

Linear Algebra Course

Music recommendations

Using word embedding

Vladyslav Bilyk

Nazar Pasternak

Nazar Todoschchuk

Ukrainian Catholic University

Spring 2020

1. Introduction

Music, and listening to it has become one of the every-day entertaining routines that most people enjoy throughout the day. Especially nowadays, with strong quarantine restrictions, many music lovers are facing the issue of finding new music that would be relevant for them and which they would like to listen to. The available music services' recommendation algorithms are targeted at what "most people like" and usually recommend music irrelevant especially to you. So we decided to create our own recommendation system, which predicts the songs for the user given his/her favourite artists and their songs respectively. To solve the problem of music recommendation, we looked at some already implemented approaches. In general, there are 3 different types of recommendation systems:

- Content-based

- Collaborative

- Popularity-based

In collaborative type the item itself, or its features, that is being recommended is not being analyzed. The assumption is made that previous information in a user's history about how they agree with other users, will be predictive in determining whether or not they will enjoy a certain item.

Content-based filtering closely examines the actual item to determine which features are most important in making recommendations and how those features interact with the user's preferences. Data collection can be complicated in content-based filtering as it is complicated to select which features of an item will be important in creating some sort of predictive model.

Lastly, popularity-based systems emphasize on the popularity of a certain item, as the name suggests.

We chose to implement a content-based approach and, since the data collection is quite complicated in this approach, we will be using Word2Vec technique to find the best features of the song to match the songs from initial dataset and result in the best possible match.

2. Data description

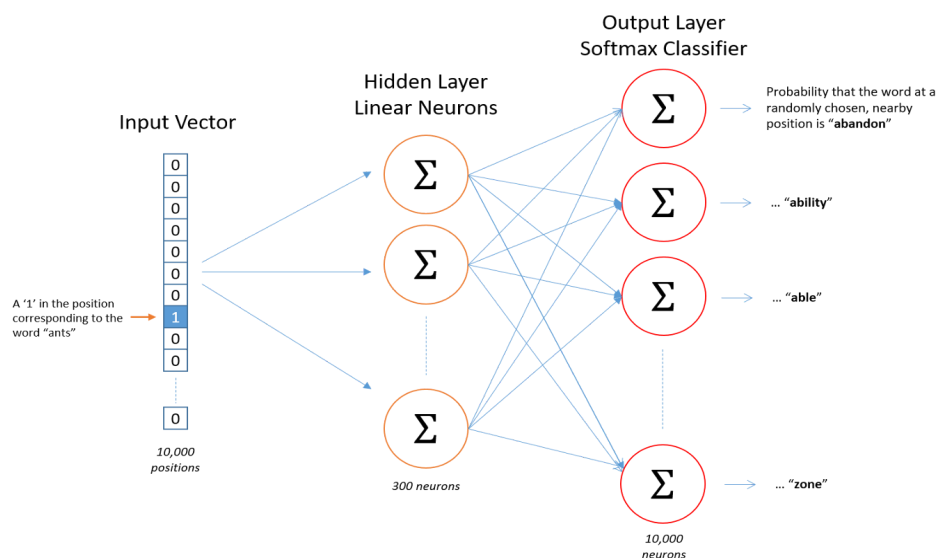
For our purposes, we searched for a huge dataset consisting of data pieces like the author, name of the song and lyrics and found one that perfectly meets our needs on the web on kaggle.com. The dataset consists of over 50,000 songs and their lyrics. For the analysis to be maximally relevant, we cleared up the song lyrics. To be more precise, we deleted the stop words like: then, here, there, his, I'm, yours etc. Also, we deleted all punctuation marks, lemmatized all the words, so that making → make, books → book. Finally, we deleted all songs, which lyrics contain less than 50% words in English.

