In [1]:

```
# Common imports
import pandas as pd
from IPython.display import Markdown, display, clear_output
```

In [2]:

```
import _pickle as cPickle
from pathlib import Path

def dumpPickle(fileName, content):
    pickleFile = open(fileName, 'wb')
    cPickle.dump(content, pickleFile, -1)
    pickleFile.close()

def loadPickle(fileName):
    file = open(fileName, 'rb')
    content = cPickle.load(file)
    file.close()

    return content

def pickleExists(fileName):
    file = Path(fileName)

    if file.is_file():
        return True

    return False
```

In [3]:

```
import spacy
from spacy import displacy
nlp = spacy.load('en_core_web_sm')

#Extract answers and the sentence they are in
def extractAnswers(qas, doc):
    answers = []

    senStart = 0
    senId = 0

    for sentence in doc.sents:
        senLen = len(sentence.text)

        for answer in qas:
            answerStart = answer['answers'][0]['answer_start']

            if (answerStart >= senStart and answerStart < (senStart + senLen)):
                answers.append({'sentenceId': senId, 'text': answer['answers'][0
]['text']})

        senStart += senLen
        senId += 1

    return answers

#TODO - Clean answers from stopwords?
def tokenIsAnswer(token, sentenceId, answers):
    for i in range(len(answers)):
        if (answers[i]['sentenceId'] == sentenceId):
            if (answers[i]['text'] == token):
                return True
    return False

#Save named entities start points

def getNEStartIndexs(doc):
    neStarts = {}
    for ne in doc.ents:
        neStarts[ne.start] = ne

    return neStarts

def getSentenceStartIndexes(doc):
    senStarts = []

    for sentence in doc.sents:
        senStarts.append(sentence[0].i)

    return senStarts

def getSentenceForWordPosition(wordPos, senStarts):
    for i in range(1, len(senStarts)):
        if (wordPos < senStarts[i]):
            return i - 1

def addWordsForParagrapgh(newWords, text):
    doc = nlp(text)
```

```
    neStarts = getNEStartIndexs(doc)
    senStarts = getSentenceStartIndexes(doc)

    #index of word in spacy doc text
    i = 0

    while (i < len(doc)):
        #If the token is a start of a Named Entity, add it and push to index to
 end of the NE
        if (i in neStarts):
            word = neStarts[i]
            #add word
            currentSentence = getSentenceForWordPosition(word.start, senStarts)
            wordLen = word.end - word.start
            shape = ''
            for wordIndex in range(word.start, word.end):
                shape += (' ' + doc[wordIndex].shape_)

            newWords.append([word.text,
                             0,
                             0,
                             currentSentence,
                             wordLen,
                             word.label_,
                             None,
                             None,
                             None,
                             shape])
            i = neStarts[i].end - 1
        #If not a NE, add the word if it's not a stopword or a non-alpha (not re
gular letters)
        else:
            if (doc[i].is_stop == False and doc[i].is_alpha == True):
                word = doc[i]

                currentSentence = getSentenceForWordPosition(i, senStarts)
                wordLen = 1

                newWords.append([word.text,
                                 0,
                                 0,
                                 currentSentence,
                                 wordLen,
                                 None,
                                 word.pos_,
                                 word.tag_,
                                 word.dep_,
                                 word.shape_])
        i += 1

def oneHotEncodeColumns(df):
    columnsToEncode = ['NER', 'POS', "TAG", 'DEP']

    for column in columnsToEncode:
        one_hot = pd.get_dummies(df[column])
        one_hot = one_hot.add_prefix(column + '_')

        df = df.drop(column, axis = 1)
        df = df.join(one_hot)

    return df
```

```
/home/aayusi/anaconda3/lib/python3.7/site-packages/requests/__init_
_.py:91: RequestsDependencyWarning: urllib3 (1.25.10) or chardet (3.
0.4) doesn't match a supported version!
  RequestsDependencyWarning)
```

In [4]:

```
def generateDf(text):
    words = []
    addWordsForParagrapgh(words, text)

    wordColums = ['text', 'titleId', 'paragrapghId', 'sentenceId','wordCount',
'NER', 'POS', 'TAG', 'DEP','shape']
    df = pd.DataFrame(words, columns=wordColums)

    return df
```

