UNIVERSITY OF ZAGREB

FACULTY OF ELECTRICAL ENGINEERING AND COMPUTING

BACHELOR THESIS No. 1621

IMPLEMENTATION OF DOCUMENT ANALYSIS AND COMPARISON FUNCTIONALITIES

Klara Zagajski

Zagreb, June 2024

UNIVERSITY OF ZAGREB
FACULTY OF ELECTRICAL ENGINEERING AND COMPUTING

BACHELOR THESIS No. 1621

IMPLEMENTATION OF DOCUMENT ANALYSIS AND COMPARISON FUNCTIONALITIES

Klara Zagajski

Zagreb, June 2024

UNIVERSITY OF ZAGREB
FACULTY OF ELECTRICAL ENGINEERING AND COMPUTING

Zagreb, 04 March 2024

BACHELOR THESIS ASSIGNMENT No. 1621

Student:	**Klara Zagajski (0036530600)**
Study:	Electrical Engineering and Information Technology and Computing
Module:	Computing
Mentor:	assoc. prof. Marina Bagić Babac

Title: **Implementation of document analysis and comparison functionalities**

Description:

This thesis explores the key features and functionalities of a system for document analysis and comparison. A system that provides tools for text recognition, information extraction, and document classification will be implemented. Text recognition enables automatic extraction of textual content from various document formats, while information extraction enables data structuring. Documents can be classified based on different properties. Additionally, the system will include a comparative analysis of documents to assess the effectiveness and reliability of each implemented functionality.

Submission date: 14 June 2024

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

Zagreb, 4. ožujka 2024.

ZAVRŠNI ZADATAK br. 1621

Pristupnica:	**Klara Zagajski (0036530600)**
Studij:	Elektrotehnika i informacijska tehnologija i Računarstvo
Modul:	Računarstvo
Mentorica:	izv. prof. dr. sc. Marina Bagić Babac

Zadatak: **Implementacija funkcionalnosti analize i usporedbe dokumenata**

Opis zadatka:

Ovaj rad istražuje ključne značajke i funkcionalnosti sustava za analizu i usporedbu dokumenata. Potrebno je implementirati sustav koji pruža alate za prepoznavanje, ekstrakciju informacija i klasifikaciju dokumenata. Prepoznavanje teksta omogućuje automatsko izdvajanje tekstualnog sadržaja iz različitih formata dokumenata, dok ekstrakcija informacija omogućuje strukturiranje podataka. Dokumente je moguće klasificirati na temelju različih svojstava. Osim toga, sustav će uključivati komparativnu analizu dokumenata kako bi se procijenila učinkovitost i pouzdanost svake od implementiranih funkcionalnosti.

Rok za predaju rada: 14. lipnja 2024.

Contents

Introduction

In the contemporary digital landscape, the analysis and comparison of documents have become indispensable in various sectors such as legal, academic, and corporate environments. The capability to extract and process textual information from diverse document formats is crucial for enhancing efficiency and accuracy in these fields and others. Traditional manual methods for document analysis are not only time-consuming but also prone to human error, highlighting the necessity for automated systems.

This thesis addresses the inefficiencies and errors inherent in manual document analysis by developing an automated system for text recognition, information extraction, and document classification. The system is designed to handle various document formats, structure the extracted data, and classify documents based on specific properties. Additionally, the use of various text embeddings in conjunction with machine learning models will be explored to enhance the accuracy and reliability of document classification. By systematically comparing these embeddings and models, the aim is to identify the most effective combinations.

Manual document analysis is especially inefficient when dealing with extensive datasets or complex documents. This thesis aims to overcome these limitations by developing an automated system that leverages advanced machine learning techniques to efficiently extract text from various document formats, preprocess the text to ensure consistency, structure the extracted information accurately, and reliably classify and compare documents. The integration of machine learning models enables the system to learn from the data and improve its performance over time, making it a robust solution for document analysis tasks. By systematically comparing different embeddings and models, it's possible to identify the most effective combinations for accurate and reliable document classification. This comparative analysis not only highlights the strengths and weaknesses of each approach but also provides valuable insights into optimizing the use of machine learning for document analysis.

