CentraleSupélec

COVAI CHALLENGE

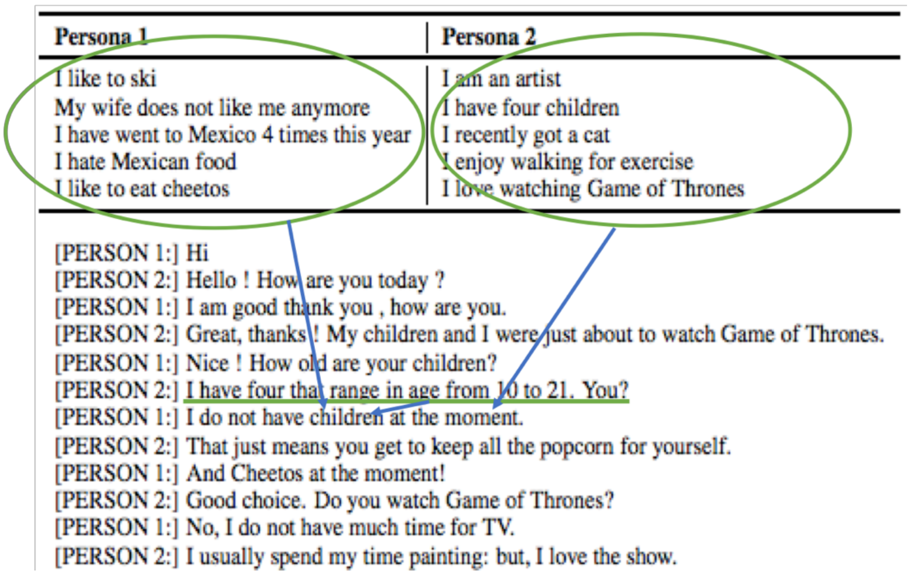
Natural Language Processing-Homework 3

April the 19th 2019

ATTYASSE Flora | BENICHOU Vadim | MICCICHE Carmelo | VIAL Jennifer

Presentation of the project

This project aims at programming an algorithm in Python that is able to build a consistent Chitchat. For any given question, it may pick the answer within a list of sentences that fits the best with a context based on an utterance and on people's characteristics. This program is elaborated through cosine similarity. This kind of algorithm is a pretty useful to ensure chatbots performance. This report sum-up the creation process of the algorithm, the choices made for the programming part, the obstacles faced, the final results and the sources.



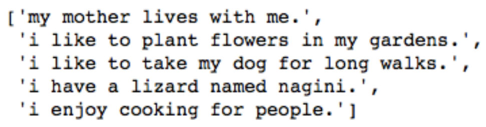
Persona 1	Persona 2
I like to ski	I am an artist
My wife does not like me anymore	I have four children
I have went to Mexico 4 times this year	I recently got a cat
I hate Mexican food	I enjoy walking for exercise
I like to eat cheetos	I love watching Game of Thrones

[PERSON 1:] Hi
[PERSON 2:] Hello ! How are you today ?
[PERSON 1:] I am good thank you , how are you.
[PERSON 2:] Great, thanks ! My children and I were just about to watch Game of Thrones.
[PERSON 1:] Nice ! How old are your children?
[PERSON 2:] I have four that range in age from 10 to 21. You?
[PERSON 1:] I do not have children at the moment.
[PERSON 2:] That just means you get to keep all the popcorn for yourself.
[PERSON 1:] And Cheetos at the moment!
[PERSON 2:] Good choice. Do you watch Game of Thrones?
[PERSON 1:] No, I do not have much time for TV.
[PERSON 2:] I usually spend my time painting: but, I love the show.

