

## ## TF-IDF Vectorization

TF-IDF (Term Frequency-Inverse Document Frequency) vectors for each document. This will give you a matrix where each column represents a word in the general vocabulary (all words that appear in at least one document) and each column represents a restaurant, as before.

TF-IDF is the statistical method of assessing the meaning of a word in a given document.

```
df_percent.set_index('name',  
inplace=True)
```

```
indices =  
pd.Series(df_percent.index)
```

```
# Creating tf-idf matrix  
tfidf =  
TfidfVectorizer(analyzer='word',  
ngram_range=(1, 2), min_df=0,  
stop_words='english')
```

```
tfidf_matrix =  
tfidf.fit_transform(df_percent['reviews_list'])
```

```
cosine_similarities =  
linear_kernel(tfidf_matrix,  
tfidf_matrix)
```

Now the last step for creating a Restaurant Recommendation System is to write a function that will recommend restaurants:

```
def recommend(name, cosine_similarities = cosine_similarities):  
    # Create a list to put top restaurants  
    recommend_restaurant = []  
  
    # Find the index of the hotel entered  
    idx = indices[indices == name].index[0]  
  
    # Find the restaurants with a similar cosine-sim value and order them from biggest number  
    score_series = pd.Series(cosine_similarities[idx]).sort_values(ascending=False)  
  
    # Extract top 30 restaurant indexes with a similar cosine-sim value  
    top30_indexes = list(score_series.iloc[0:31].index)  
  
    # Names of the top 30 restaurants  
    for each in top30_indexes:  
        recommend_restaurant.append(list(df_percent.index)[each])  
  
    # Creating the new data set to show similar restaurants  
    df_new = pd.DataFrame(columns=['cuisines', 'Mean Rating', 'cost'])  
  
    # Create the top 30 similar restaurants with some of their columns  
    for each in recommend_restaurant:  
        df_new = pd.concat([df_new, df_percent[['cuisines', 'Mean Rating', 'cost']][df_percent.index == each].sample()] )  
  
    # Drop the same named restaurants and sort only the top 10 by the highest rating  
    df_new = df_new.drop_duplicates(subset=['cuisines', 'Mean Rating', 'cost'], keep=False)  
    df_new = df_new.sort_values(by='Mean Rating', ascending=False).head(10)  
  
    print('TOP %s RESTAURANTS LIKE %s WITH SIMILAR REVIEWS: ' % (str(len(df_new)), name))  
  
    return df_new  
recommend('Pai Vihar')
```