In [5]:

```
def prepareDf(df):
    #One-hot encoding
    wordsDf = oneHotEncodeColumns(df)

    #Drop unused columns
    columnsToDrop = ['text', 'titleId', 'paragrapghId', 'sentenceId', 'shape']
    wordsDf = wordsDf.drop(columnsToDrop, axis = 1)

    #Add missing colums
    predictorColumns = ['wordCount','NER_CARDINAL','NER_DATE','NER_EVENT','NER_F
AC','NER_GPE','NER_LANGUAGE','NER_LAW','NER_LOC','NER_MONEY','NER_NORP','NER_ORD
INAL','NER_ORG','NER_PERCENT','NER_PERSON','NER_PRODUCT','NER_QUANTITY','NER_TIM
E','NER_WORK_OF_ART','POS_ADJ','POS_ADP','POS_ADV','POS_CCONJ','POS_DET','POS_IN
TJ','POS_NOUN','POS_NUM','POS_PART','POS_PRON','POS_PROPN','POS_PUNCT','POS_SYM'
,'POS_VERB','POS_X','TAG_''','TAG_-LRB-','TAG_.','TAG_ADD','TAG_AFX','TAG_CC','T
AG_CD','TAG_DT','TAG_EX','TAG_FW','TAG_IN','TAG_JJ','TAG_JJR','TAG_JJS','TAG_LS'
,'TAG_MD','TAG_NFP','TAG_NN','TAG_NNP','TAG_NNPS','TAG_NNS','TAG_PDT','TAG_POS',
'TAG_PRP','TAG_PRP$','TAG_RB','TAG_RBR','TAG_RBS','TAG_RP','TAG_SYM','TAG_TO','T
AG_UH','TAG_VB','TAG_VBD','TAG_VBG','TAG_VBN','TAG_VBP','TAG_VBZ','TAG_WDT','TAG
_WP','TAG_WRB','TAG_XX','DEP_ROOT','DEP_acl','DEP_acomp','DEP_advcl','DEP_advmo
d','DEP_agent','DEP_amod','DEP_appos','DEP_attr','DEP_aux','DEP_auxpass','DEP_ca
se','DEP_cc','DEP_ccomp','DEP_compound','DEP_conj','DEP_csubj','DEP_csubjpass',
'DEP_dative','DEP_dep','DEP_det','DEP_dobj','DEP_expl','DEP_intj','DEP_mark','DE
P_meta','DEP_neg','DEP_nmod','DEP_npadvmod','DEP_nsubj','DEP_nsubjpass','DEP_num
mod','DEP_oprd','DEP_parataxis','DEP_pcomp','DEP_pobj','DEP_poss','DEP_preconj',
'DEP_predet','DEP_prep','DEP_prt','DEP_punct','DEP_quantmod','DEP_relcl','DEP_xc
omp']

    for feature in predictorColumns:
        if feature not in wordsDf.columns:
            wordsDf[feature] = 0

    return wordsDf
```

In [16]:

```
def predictWords(wordsDf, df):

    predictorPickleName = 'nb-predictor.pkl'
    predictor = loadPickle(predictorPickleName)

    y_pred = predictor.predict_proba(wordsDf)

    labeledAnswers = []
    for i in range(len(y_pred)):
        labeledAnswers.append({'word': df.iloc[i]['text'], 'prob': y_pred[i][0
]})

    return labeledAnswers
```

In [17]:

```
def blankAnswer(firstTokenIndex, lastTokenIndex, sentStart, sentEnd, doc):
    leftPartStart = doc[sentStart].idx
    leftPartEnd = doc[firstTokenIndex].idx
    rightPartStart = doc[lastTokenIndex].idx + len(doc[lastTokenIndex])
    rightPartEnd = doc[sentEnd - 1].idx + len(doc[sentEnd - 1])

    question = doc.text[leftPartStart:leftPartEnd] + '_____' + doc.text[rightPar
tStart:rightPartEnd]

    return question
```

