

Constitution || 6533648n of the United || 2469835v States, and a conscientious || 310138s determination || 151497n to support || 1215902n it until it shall be altered || 126264v by the judgments || 5837957n and wishes || 7486229n of the people, expressed || 943837v in the mode || 4928903n prescribed || 747135v in it; if a respectful || 1993940a attention || 5702275n to the constitutions || 6533648n of the individual || 7846n States || 8654360n and a constant || 5858936n caution || 4664058n and delicacy || 4813066n toward the State || 8654360n governments; if an equal || 9626238n and impartial || 1723308a regard || 5820170n to the rights, interest, honor, and happiness || 13987423n of all the States || 8654360n in the Union, without preference || 7498210n or regard || 5820170n to a northern || 6949207n or southern, an eastern || 823971s or western, position, their various || 2065665s political || 1814385a opinions || 5945642n on unessential || 902652a points || 5865998n or their personal || 6271288n attachments; if a love || 7543288n of virtuous || 2513269a men || 8212347n of all parties || 8256968n and denominations; if a love || 7543288n of science || 5999797n and letters || 6624161n and a wish || 7486229n to patronize || 2219940v every || 2269794s rational || 13730469n effort || 786195n to encourage || 2554922v schools, colleges, universities, academies, and every || 2269794s institution || 8053576n for propagating || 2230447v knowledge, virtue, and religion || 5946687n among all classes || 7997703n of the people, not only for their benign || 2594565a influence || 5194151n on the happiness || 13987423n of life || 13963192n in all its stages || 15290337n and classes, and of society || 7966140n in all its forms, but as the only means || 6023969n of preserving || 2679899v our Constitution || 6533648n from its natural || 10346392n enemies, the spirit || 10636598n of sophistry, the spirit || 10636598n of party, the spirit || 10636598n of intrigue, the profligacy || 4894807n of corruption, and the pestilence || 14138691n of foreign || 1037540a influence, which is the angel || 9538915n of destruction || 217014n to elective || 890808n governments; if a love || 7543288n of equal || 9626238n laws, of justice, and humanity || 4829182n in the interior || 8588294n administration; if an inclination || 6196584n to improve || 205885v agriculture, commerce, and manufacturers || 8060446n for necessity, convenience, and defense; if a spirit || 10636598n of equity || 13333696n and humanity || 4829182n toward the aboriginal || 9676490n nations || 8168978n of America, and a disposition || 4623612n to meliorate || 205885v their condition || 13920835n by inclining || 335384n them to be more friendly || 8397489n to us, and our citizens || 9923673n to be more friendly || 8397489n to them; if an inflexible || 1024597a determination || 151497n to maintain || 2681795v peace || 13970236n and inviolable || 2510604a faith || 5946687n with all nations, and that system || 4377057n of neutrality || 1240850n and impartiality || 6202686n among the belligerent || 9939313n powers || 5190804n of Europe || 9275473n which has || 13888783n been adopted || 2346895v by this Government || 8050678n and so solemnly || 189960r sanctioned || 806502v by both Houses || 3544360n of Congress || 8161757n and applauded || 861929v by the legislatures || 8163273n of the States || 8654360n and the public || 8179689n opinion, until it shall be otherwise || 2071301s ordained || 2427916v by Congress; if a personal || 6271288n esteem || 14437552n for the French || 6964901n nation, formed || 2448185v in a residence || 8558963n of seven ||

13744916n years || 15203791n chiefly || 73897r among them, and a sincere || 2179279a desire || 7484265n to preserve || 14515463n the friendship || 13931145n which has || 13888783n been so much || 13776621n for the honor || 6696483n and interest || 5682950n of both nations; if, while the conscious || 1337767s honor || 6696483n and integrity || 14460565n of the people || 7942152n of America || 9044862n and the internal || 948670a sentiment || 7481951n of their own power || 5190804n and energies || 11452218n must || 9363970n be preserved, an earnest || 13350182n endeavor || 796886n to investigate || 789138v every || 2269794s just cause || 7326557n and remove || 5090255n every || 2269794s colorable pretense || 754956n of complaint; if an intention || 5982152n to pursue || 2375131v by amicable || 1246579a negotiation || 7148192n a reparation || 13292613n for the injuries || 14285662n that have been committed || 2582615v on the commerce || 1090446n of our fellow-citizens by whatever || 2267686s nation, and if success || 7319103n can not be obtained, to lay || 7049713n the facts || 5817396n before the Legislature, that they may || 15211484n consider || 690614v what further measures || 174412n the honor || 6696483n and interest || 5682950n of the Government || 8050678n and its constituents || 3081021n demand; if a resolution || 6511874n to do justice || 4850117n as far || 8016900n as may || 15211484n depend || 2664234v upon me, at all times || 7309599n and to all nations, and maintain || 2681795v peace, friendship, and benevolence || 7545717n with all the world; if an unshaken || 1991783s confidence || 5697363n in the honor, spirit, and resources || 13331778n of the American || 9738708n people, on which I have so often || 35058r hazarded || 916909v my all and never || 20759r been deceived; if elevated || 3280813n ideas || 5833840n of the high || 5097536n destinies || 7330007n of this country || 8168978n and of my own duties || 1129920n toward it, founded || 2427103v on a knowledge || 23271n of the moral || 6606044n principles || 5913538n and intellectual || 9621545n improvements || 7357388n of the people || 7942152n deeply || 173353r engraven on my mind || 5611302n in early || 812952a life, and not obscured || 2157731v but exalted || 860620v by experience || 5758059n and age; and, with humble || 1801697v reverence, I feel || 5677340n it to be my duty || 1129920n to add, if a veneration || 7521039n for the religion || 5946687n of a people || 7942152n who profess || 819756v and call || 6272803n themselves Christians, and a fixed || 260648v resolution || 6511874n to consider || 690614v a decent || 1993408s respect || 5820170n for Christianity || 6226057n among the best || 127531n recommendations || 6671637n for the public || 8179689n service, can enable || 512877v me in any degree || 5093890n to comply || 2542280v with your wishes, it shall be my strenuous || 875235s endeavor || 796886n that this sagacious || 2569558s injunction || 7170467n of the two || 13743269n Houses || 3544360n shall not be without effect.'