1/The persona information and the previous utterances to predict the answer
Source: Personalizing Dialogue Agents: I have a dog, do you have a pet by Zhang, Dinan, Urbanek, Szlam, Kiela, Weston

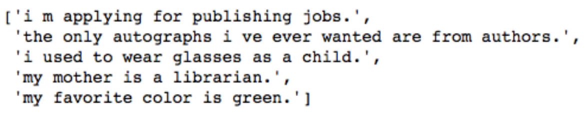
Data provided

The model is elaborated from a text that contains several dialogues and personas. Two kinds of personas are to be distinguished: Your personas that represent the characteristics about the person who is supposed to answer and the partner personas that correspond to the criteria of the person who asks the questions. Each persona contains about four information.



```
['my mother lives with me.',
 'i like to plant flowers in my gardens.',
 'i like to take my dog for long walks.',
 'i have a lizard named nagini.',
 'i enjoy cooking for people.']
```

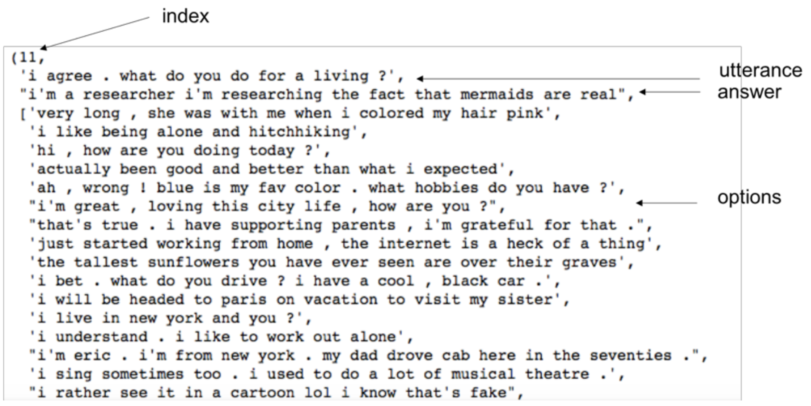
2/Your persona example



```
['i m applying for publishing jobs.',
 'the only autographs i ve ever wanted are from authors.',
 'i used to wear glasses as a child.',
 'my mother is a librarian.',
 'my favorite color is green.']
```

3/Partner persona example

Each dialogue is composed of an index, an utterance and several options. The correct answer is also indicated in the correct set in order to evaluate correctly the accuracy of the model.



```
(11,
 'i agree . what do you do for a living ?',
 "i'm a researcher i'm researching the fact that mermaids are real",
 ['very long , she was with me when i colored my hair pink',
  'i like being alone and hitchhiking',
  'hi , how are you doing today ?',
  'actually been good and better than what i expected',
  'ah , wrong ! blue is my fav color . what hobbies do you have ?',
  "i'm great , loving this city life , how are you ?",
  "that's true . i have supporting parents , i'm grateful for that .",
  'just started working from home , the internet is a heck of a thing',
  'the tallest sunflowers you have ever seen are over their graves',
  'i bet . what do you drive ? i have a cool , black car .',
  'i will be headed to paris on vacation to visit my sister',
  'i live in new york and you ?',
  'i understand . i like to work out alone',
  "i'm eric . i'm from new york . my dad drove cab here in the seventies .",
  'i sing sometimes too . i used to do a lot of musical theatre .',
  "i rather see it in a cartoon lol i know that's fake",
```

4/Dialogue example

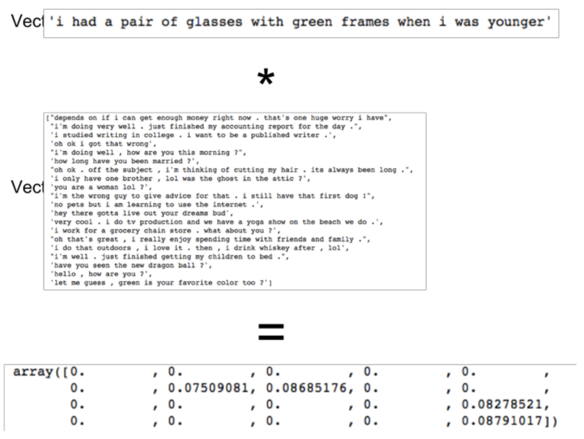
Data Management

A generator is created to load correctly the text. For any given index, a dialogue contained into a list of sentences is generated. Thus, one can extract also the options, the utterance, the answer and the personas descriptions associated.

