**Project scanner resume**

In this project, I decided to tackle a challenging problem that spread through almost every field of global employment and recruitment **- the hard task of reading and evaluating a lot of resumes**. This monotonous yet crucial process is a significant bottleneck for companies and organizations seeking to identify and hire the most qualified candidates efficiently.

Reading a large volume of resumes while recruiting for a company or any other business can be a daunting and time-consuming endeavor. Recruiters often find themselves

flooded with hundreds, if not thousands, of applications for a single open position. Manually sort through each resume to assess the candidate's qualifications, experience, and suitability for the role can be an overwhelming and error-prone task, especially when facing tight deadlines or staff shortages.

This project proposes a solution that can streamline the resume screening process by leveraging natural language processing and machine learning techniques. The proposed feature can be seamlessly integrated into a company's existing recruitment system, minimizing the need for extensive training or specialized knowledge. With a user-friendly interface and intuitive design, even recruiters with limited technical expertise can use this tool effectively.

By harnessing the power of text analytics and similarity algorithms, the software can automatically analyze and compare resumes against job descriptions, highlighting the most relevant candidates based on their qualifications and experience. This intelligent filtering mechanism can significantly reduce the time and effort required for manual resume review, allowing recruiters to focus their attention on the most promising applicants.

Moreover, with a basic understanding of front-end development techniques, the software can be easily customized and adapted to suit the unique requirements of different organizations or industries. This flexibility ensures that the tool remains relevant and valuable across a wide range of recruitment scenarios, catering to the diverse needs of modern businesses.

After the introduction and explanation of the problem and its solution, I will explain the course of the project using examples and formulas.

At first, I used a tool called docx2text.   
The reason I used it is because I don't want a complex text file like a Word or PDF file, I want a text file as simple as possible, and this tool does that conversion, from a complex text file to a simple text file.

I did the conversion process for the job requirements and also for every resume entered into the system.

**CountVectorizer**

The second tool I used is CountVectorizer located in the sklearn library.

sklearn library: Scikit-learn is probably the most useful library for machine learning in Python. The sklearn library contains a lot of efficient tools for machine learning and statistical modeling including classification, regression, clustering and dimensionality reduction.

I will explain the CountVectorizer operation with an example:

1. I like to play football.
2. Did you go outside to play tennis.
3. John and I play tennis.

Three sentences from which we will create a list of "tokens".  
In this list, all the words that are in the three sentences will be stored only once, that is, if a word is in several sentences, it will be saved only once.

\*\*tokens = They are small pieces of text that are created by breaking down a longer text.

The list look like this:

["I", "like", "to", "play", "football", "Did", "you", "go", "outside", "tennis", "John" ,"and"].

This step is called "fit".

The next step in the process will be to create a matrix in whose columns will appear the words in the list of tokens, and in the rows will appear the number of sentences, so that the dimension of the matrix will look like this: sentences\*tokens.

The matrix look like this:

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | "I" | "like" | "to" | "play" | "football" | "Did" | "you" | "go" | "Outside" | "tennis" | "john" | "and" |
| Sentence 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Sentence 2 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 0 |
| Sentence 3 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 |

The matrix is ​​called: "document term matrix".

And this phase is called: "transform".

**Cosine Similarity**

To find the similarity between the job requirements and the resume we used "Cosine Similarity".

I will explain "Cosine Similarity" with some examples.

1. I like meat.

2. I love meat.

3. I love meat, I really like meat.

4. I hate meat.

These are 4 sentences that you immediately notice that the first three sentences are similar and the last sentence is not similar at all, and this is a classic example of similarity between sentences.

Suppose and collect many sentences related to meat in this way:

I like meat.

I love meat.

I really enjoy meat.

I hate meat.

I can't stand meat.

Meat is delicious.

Meat is disgusting.

I adore eating meat.

I despise meat.

Meat is my favorite food.

We will not be able to differentiate between the similarities between the posts quickly with the help of our eyes, for this we will use a computer that will make the comparison quickly, and this is where "Cosine Similarity" comes in.

"Cosine Similarity" is a quick way to calculate matrices and see how similar or different things are.

So, how does it work?

As I said, I will explain with examples.

1. "Hello, World!"  
2. "Hello!"

We have two sentences and we want to check similarity between them, so first of all we will build a matrix whose rows are the sentences, and the columns are the words, inside the matrix we will write down how many times the word appeared in the sentence.

(In the code we get the matrix from CountVectorizer)

|  |  |  |
| --- | --- | --- |
|  | Hello | World |
| "Hello, World!" | 1 | 1 |
| "Hello!" | 1 | 0 |

Then we will draw a two-dimensional graph that looks like this:

2

1

World

Hello

2

1

The X axis is the number of times we saw the word Hello, and the Y axis is the number of times we saw the word World.

Because in the first sentence both words appear in it, we will place his point in the middle of the graph at the point .(1,1)

In the second sentence there is only one word, the word "Hello", so its point will be (1,0), and we will place it only on the X-axis.

The graph looks like this after placing the points:

(1,1)

2

1

(1,0)

World

Hello

2

1

Now we draw lines to each point in the graph from the beginning of the axes (0,0).

We can see that we have created an angle of 45 degrees.

(1,1)

2

1

45

(1,0)

World

Hello

2

1

**edge cases**

1. that the second sentence is: "Hello, Hello, Hello".

Its point on the graph will be (3,0), and the "Cosine Similarity" will remain 0.71 because the angle remains the same angle.

1. The two sentences are the same, that is, the points in the graph are the same, so the angle between the points is zero, because there is only line 1, so the "Cosine Similarity" will be equal to 1.

1. The two sentences look like this:
2. Hello. 2.World.

Their points are:(1,0),(0,1).

2

(0,1)

1

90

(1,0)

World

Hello

2

1

There is no similarity in them at all.

**the formula**

There is a formula that can be used without placing the points on the graph and calculating the angle between them.

I will present the formula now.

Explanation of the parameters:

= total calculation

= An index that runs over the list of all the words of the sentences.

*n = Count all the words in all the sentences.*

*A,B = Denotes the sentences.*

== 0.71

With the help of this equation it is possible to compare only two sentences at the same time.