3. Methodology

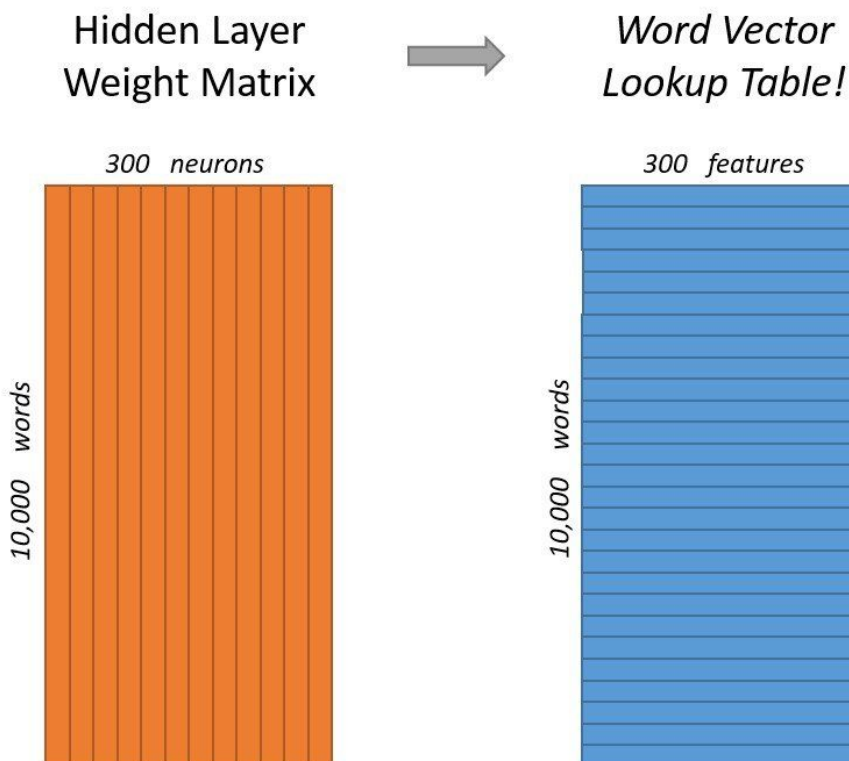
To solve the problem of recommendations, we are using Word2Vec technique. Word2Vec, is a language processing technique, which helps transform words into vectors (the name tells us everything about it). This approach can be used with the help of gensim library in python. It contains two main models for the word embedding generation: CBOW and SkipGram. Both of them are already implemented in Word2Vec.

Word2Vec can serve as a solution to many problems, and indeed it does so for our song recommendations system.

The model we chose for the solution is the Skip-Gram Model:



We trained our model on vocabulary, which consists of unique words, that are in the song lyrics and simultaneously receive each word representation as a vector. The training is as follows: Input Vector x - is a vector with 1 at a position of a certain word in vocabulary and zeros otherwise. The Hidden Layer is $n \times m$ matrix, where n - the amount of words in vocabulary and m - number of features (neurons) used to describe each word (this parameter has to be adjusted). The output layer is a softmax regression classifier. It all comes down to multiplying vectors and matrices. Firstly, the input vector is multiplied by the hidden layer matrix. Then, the resulting vector is multiplied by the output layer matrix ($m \times n$, transposed hidden layer matrix). And lastly softmax is applied to generate a vector of probabilities of each word being a context word. At the end of the training iterations, the Hidden Layer is constructed in a way that it assigns the certain word it's unique representation in our vocabulary, and from it we can get word representation as a vector.




**Remark:* if a word occurs less than 6 times in a vocabulary, the model accepts it as irrelevant and does not include it into the training process. We also deleted all the songs from list, that includes less than 10 relevant words.

Our following planned step was to represent each song as a matrix, where each column is a word in a song. But we faced the problem that the length of each song can differ, so we have decided to choose ten most frequent words that occur in a song. After that, the matrix representation of a song is a 200×10 matrix with columns representing those ten words, and 200 is a length of a word vector (number of features that describes a word in our model).

4. Working process

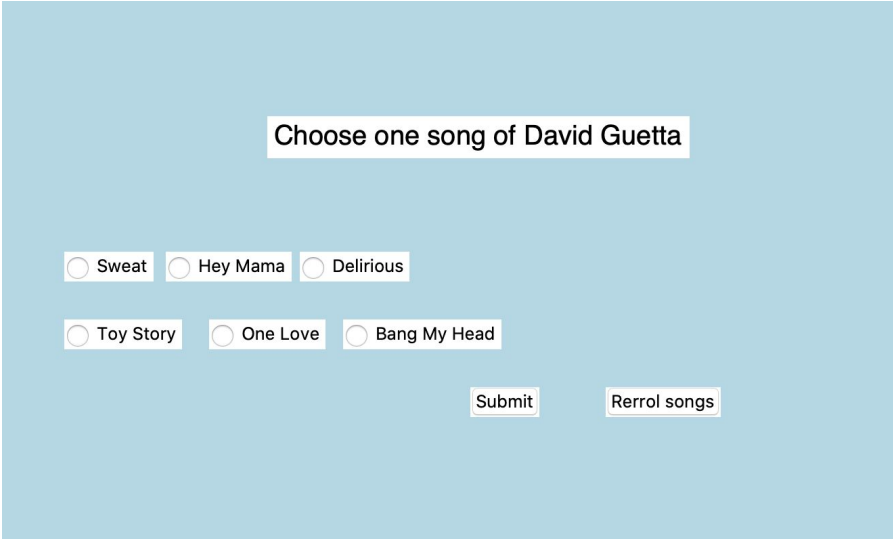
At the beginning of work of a program the user is asked to choose up to three artists.



Choose bands/artist that you listen to (up to 3)

☐ Underworld ☐ David Guetta ☐ Ed Sheeran ☐ Kelly Clarkson ☐ Incognito ☐ Foreigner

After that, the user must choose one song for each of the chosen artists.