The similarity computation

To assess which option fits the best with the context, the cosine similarity between each option, the personas characteristics and the utterance needs to be computed. Firstly, the str options must be converted into vectors, the vectorizer from the library scikitlearn is used to realized this task. Once the vector is created it is normalized in order to unit length for the similarity computations.

The similarity between the utterance and the option is computed by making the dot product between the normalized option vector and the normalized utterance vector. This task is made for every option to get at the end a matrix with the similarity scores between the different options and the utterance.



```
'i had a pair of glasses with green frames when i was younger'
```

```
["depends on if i can get enough money right now . that's one huge worry i have",
 "i'm doing very well . just finished my accounting report for the day .",
 'i studied writing in college . i want to be a published writer .',
 'oh ok i got that wrong',
 "i'm doing well , how are you this morning ?",
 'how long have you been married ?',
 "oh ok . off the subject , i'm thinking of cutting my hair . its always been long .",
 'i only have one brother , lol was the ghost in the attic ?',
 'you are a woman lol ?',
 "i'm the wrong guy to give advice for that . i still have that first dog !",
 'no pets but i am learning to use the internet .',
 'hey there gotta live out your dreams bud',
 'very cool . i do tv production and we have a yoga show on the beach we do .',
 'i work for a grocery chain store . what about you ?',
 "oh that's great , i really enjoy spending time with friends and family .",
 'i do that outdoors , i love it . then , i drink whiskey after , lol',
 "i'm well . just finished getting my children to bed .",
 'have you seen the new dragon ball ?',
 'hello , how are you ?',
 'let me guess , green is your favorite color too ?']
```

```
array([0.        , 0.        , 0.        , 0.        , 0.        ,
       0.        , 0.07509081, 0.08685176, 0.        , 0.        ,
       0.        , 0.        , 0.        , 0.        , 0.08278521,
       0.        , 0.        , 0.        , 0.        , 0.08791017])
```

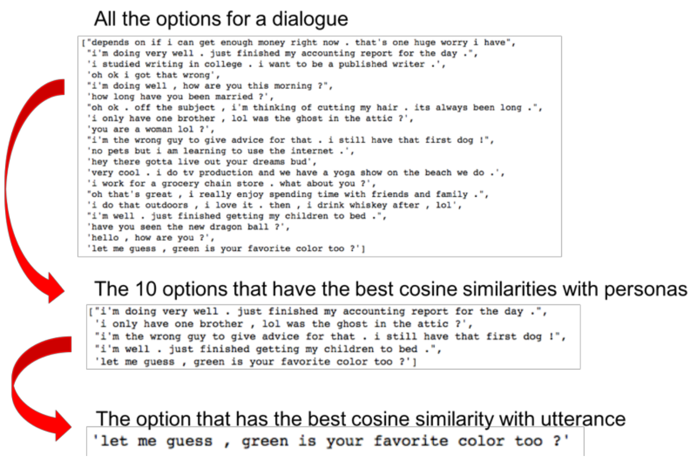
5/Visualization of similarity computation

The dot product is also computed between a given option and all the characteristics of a persona between the normalized option vector and the normalized utterance vector to get also a similarity matrix.

