Resume Classification using Transformer Models, Supervised Learning, and Unsupervised Learning

1. Dataset

The dataset comprises approximately 2400 resumes, which are labelled according to the position the candidate is applying for. There are 24 unique candidate positions. The dataset contains the candidate's resume as a string and also the candidate's resume as the html file. For this project, only the string version was used. At the time of writing, the dataset is available at: https://www.kaggle.com/datasets/snehaanbhawal/resume-dataset.

2. Feature extraction and preprocessing

All models used in this project are trained on the same 80% of the data and were tested on the other 20% of the data. The split was stratified to preserve the distribution of labels between the training and the testing subset.

There are a variety of text preprocessing methods that can improve the quality of the dataset. Examples of this are conversion of characters from uppercase to lowercase, correction of spelling errors (which in this dataset were too much of an uncommon occurence), removal of punctuation, reduction of replicated characters, removal of stopwords and word stemming/lemmatization.

For this task and the feature extraction process, we used TF-IDF and removed the stopwords for all of the models used. For the unsupervised Bisecting K-Means model and the XLNet model we also tokenized (with Bert, XLNetTokenizer respectively) the text values. Removal of stopwords was done mostly due to memory limitations as it has not been proven to provide any significant jumps in accuracy.

3. Supervised Model

The supervised model used is a SVM with rbf kernel. Automatic hyperparameter search was performed in order to find the best C and Gamma parameters. The search maximizes the 10 fold cross-validation F1 weighted score. Hyperparameters search ranges are as follows:

- C - [1e-2, 1e5]
- Gamma - [1e-5, 1e5]

Hyperparameter optimization results

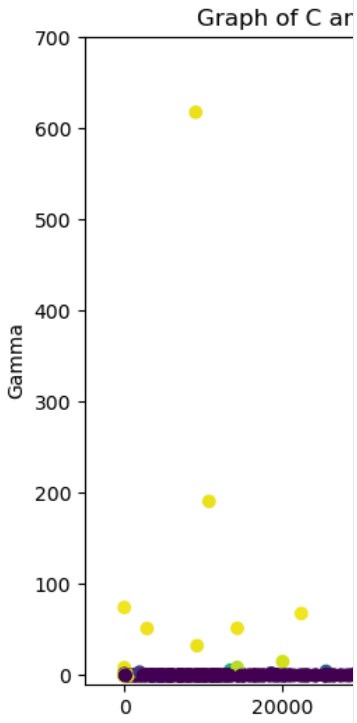
Best results:

- C = 13018.765058090445
- Gamma = 0.00010122287630670668

Classification report

Category	Precision	Recall	F1-Score	Support
ACCOUNTANT	0.8077	0.8750	0.8400	24.0
ADVOCATE	0.4800	0.5000	0.4898	24.0
AGRICULTURE	0.5000	0.2308	0.3158	13.0
APPAREL	0.7647	0.6842	0.7222	19.0
ARTS	0.5556	0.7143	0.6250	21.0
AUTOMOBILE	1.0000	0.1429	0.2500	7.0
AVIATION	0.7222	0.5417	0.6190	24.0

BANKING	0.5926	0.6957	0.6400	23.0
BPO	1.0000	0.2500	0.4000	4.0
BUSINESS-DEVELOPMENT	0.5385	0.5833	0.5600	24.0
CHEF	0.9048	0.7917	0.8444	24.0
CONSTRUCTION	0.8095	0.7727	0.7907	22.0
CONSULTANT	0.2632	0.2174	0.2381	23.0
DESIGNER	0.8636	0.9048	0.8837	21.0
DIGITAL-MEDIA	0.7000	0.7368	0.7179	19.0
ENGINEERING	0.6786	0.7917	0.7308	24.0
FINANCE	0.8571	0.7500	0.8000	24.0

 FITNESS	0.8421	0.6957	0.7619	23.0
HEALTHCARE	0.6071	0.7391	0.6667	23.0
HR	0.7917	0.8636	0.8261	22.0
INFORMATION-TECHNOLOGY	0.6129	0.7917	0.6909	24.0
PUBLIC-RELATIONS	0.6154	0.7273	0.6667	22.0
SALES	0.6000	0.5217	0.5581	23.0
TEACHER	0.7391	0.8500	0.7907	20.0
accuracy	0.6761	0.6761	0.6761	0.6761
macro avg	0.7019	0.6405	0.6429	497.0
weighted avg	0.6840	0.6761	0.6685	497.0

4. Unsupervised Model

Bisecting K-Means is a hierarchical clustering algorithm that iteratively splits clusters into 2 subclusters, using K-Means (k=2). The choice for the cluster to be split is either the cluster that contains the most data points (largest cluster) or the cluster with the biggest sum of squared errors within (biggest inertia). Thus, the model takes 2 hyperparameters: bisecting strategy and k (number of clusters needed to stop the algorithm) [3].