Choose one song of David Guetta

☐ Sweat ☐ Hey Mama ☐ Delirious

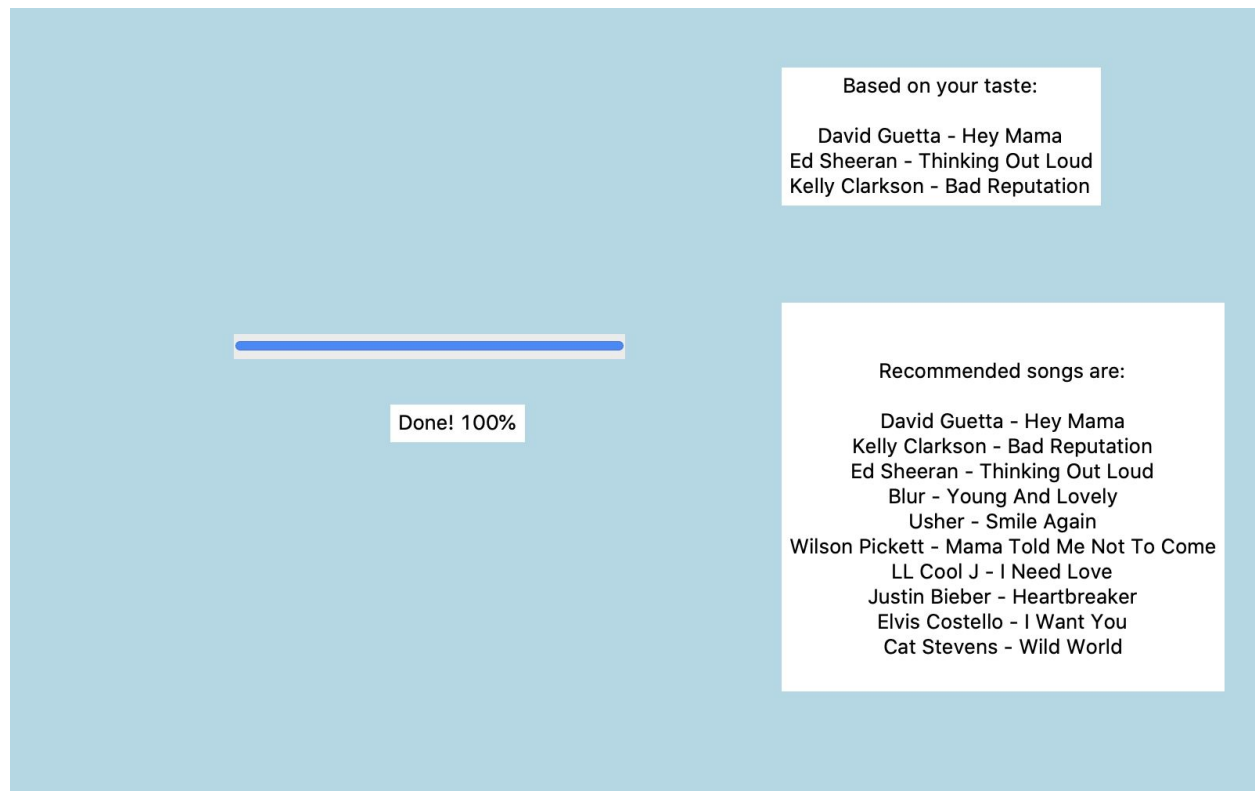
☐ Toy Story ☐ One Love ☐ Bang My Head

After this part, the song matrix for each song is built and then, the corresponding coordinates of each of the matrices are summed up. The x_{ij} of the resulting matrix A is equal to $B_{ij} + C_{ij} + D_{ij}$ where B, C, D are the matrices of songs chosen by the user.

In the next step, we build a matrix for each song in the cleared dataset and find distances between two matrices: our matrix A and all other songs. We find distance between two matrices in the following way:

$$d = \sqrt{(A_{1,1} - S_{1,1})^2 + (A_{1,2} - S_{1,2})^2 + \dots + (A_{n,n} - S_{n,n})^2}$$
, where S is a matrix representation of a song from the dataset.

The matrices with the smallest differences are the closest and that means their respective songs are the most similar in context of words. We choose 10 most similar songs and here is the result:



In the output, the first three songs are almost always the same as in input and, that indicates that the algorithm works properly. The following 7 are recommendations based on the user preferences.

5. Conclusion

The aim of this project was to give a user music recommendations based on the preferences, and the program does this pretty well. The recommended songs are mostly common with preferable songs in content or in mood. Not only famous songs are recommended, but also, some that the developers of the project have never met before. This way, it is easy to find new, previously unknown music and that is what makes our program different from available famous music recommendation platforms.



[GitHub link](#)