The Structure of our program

The program aims at finding the most likely answer for any given dialogue. The function findBest returns this latter. It is divided into two main steps:

- Firstly, it computes the similarity between the all persona characteristics (your persona, and partner) and it returns the 10 options with the best cosine similarity.
- Secondly, it keeps those 5 options and the cosine similarity with the utterance is computed. The best is then returned.



```
["depends on if i can get enough money right now . that's one huge worry i have",
 "i'm doing very well . just finished my accounting report for the day .",
 'i studied writing in college . i want to be a published writer .',
 'oh ok i got that wrong',
 "i'm doing well , how are you this morning ?",
 'how long have you been married ?',
 "oh ok . off the subject , i'm thinking of cutting my hair . its always been long .",
 'i only have one brother , lol was the ghost in the attic ?',
 'you are a woman lol ?',
 "i'm the wrong guy to give advice for that . i still have that first dog !",
 'no pets but i am learning to use the internet .',
 'hey there gotta live out your dreams bud',
 'very cool . i do tv production and we have a yoga show on the beach we do .',
 'i work for a grocery chain store . what about you ?',
 "oh that's great , i really enjoy spending time with friends and family .",
 'i do that outdoors , i love it . then , i drink whiskey after , lol',
 "i'm well . just finished getting my children to bed .",
 'have you seen the new dragon ball ?',
 'hello , how are you ?',
 'let me guess , green is your favorite color too ?']
```

The 10 options that have the best cosine similarities with personas

```
["i'm doing very well . just finished my accounting report for the day .",
 'i only have one brother , lol was the ghost in the attic ?',
 "i'm the wrong guy to give advice for that . i still have that first dog !",
 "i'm well . just finished getting my children to bed .",
 'let me guess , green is your favorite color too ?']
```

The option that has the best cosine similarity with utterance

```
'let me guess , green is your favorite color too ?'
```

6/Visualization of the findBest method

Other trials

We tried to vectorize the sentences through our own encoding and Bag of words algorithm based on crawl-300d-200k.vec document. We also coded two functions; a similarity algorithm and a most similar model that returns the k most similar sentences to any given string. Unfortunately, the results were not as satisfying as expected. (see the functions in the annexes)

We also tried a different findBest structure: the first step consisted in returning the 10 options that have the best cosine similarity with utterance, then the second step returned the option that has the best cosine similarity with personas.

The Results

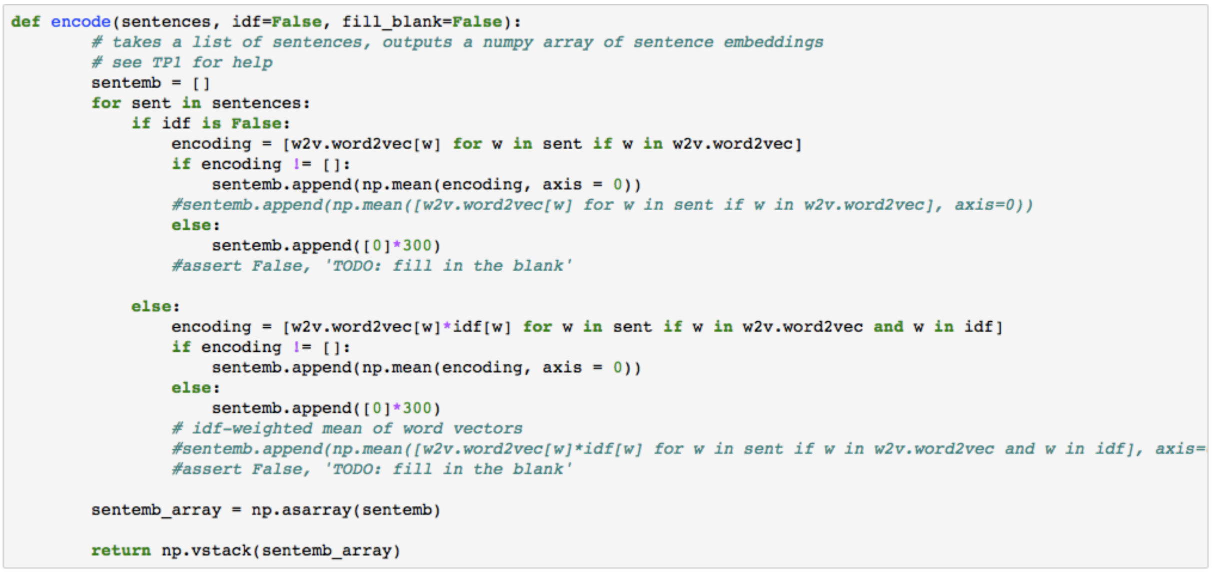
The accuracy with our own word embedding were not conclusive; indeed it was around 9%. When we used the the scickit learn library for the vectorizer the result was much better and went up to 30% with the findBest structure that takes the utterance cosine similarity as first step. By changing the findBest structure we reached 43% and by normalizing the vectors we finally get an accuracy of 47%.