The optimization objective used was the Silhouette score and 100 search trials were run with the following ranges used for the hyperparameters:

- K - [10, 30]
- Bisecting strategy - {‘largest_cluster, ‘biggest_inertia}

Both Tf-Idf and embeddings extracted with Bert pretrained model were used as feature selection techniques for this model. The results for both techniques are discussed in the following chapters.

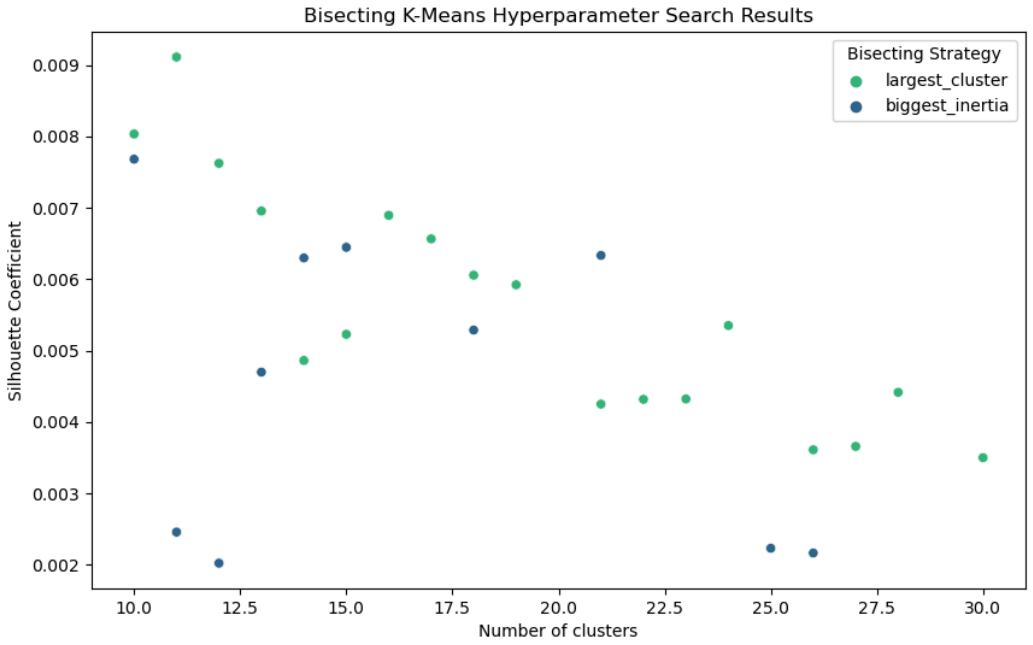
4.1. TF-IDF preprocessing

Hyperparameter optimization results

Best Silhouette score: 0.009

Best parameters: n_clusters: 11, bisecting_strategy: 'largest_cluster'.

From the graph below, it can be observed that trials with the ‘largest_cluster’ bisecting strategy obtain higher Silhouette coefficients, compared to choosing by ‘biggest_inertia’, when the number of clusters is equal.



Number of examples in each cluster:

- Cluster 1 - 94
- Cluster 2 - 233
- Cluster 3 - 233
- Cluster 4 - 329
- Cluster 5 - 49
- Cluster 6 - 171
- Cluster 7 - 168
- Cluster 8 - 262
- Cluster 9 - 122
- Cluster 10 - 57
- Cluster 11 - 269

As seen in the cluster distribution, the cluster sizes are relatively equal, which is a commont tendency for the bisecting k-mean algorithm. This is likely being caused by

the largest cluster being split at each step in the algorithm. The accuracy metrics on the test set are unbalanced, with a lot of classes having 0 accuracy and a few of them having considerably higher accuracy, over 0.5.