In [18]:

```
def addQuestions(answers, text):
    doc = nlp(text)
    currAnswerIndex = 0
    qaPair = []

    #Check wheter each token is the next answer
    for sent in doc.sents:
        for token in sent:

            #If all the answers have been found, stop looking
            if currAnswerIndex >= len(answers):
                break

            #In the case where the answer is consisted of more than one token, c
heck the following tokens as well.
            answerDoc = nlp(answers[currAnswerIndex]['word'])
            answerIsFound = True

            for j in range(len(answerDoc)):
                if token.i + j >= len(doc) or doc[token.i + j].text != answerDoc
[j].text:
                    answerIsFound = False

            #If the current token is corresponding with the answer, add it
            if answerIsFound:
                question = blankAnswer(token.i, token.i + len(answerDoc) - 1, se
nt.start, sent.end, doc)

                qaPair.append({'question' : question, 'answer': answers[currAnsw
erIndex]['word'], 'prob': answers[currAnswerIndex]['prob']})

                currAnswerIndex += 1

    return qaPair
```

In [19]:

```
def sortAnswers(qaPairs):
    orderedQaPairs = sorted(qaPairs, key=lambda qaPair: qaPair['prob'])

    return orderedQaPairs
```

In [20]:

```
import gensim
from gensim.test.utils import datapath, get_tmpfile
from gensim.models import KeyedVectors

glove_file = 'glove.6B.300d.txt'
tmp_file = 'word2vec-glove.6B.300d.txt'

from gensim.scripts.glove2word2vec import glove2word2vec
glove2word2vec(glove_file, tmp_file)
model = KeyedVectors.load_word2vec_format(tmp_file)
```

In [24]:

```
def generate_distractors(answer, count):
    answer = str.lower(answer)

    ##Extracting closest words for the answer.
    try:
        closestWords = model.most_similar(positive=[answer], topn=count)
    except:
        #In case the word is not in the vocabulary, or other problem not loading
 embeddings
        return []

    #Return count many distractors
    distractors = list(map(lambda x: x[0], closestWords))[0:count]

    return distractors
```

In [25]:

```
def addDistractors(qaPairs, count):
    for qaPair in qaPairs:
        distractors = generate_distractors(qaPair['answer'], count)
        qaPair['distractors'] = distractors

    return qaPairs
```

In [26]:

```
def generateQuestions(text, count):

    # Extract words
    df = generateDf(text)
    wordsDf = prepareDf(df)

    # Predict
    labeledAnswers = predictWords(wordsDf, df)

    # Transform questions
    qaPairs = addQuestions(labeledAnswers, text)

    # Pick the best questions
    orderedQaPairs = sortAnswers(qaPairs)

    # Generate distractors
    questions = addDistractors(orderedQaPairs[:count], 4)

    # Print
    for i in range(count):
        display(Markdown('### Question ' + str(i + 1) + ':'))
        print(questions[i]['question'])

        display(Markdown('#### Answer:'))
        print(questions[i]['answer'])

        display(Markdown('#### Incorrect answers:'))
        for distractor in questions[i]['distractors']:
            print(distractor)

        print()
```

In [55]:

```
text1 = "Bansoori is an Indian classical instrument. Akhil plays Bansoori and Gu
itar. Puliyogare is a South Indian dish made of rice and tamarind. Priya writes
 poems. Osmosis is the movement of a solvent across a semipermeable membrane tow
ard a higher concentration of solute. In biological systems, the solvent is typi
cally water, but osmosis can occur in other liquids, supercritical liquids, and
 even gases. I like Bansoori. When a cell is submerged in water, the water molec
ules pass through the cell membrane from an area of low solute concentration to
 high solute concentration. For example, if the cell is submerged in saltwater,
 water molecules move out of the cell. If a cell is submerged in freshwater, wat
er molecules move into the cell. Raja-Yoga is divided into eight steps, the firs
t is Yama non killing, truthfulness, non stealing, continence, and non receiving
 of any gifts."
```

In [62]:

```
generateQuestions(text1, 10)
```

```
/home/aayusi/anaconda3/lib/python3.7/site-packages/sklearn/base.py:3
34: UserWarning: Trying to unpickle estimator GaussianNB from versio
n 0.20.3 when using version 0.23.2. This might lead to breaking code
or invalid results. Use at your own risk.
  UserWarning)
```

Question 1:

It was July 21, 1969, and Neil Armstrong _____ with a start.