Accuracy table	
Accuracy with our word embedding	9%
Accuracy with findBest structure 1	30%
Accuracy with findBest structure 2	43%
Accuracy with findBest structure 2+normalization	**47%**

The Sources

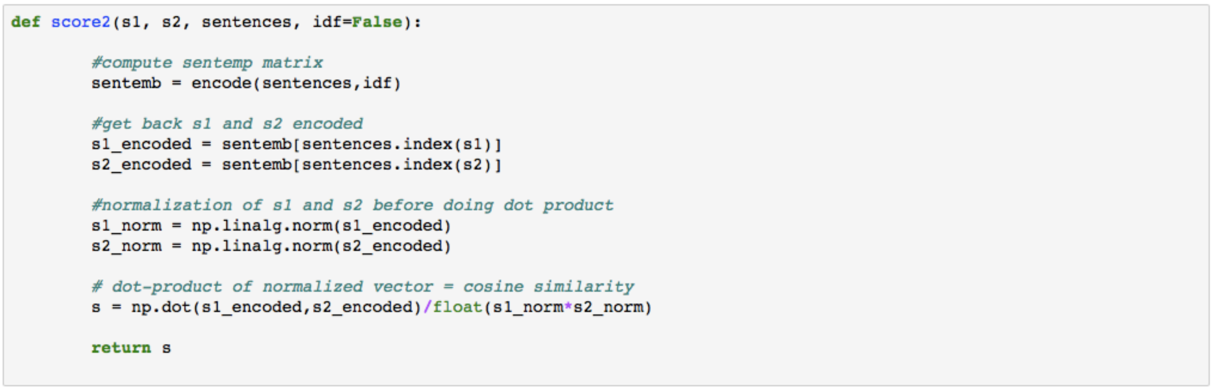
The work is based on *Personalizing Dialogue Agents: I have a dog, do you have a pet* by Zhang, Dinan, Urbanek, Szlam, Kiela, Weston. This paper also aims at finding the best method to get the most consistent answer to a chatbot sentence given personas. It introduces different approach to deal with it, generative and ranking based ones. Our approach is inspired by the ranking process and the way they computed similarity with personas. Furthermore, *An IR baseline* by Sordoni and supervised embedding model, Starspace by Wu were interesting tools to think over the findBest structure.