Classification report

Category	Precision	Recall	F1-Score	Support
ACCOUNTANT	0.487	0.833	0.615	24.0
ADVOCATE	0.0	0.0	0.0	24.0
AGRICULTURE	0.0	0.0	0.0	13.0
APPAREL	0.0	0.0	0.0	19.0
ARTS	0.0	0.0	0.0	21.0
AUTOMOBILE	0.0	0.0	0.0	7.0
AVIATION	0.0	0.0	0.0	24.0
BANKING	0.0	0.0	0.0	23.0
BPO	0.0	0.0	0.0	4.0
BUSINESS-DEVELOPMENT	0.228	0.542	0.320	24.0
CHEF	1.0	0.416	0.588	24.0
CONSTRUCTION	0.812	0.590	0.684	22.0
CONSULTANT	0.0	0.0	0.0	23.0
DESIGNER	0.0	0.0	0.0	21.0
DIGITAL-MEDIA	0.105	0.421	0.168	19.0

ENGINEERING	0.538	0.583	0.560	24.0
FINANCE	0.0	0.0	0.0	24.0
FITNESS	0.0	0.0	0.0	23.0
HEALTHCARE	0.118	0.391	0.182	23.0
HR	0.666	0.727	0.695	22.0
INFORMATION-TECHNOLOGY	0.381	0.666	0.484	24.0
PUBLIC-RELATIONS	0.0	0.0	0.0	22.0
SALES	0.220	0.565	0.317	23.0
TEACHER	0.2	0.7	0.311	20.0
accuracy	0.294	0.294	0.294	0.294
macro avg	0.198	0.268	0.205	497.0
weighted avg	0.220	0.294	0.227	497.0

4.2. Tokenization and extraction of embeddings with Bert pretrained model

Hyperparameter optimization results

Best Silhouette score: 0.056

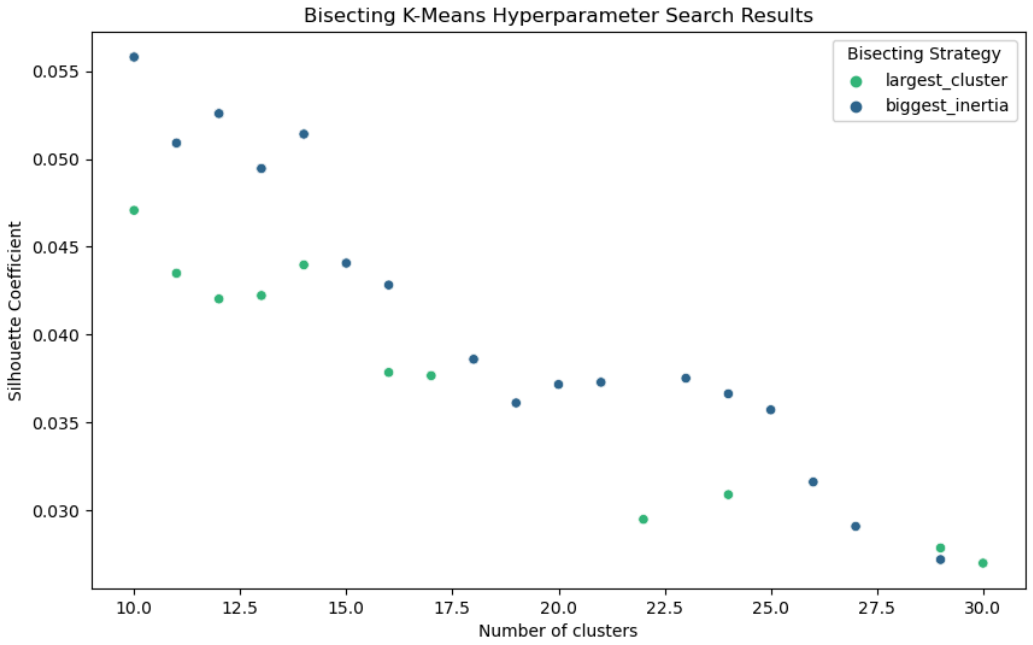
Best parameters: n_clusters: 10, bisecting_strategy: 'biggest_inertia'

- Cluster 1: 331
- Cluster 2: 353
- Cluster 3: 79

- Cluster 4: 200
- Cluster 5: 182
- Cluster 6: 114
- Cluster 7: 278
- Cluster 8: 196
- Cluster 9: 195
- Cluster 10: 59

In the graph below, it can be observed that trials with the splitting strategy ‘biggest_inertia’ scored a higher Silhouette coefficient, unlike the trials run with the Tf-Idf feature selection.