3 Part 3

3.1 Exercise 11.3

Rewrite the CKY algorithm given in Fig. 11.5 on page 228 so that it can accept grammars that contain unit productions.

The CKY algorithm is expanded so when a symbol is added as a cell to the table, all symbols that

produced the original symbol are added.

[5]:
```
from cky import parser

grammar_file = "cky\grammar.txt"
grammar = parser.Grammar([line.rstrip('\n') for line in open(grammar_file)])

sentence = "the beautiful tropical fish"


parser.parse(sentence, grammar)
```

[5]:
```
[[['ART'], [], [], ['S']],
 [[], ['ADJ'], [], ['X']],
 [[], [], ['ADJ'], ['NP']],
 [[], [], [], ['NOUN']]]
```

3.2 Exercise 11.4

Discuss the relative advantages and disadvantages of partial versus full parsing

Partial parsing is faster than full parsing but provides lower quality of syntactical details. When a great amount of syntactic details is needed - a full parser performes better but requires more time.

3.3 Exercise 11.5

Discuss how to augment a parser to deal with input that may be incorrect, for example, containing spelling errors or mistakes arising from automatic speech recognition.

Take partial syntactic structures that the parser can identify and combine them together. These new combined parses would introduce new rules.

4 Part 4

Look up Latent Dirichlet Allocation (LDA), a method to define and discover topics in text, which is a generalization of sentence parsing, and find a Python package or program that implements it. Choose a short document (like a research paper or blog) to use as a training documents and two test documents, one with topics like the training document and one rather different. Run the results of the training documents and determine if they correctly identify the test documents as being similar or not. Interpret the result. It is possible this may not be able to be done in the rest of the semester; if so, describe what you learned about LDA and what you managed to do.

[3]:
```
from tika import parser

training = parser.from_file("data\Stevens Drone Detection Acoustic System.
 ↪pdf")["content"]
test1 = parser.from_file("data\Long-term testing of acoustic system for␣
 ↪tracking.pdf")["content"]
```

```
test2 = parser.from_file("data\A Playboy Interview With Miles Davis.
 ↪pdf")["content"]
```

[4]:
```
from newspaper import Article

article = Article("https://www.bbc.com/news/election-us-2020-55318269")
article.download()
article.parse()

training = article.text

article = Article("https://www.aljazeera.com/news/2020/12/15/
 ↪putin-finally-congratulates-biden-on-winning-us-presidency")
article.download()
article.parse()

test1 = article.text

article = Article("https://www.foxnews.com/us/
 ↪noreaster-suspends-outdoor-dining-new-york-city-after-snow-alert-issued")
article.download()
article.parse()

test2 = article.text
```

[5]:
```
from sklearn.feature_extraction.text import CountVectorizer

countVectorizer = CountVectorizer(stop_words="english")
termFrequency = countVectorizer.fit_transform([training])
featureNames = countVectorizer.get_feature_names()
```

[6]:
```
from sklearn.decomposition import LatentDirichletAllocation

lda = LatentDirichletAllocation(n_components=5)
lda.fit(termFrequency)
```

[6]:
```
LatentDirichletAllocation(batch_size=128, doc_topic_prior=None,
                          evaluate_every=-1, learning_decay=0.7,
                          learning_method='batch', learning_offset=10.0,
                          max_doc_update_iter=100, max_iter=10,
                          mean_change_tol=0.001, n_components=5, n_jobs=None,
                          perp_tol=0.1, random_state=None,
                          topic_word_prior=None, total_samples=1000000.0,
                          verbose=0)
```