Annexes



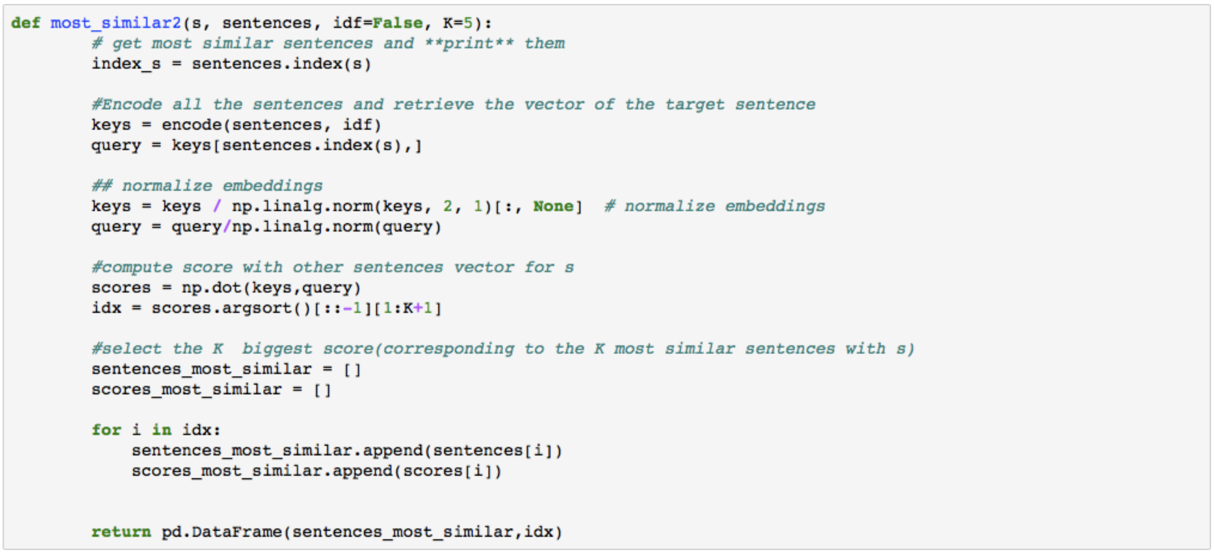
```
def encode(sentences, idf=False, fill_blank=False):
        # takes a list of sentences, outputs a numpy array of sentence embeddings
        # see TP1 for help
        sentemb = []
        for sent in sentences:
            if idf is False:
                encoding = [w2v.word2vec[w] for w in sent if w in w2v.word2vec]
                if encoding != []:
                    sentemb.append(np.mean(encoding, axis = 0))
                #sentemb.append(np.mean([w2v.word2vec[w] for w in sent if w in w2v.word2vec], axis=0))
                else:
                    sentemb.append([0]*300)
                #assert False, 'TODO: fill in the blank'

            else:
                encoding = [w2v.word2vec[w]*idf[w] for w in sent if w in w2v.word2vec and w in idf]
                if encoding != []:
                    sentemb.append(np.mean(encoding, axis = 0))
                else:
                    sentemb.append([0]*300)
                # idf-weighted mean of word vectors
                #sentemb.append(np.mean([w2v.word2vec[w]*idf[w] for w in sent if w in w2v.word2vec and w in idf], axis=
                #assert False, 'TODO: fill in the blank'

        sentemb_array = np.asarray(sentemb)

        return np.vstack(sentemb_array)
```

6/Encode function



```python
def score2(s1, s2, sentences, idf=False):

        #compute sentemp matrix
        sentemb = encode(sentences,idf)

        #get back s1 and s2 encoded
        s1_encoded = sentemb[sentences.index(s1)]
        s2_encoded = sentemb[sentences.index(s2)]

        #normalization of s1 and s2 before doing dot product
        s1_norm = np.linalg.norm(s1_encoded)
        s2_norm = np.linalg.norm(s2_encoded)

        # dot-product of normalized vector = cosine similarity
        s = np.dot(s1_encoded,s2_encoded)/float(s1_norm*s2_norm)

        return s
```

7/Score computation function



```python
def most_similar2(s, sentences, idf=False, K=5):
        # get most similar sentences and **print** them
        index_s = sentences.index(s)

        #Encode all the sentences and retrieve the vector of the target sentence
        keys = encode(sentences, idf)
        query = keys[sentences.index(s),]

        ## normalize embeddings
        keys = keys / np.linalg.norm(keys, 2, 1)[:, None]  # normalize embeddings
        query = query/np.linalg.norm(query)

        #compute score with other sentences vector for s
        scores = np.dot(keys,query)
        idx = scores.argsort()[::-1][1:K+1]

        #select the K  biggest score(corresponding to the K most similar sentences with s)
        sentences_most_similar = []
        scores_most_similar = []

        for i in idx:
            sentences_most_similar.append(sentences[i])
            scores_most_similar.append(scores[i])


        return pd.DataFrame(sentences_most_similar,idx)
```

8/K-most similar function