Answer:

awoke

Incorrect answers:

woke
awakened
awakens
awaken

Question 2:

The journey had begun several days earlier, when on July 16th, the A
pollo 11 _____ from Earth headed into outer space.

Answer:

launched

Incorrect answers:

launch
launching
launches
initiated

Question 3:

The journey had begun several days earlier, when on July 16th, the A
pollo 11 launched from Earth _____ into outer space.

Answer:

headed

Incorrect answers:

heading
heads
head
arrived

Question 4:

Take out all of the trash, and _____ all of the dirty dishes in the kitchen sink.

Answer:

place

Incorrect answers:

places
time
.
where

Question 5:

For any remaining items, see if you can squeeze them in under your b ed or _____ them into the back of your closet.

Answer:

stuff

Incorrect answers:

things
thing
really
guys

Question 6:

Oceans and _____ have much in common, but they are also quite differ ent.

Answer:

lakes

Incorrect answers:

lake
rivers
ponds
streams

Question 7:

Both have plants and _____ living in them.

Answer:

animals

Incorrect answers:

animal
humans
birds
mammals

Question 8:

When it is time for a vacation, both will make a great place to visi
t and _____.

Answer:

enjoy

Incorrect answers:

enjoying
enjoyed
enjoys
want

Question 9:

The journey had begun several days earlier, when on July 16th, the _
____ 11 launched from Earth headed into outer space.

Answer:

Apollo

Incorrect answers:

spacecraft
gemini
astronauts
moon

Question 10:

_____ are usually surrounded by land, while oceans are what surround
continents.

Answer:

Lakes

Incorrect answers:

lake
rivers
ponds
streams

In [60]:

```
f = open("file1.txt", 'r')
text1 = f.read()
```

In [61]:

```
generateQuestions(text1, 10)
```

```
/home/aayusi/anaconda3/lib/python3.7/site-packages/sklearn/base.py:3
34: UserWarning: Trying to unpickle estimator GaussianNB from versio
n 0.20.3 when using version 0.23.2. This might lead to breaking code
or invalid results. Use at your own risk.
  UserWarning)
```

Question 1:

It was July 21, 1969, and Neil Armstrong _____ with a start.

Answer:

awoke

Incorrect answers:

woke
awakened
awakens
awaken

Question 2:

The journey had begun several days earlier, when on July 16th, the A
pollo 11 _____ from Earth headed into outer space.

Answer:

launched

Incorrect answers:

launch
launching
launches
initiated

Question 3:

The journey had begun several days earlier, when on July 16th, the A
pollo 11 launched from Earth _____ into outer space.

Answer:

headed

Incorrect answers:

heading
heads
head
arrived

Question 4:

Take out all of the trash, and _____ all of the dirty dishes in the kitchen sink.

Answer:

place

Incorrect answers:

places
time
.
where

Question 5:

For any remaining items, see if you can squeeze them in under your b ed or _____ them into the back of your closet.

Answer:

stuff

Incorrect answers:

things
thing
really
guys

Question 6:

Oceans and _____ have much in common, but they are also quite differ ent.

Answer:

lakes

Incorrect answers:

lake
rivers
ponds
streams

Question 7:

Both have plants and _____ living in them.

Answer:

animals

Incorrect answers:

animal
humans
birds
mammals

Question 8:

When it is time for a vacation, both will make a great place to visi
t and _____.

Answer:

enjoy

Incorrect answers:

enjoying
enjoyed
enjoys
want

Question 9:

The journey had begun several days earlier, when on July 16th, the _
____ 11 launched from Earth headed into outer space.

Answer:

Apollo

Incorrect answers:

spacecraft
gemini
astronauts
moon

Question 10:

_____ are usually surrounded by land, while oceans are what surround
continents.

Answer:

Lakes

Incorrect answers:

lake
rivers
ponds
streams

In [ ]:

```

```

In [ ]:

```

```