Content Based Recommender System using Python

One of the most surprising and fascinating applications of Machine Learning **recommender systems**. In this lesson we will look at Content Based recommender system.

In a nutshell, a recommender system is a tool that suggests you the next content given what you have already seen and liked. Companies like **Spotify, Netflix** or **Youtube** use recommender systems to suggest you the next video or song to watch given what you have already seen or listened to.

In this notebook we show you how to build a content based recommender system using few lines of code and some domain knowledge about **machine learning** and **algebra**.

Let’s dive in :)

0. The Libraries

This is what you’ll need to make it work:

```
!pip install sentence_transformers
import numpy as np
from sentence_transformers import SentenceTransformer
import pandas as pd
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.metrics.pairwise import cosine_similarity
```

1. The Dataset

The first thing we need is of course the **dataset**. I found the dataset here and it is basically a collection extracted from **IMDb** . In this collection we have a list of movies with their correspondent features. The features we are using are:

- An **overview** of the movie (i.e. a brief description)
- Its **title**
- Its **genre** (actually, each movie has multiple genres)

Let's give a look then:

1.1 Importing it:

```
data = pd.read_csv('imdb_top_1000.csv')

X = np.array(data.Overview)

data
```

1.2 Viewing it:

```
data = pd.read_csv('imdb_top_1000.csv')
X = np.array(data.Overview)
data
```

	Genre	Overview	Series_Title
0	Drama	Two imprisoned men bond over a number of years...	The Shawshank Redemption
1	Crime, Drama	An organized crime dynasty's aging patriarch t...	The Godfather
2	Action, Crime, Drama	When the menace known as the Joker wreaks havo...	The Dark Knight
3	Crime, Drama	The early life and career of Vito Corleone in ...	The Godfather: Part II

2. The Approach

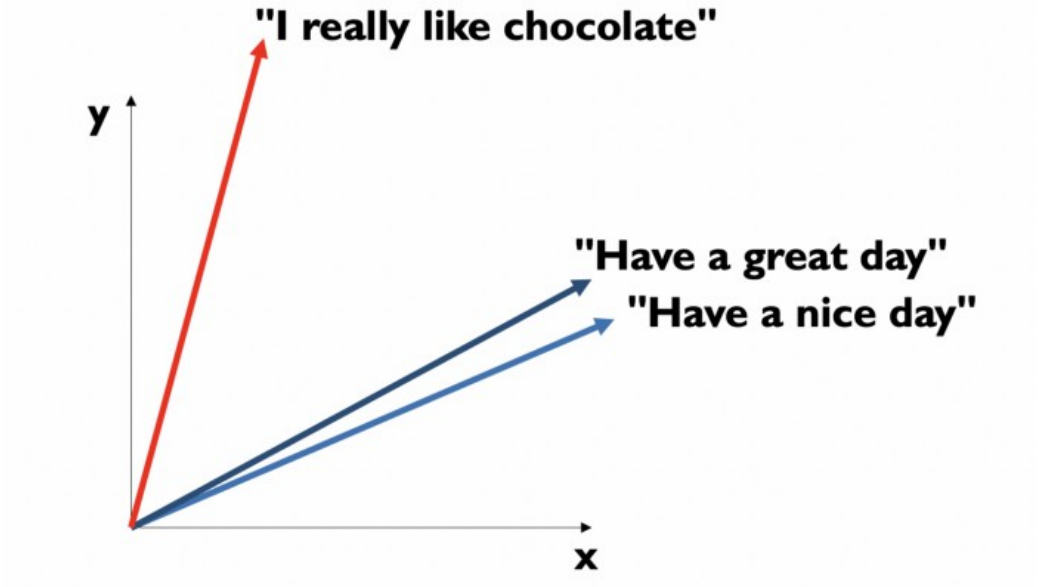
When working with textual data the first thing to do is to convert this text into **numbers**. More precisely we are converting a string of text into a **vector**.

But why do we want to convert texts into vectors in the first place?

Well, because now that we have **vectors** we are able to do all the operations that you usually do on vectors :)

Imagine you are doing a classification task and you want to use Support Vector Machines. Well, of course, you will need to have **vectors** to do that. So if you have to classify a text you will first convert the text into a vector and then apply the SVM algorithm.

In our case we will use the vectorize test to find the **similarity** between two vectors. Our recommendation will be the (5) **most similar vectors** with the one we are considering.



Let’s say x and y are two components (we’ll have way more than two). If a movie talks about **science, space and rockets** and another one talks more or less about the same stuff, we expect the two vectors to be close, like the blue ones. On the other hand, if the other movie is about a **love story** we will expect this vector to be far away, like the red and the blue ones.

So here is what we’ll do:

1. Use **BERT** to convert our text into a vector
2. Get the **cosine similarity** (the cosine of the angle between the two vectors) of a fixed movie (vector) and all the other ones
3. **Pick the movies (vectors) with the largest cosine similarity.** We are going to pick 5 of them.

3. The method

3.1 From text to vector:

Here’s how you convert a **text** to a **vector**

```
text_data = X
model = SentenceTransformer('distilbert-base-nli-mean-tokens')
embeddings = model.encode(text_data, show_progress_bar=True)
# https://huggingface.co/sentence-transformers
```

And when we print the embeddings, we get a list of 768 vectors per item

```
cos_sim_data = pd.DataFrame(embeddings)
cos_sim_data
```

	0	1	2	3	...	768	
0	-	-	1.367879	-0.779558	-	0.2311	-0.606564

	0	1	2	3	...	768	
	0.082307	0.454635			0.604250	30	
1	-1.020697	-1.051922	0.163983	-1.064686	-0.841295	0.063894	-0.731421
2	-0.909217	-0.519028	0.496283	-0.862778	-0.459598	-0.079323	-0.656419
3	-0.553092	-0.606065	-0.263538	-2.065412	-0.200945	0.303858	-0.427347
4	-0.211580	-0.223869	0.814861	-0.198402	-0.256405	0.210274	-0.757677

3.3 Cosine Similarity and Recommendation Function

Here is how to compute the **cosine similarity** (one line of code) and **the function** that we'll use to get our recommendations:

```
cos_sim_data = pd.DataFrame(cosine_similarity(embeddings))
cos_sim_data
def give_recommendations(index):
index_recomm =cos_sim_data.loc[index].index.tolist()[1:5]
movies_recomm = data['Series_Title'].loc[index_recomm].values
result = {'Movies':movies_recomm,'Index':index_recomm}
print('The watched movie is this one: %s \n'(data['Series_Title'].loc[index]))
k=1
for movie in movies_recomm:
print('The number %i recommended movie is this one: %s \n'%(k,movie))

return result
```

Lets call above method person the movie id

give_recommendations(2)

The algorithm recognize that **The Godfather, the Godfather II and the Godfather III are similar!**

Let’s **explore** it deeper.

The watched movie is this one: The Godfather

The number 1 recommended movie is this one: The Godfather

The number 1 recommended movie is this one: The Dark Knight

The number 1 recommended movie is this one: The Godfather: Part II

The number 1 recommended movie is this one: 12 Angry Men

{'Index': [1, 2, 3, 4],
 'Movies': array(['The Godfather', 'The Dark Knight', 'The Godfather: Part II',
       '12 Angry Men'], dtype=object)}

Useful Links

https://huggingface.co/sentence-transformers

https://www.sciencedirect.com/topics/computer-science/cosine-similarity

https://www.machinelearningplus.com/nlp/cosine-similarity/