[7]:
```

```

```
for idx, topic in enumerate(lda.components_):
    print ("Topic ", idx, " ".join(featureNames[i] for i in topic.argsort()[:
↪-10 - 1:-1]))
```

```
Topic  0 working kind america approach arms biden bracing control develop
differences
Topic  1 russia working kind america approach arms biden bracing control develop
Topic  2 working kind america approach arms biden bracing control develop
differences
Topic  3 working kind america approach arms biden bracing control develop
differences
Topic  4 working kind america approach arms biden bracing control develop
differences
```

Comparing the two similar articles shows a high similarity for Topic 1 (Russia)

[8]:
```
lda.transform(countVectorizer.transform([test1]))
```

[8]:
```
array([[0.00953391, 0.96186435, 0.00953391, 0.00953391, 0.00953391]])
```

Comparing the different topic article shows that they are not similar

[9]:
```
lda.transform(countVectorizer.transform([test2]))
```

[9]:
```
array([[0.2, 0.2, 0.2, 0.2, 0.2]])
```

5 Part 5

J&M takes an academic approach to NLP involving theory, history, and various approaches with examples. Recently books and software have become available that allows users to proceed without such extensive knowledge by using Python packages whose authors have made many choices for the user, in effect claiming these are the best. Two of these (by now there are likely many more, A&S might be one of them) taking this practical approach, as opposed to what they call the "bloated academic approach", are listed below

Thushan Ganegedara (2018) Natural Language Processing with TensorFlow, Packt Publishing Ltd., ISBN 978-1-78847-831-1

Read Chapters 1-3, run some of the suggested code from Chapter 3, briefly explain what it is for, and describe and interpret its results.

Ganegadara's "Natural Language Processing with TensorFlow" provides great insight to topics discussed in this course and how to approach them using Python's powerful TensorFlow package. The benefit of TensorFlow is its ability to use the powerful GPU for processing.

Chapter 3 of the book focuses on a nueral networks, a prominent feature of TensorFlow, for word representation with Word2Vec. Ganegadara demonstrates the use of TensorFlow for the skip-gram and Continuous Bag-of-Words algorithm.

My only problem with this book is that it features incomplete and error filled code. Below, is the first code example in the chapter that then gets referenced in further sections. The code requires a

function “get_common_and_rare_word_ids” however no where else in the book is this function defined, explained, or referenced. It is a little unbelievable that a paid book lacks this oversight.

[3]:
```
import tensorflow as tf

batch_size = 128
embedding_size = 128 # Dimension of the embedding vector.
window_size = 4 # How many words to consider left and right.
valid_size = 16 # Random set of words to evaluate similarity on.
# Only pick dev samples in the head of the distribution.
valid_window = 100
valid_examples = get_common_and_rare_word_ids(valid_size//2,valid_size//2)
num_sampled = 32 # Number of negative examples to sample.

train_dataset = tf.placeholder(tf.int32, shape=[batch_size])
train_labels = tf.placeholder(tf.int32, shape=[batch_size, 1])
valid_dataset = tf.constant(valid_examples, dtype=tf.int32)

embeddings = tf.Variable(
tf.random_uniform([vocabulary_size, embedding_size], -1.0, 1.0))
softmax_weights = tf.Variable(tf.truncated_normal([vocabulary_size,␣
 ↪embedding_size], stddev=0.5 / math.sqrt(embedding_size)))
softmax_biases = tf.Variable(tf.random_uniform([vocabulary_size],0.0,0.01))

embed = tf.nn.embedding_lookup(embeddings, train_dataset)

loss = tf.reduce_mean(tf.nn.sampled_softmax_loss(weights=softmax_weights,␣
 ↪biases=softmax_biases, inputs=embed, labels=train_labels,␣
 ↪num_sampled=num_sampled,num_classes=vocabulary_size))

optimizer = tf.train.AdagradOptimizer(1.0).minimize(loss)

norm = tf.sqrt(tf.reduce_sum(tf.square(embeddings), 1, keepdims=True))
normalized_embeddings = embeddings / norm
valid_embeddings = tf.nn.embedding_lookup(normalized_embeddings, valid_dataset)
similarity = tf.matmul(valid_embeddings, tf.transpose(normalized_embeddings))
```

```
      ␣
↪---------------------------------------------------------------------------

      NameError                                 Traceback (most recent call␣
↪last)

      <ipython-input-3-7e913081ffea> in <module>
        7 # Only pick dev samples in the head of the distribution.
        8 valid_window = 100
```

```
   ----> 9 valid_examples = get_common_and_rare_word_ids(valid_size//
↪2,valid_size//2)
        10 num_sampled = 32 # Number of negative examples to sample.
        11


       NameError: name 'get_common_and_rare_word_ids' is not defined
```