Although the Silhouette coefficients are overall higher than the trials run with Tf-Idf feature selection, the overall test scores are lower. This indicates a poor correlation between the silhouette coefficient and the test evaluation metrics (accuracy, F1-score). For this reason, an evaluation metric that would penalize differences between cluster sizes (as measured by the count of points) might be more suitable, especially for a large number of categories.



Classification report

Category	Precision	Recall	F1-Score	Support
ACCOUNTANT	0.27475	0.708	0.395	24.0
ADVOCATE	0.1	0.25	0.143	24.0
AGRICULTURE	0.0	0.0	0.0	13.0
APPAREL	0.0	0.0	0.0	19.0
ARTS	0.0	0.0	0.0	21.0
AUTOMOBILE	0.0	0.0	0.0	7.0
AVIATION	0.0	0.0	0.0	24.0
BANKING	0.0	0.0	0.0	23.0
BPO	0.0	0.0	0.0	4.0
BUSINESS-DEVELOPMENT	0.12	0.5	0.193	24.0
CHEF	0.846	0.458	0.594	24.0
CONSTRUCTION	0.0	0.0	0.0	22.0
CONSULTANT	0.0	0.0	0.0	23.0
DESIGNER	0.0	0.0	0.0	21.0
DIGITAL-MEDIA	0.121	0.421	0.188	19.0
ENGINEERING	0.286	0.583	0.383	24.0
FINANCE	0.0	0.0	0.0	24.0

FITNESS	0.0	0.0	0.0	23.0
HEALTHCARE	0.0	0.0	0.0	23.0
HR	0.0	0.0	0.0	22.0
INFORMATION-TECHNOLOGY	0.2856	0.333	0.308	24.0
PUBLIC-RELATIONS	0.0	0.0	0.0	22.0
SALES	0.172	0.739	0.279	23.0
TEACHER	0.35	0.35	0.35	20.0
accuracy	0.201	0.201	0.201	0.201
macro avg	0.106	0.181	0.118	497.0
weighted avg	0.119	0.201	0.132	497.0

5. State of the art - XLNet Model

XLNet is an auto-regressive language model which outputs the joint probability of a sequence of tokens based on the transformer architecture with recurrence. Its performance grants it the state-of-the-art status in NLP tasks. The model calculates the probability of a word token to appear against all permutations of tokens in a sentence, unlike BERT, which only calculates the probability of a word token occurring in a sentence based on the tokens on the left and right of the target token.

What the model does is to multiply all of the conditional distributions of one token occurring, given the other tokens in the observed sequence. It predicts each word in a sequence using any combination of other words in that sequence.

Some other well known architecture for NLP tasks is LSTM which takes into account the relation between adjacent tokens. But transformers like BERT and XLNet have

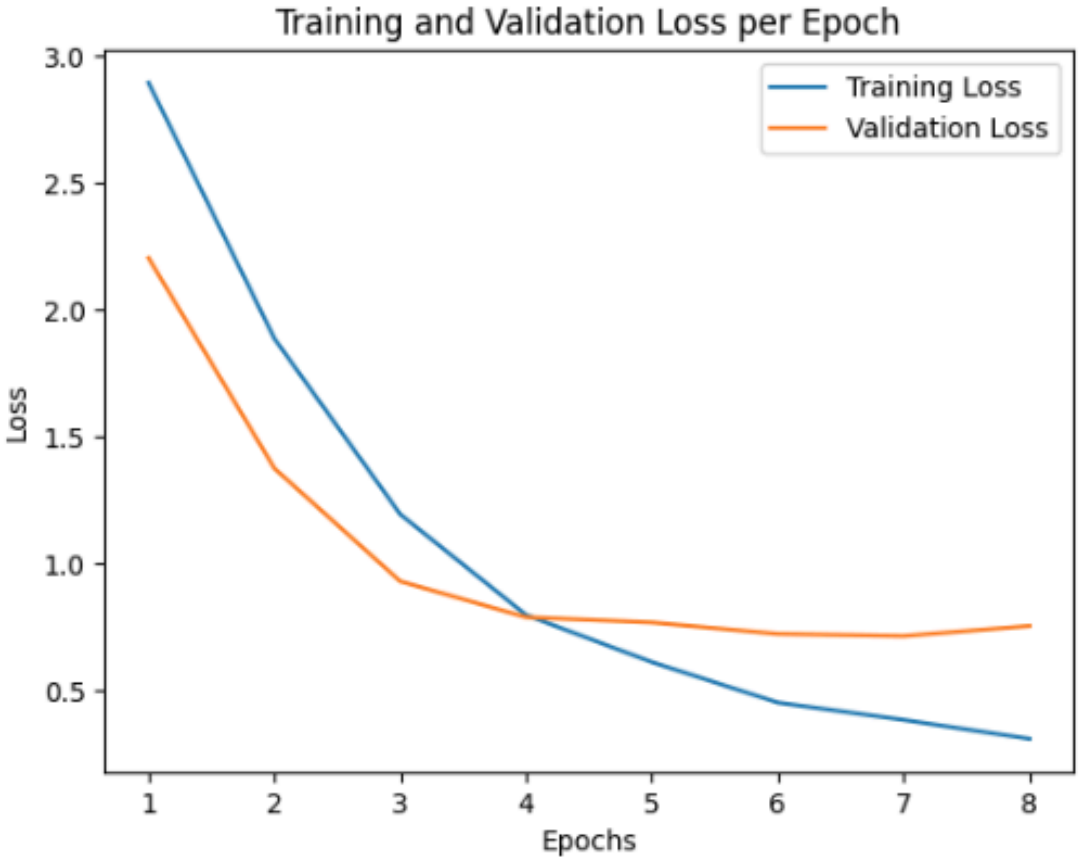
been proven to work better in learning from long-distance dependencies that exist in the text.

5.1 Implementation details

The implementation used was the one from Hugging Face' s Transformer Library. The categorical labels are encoded into numerical data. The XLNetTokenizer was used for extracting features from the resumes and it was configured with a max token length of 512, while also allowing truncation and paddding of documents to fit within the max token limit. The dataset was divided into training, validation and test using a 64-16-20 split. This way, the hyperparameters of the model were tuned to minimize the validation loss. The model used for fine tuning is of type XLNetForSequenceClassification and the hyperparameter optimization lead to the following configuration:

- learning rate: 1e-5 (Adam optimizer)
- number of training epochs: 8 (early stopped, maximum of 10)
- batch size: 12 (highest available, limited by computational resources)

The project was run on GPU P-100 in a Kaggle Notebook, which helped speed up the training.



Classification report

Label	Precision	Recall	F1-Score	Support
0	0.96	0.96	0.96	24
1	0.83	0.62	0.71	24
2	0.80	0.31	0.44	13
3	0.55	0.58	0.56	19
4	0.75	0.71	0.73	21
5	0.00	0.00	0.00	7
6	0.80	0.67	0.73	24
7	0.67	0.70	0.68	23
8	0.00	0.00	0.00	4
9	0.92	0.96	0.94	24
10	1.00	0.92	0.96	24
11	0.91	0.91	0.91	22
12	0.78	0.91	0.84	23
13	0.95	0.95	0.95	21
14	0.74	0.89	0.81	19
15	0.92	1.00	0.96	24

16	0.96	1.00	0.98	24
17	0.81	0.74	0.77	23
18	0.69	0.87	0.77	23
19	0.88	1.00	0.94	22
20	0.85	0.92	0.88	24
21	0.70	0.86	0.78	22
22	0.96	0.96	0.96	23
23	0.86	0.95	0.90	20
Accuracy	0.83	0.83	0.83	497
Macro/Weighted	0.76 / 0.82	0.76 / 0.82	0.76 / 0.82	497

6. Comparison

Method	**Accuracy**
TF-IDF + SVM	0.68
TF-IDF + Bisecting K-Means	0.29
Bert features + Bisecting K-Means	0.2
XLNet end-to-end	0.83

7. Conclusion

We tested classical supervised ML approaches, unsupervised algorithms, as well as different preprocessing techniques and finally one of the state of the art pretrained models (XLNet), which we fine-tuned on our own dataset for end-to-end classification. On this particular dataset, the clustering algorithm was difficult to tune, as typical clustering evaluation metrics were often uncorrelated with the prediction accuracy on the test set. The supervised approach and the Tf-Idf techniques were outperformed by the XLNet pretrained model, after fine-tuning on the given dataset. This leads to the conclusion that for a similar dataset (low number of datapoints, high features and labels dimensionality), a supervised approach such as SVM outperforms an unsupervised clustering algorithm like Bisecting K-Means, but is ultimately outperformed by a pretrained end-to-end model, like XLNet.