The contributions of this thesis include the implementation of a system capable of extracting text from multiple document formats, implementation of preprocessing techniques to normalize and prepare text data for further analysis, and the application of sophisticated machine learning models for effective document classification. Furthermore, this thesis establishes a robust evaluation framework for assessing the reliability and effectiveness of these machine learning models in text recognition, information extraction, and document classification tasks. This approach not only enhances the accuracy and efficiency of the document analysis process but also contributes valuable insights into the best practices for implementing machine learning in document analysis systems.

1. Relevant Technologies and Tools

Document analysis has been extensively researched, resulting in a variety of methods and technologies aimed at improving text processing tasks. Automated approaches have been developed, leveraging advancements in natural language processing (NLP) and machine learning. This section outlines the key technologies and tools utilized in this thesis, focusing on their relevance and application in text recognition, information extraction, and document classification.

Natural Language Processing (NLP): NLP is a field of artificial intelligence that focuses on the interaction between computers and human language. It involves the development of algorithms and models to process and analyze large amounts of natural language data. NLP techniques are essential for processing and analyzing textual data. Libraries such as NLTK (Natural Language Toolkit), SpaCy, and Stanford NLP provide robust tools for tasks such as tokenization. These tools facilitate the extraction of structured information from unstructured text, enabling more accurate analysis and classification.

Machine Learning: Machine learning is a branch of artificial intelligence that involves training algorithms to learn from and make predictions or decisions based on data. In the context of document analysis, machine learning models, including unsupervised and supervised learning algorithms, have been widely adopted for document classification tasks. Unsupervised learning approaches, such as clustering algorithms, are used to group similar documents without prior knowledge of the categories. Supervised learning models such as Support Vector Machines (SVM), Decision Trees, and Naive Bayes classifiers are trained on labeled datasets to categorize documents into predefined classes. These classifiers are evaluated in this thesis. SVM is effective for high-dimensional spaces and is widely used for text classification. KNN is a simple, instance-based learning algorithm that classifies documents based on the majority class among the nearest neighbors. Naive Bayes is a probabilistic classifier that applies Bayes' theorem with strong independence assumptions between features.

Text Embeddings: Embedding methods transform text into numerical vectors that can be used as input for machine learning models. This thesis explores several embedding techniques; GloVe, Word2Vec, TF-IDF, and Doc2Vec. GloVe (Global Vectors for Word

Representation) and Word2Vec generate dense vector representations based on word co-occurrence in large corpora, capturing semantic relationships between words. TF-IDF (Term Frequency-Inverse Document Frequency) is a statistical measure that evaluates the importance of a word in a document relative to a corpus, while Doc2Vec extends Word2Vec to generate vector representations for entire documents.

Evaluation Metrics: Evaluating the performance of the document analysis system requires robust metrics. Common metrics include accuracy, precision, recall, and F1-score. These metrics provide insights into the effectiveness of the text recognition, information extraction, and classification functionalities, enabling a comprehensive assessment of the system's performance.

By integrating these technologies and tools, this thesis aims to develop a robust and efficient system for automated document analysis, capable of handling diverse document formats and delivering high accuracy in text recognition, information extraction, and classification tasks.

2. Data Description

2.1. Overview

The dataset utilized for this thesis is sourced from Kaggle and consists of resumes labeled as job categories [1]. The individual resume examples were obtained from livecareer.com website. It comprises of an archive file organized into folders, each named according to 24 distinct professional classes, such as Engineering, Finance, and Healthcare. Each folder contains multiple PDF files of resumes. In total, the dataset includes 2,485 documents, providing a diverse and substantial collection of documents for analysis.

The dataset is fundamental for implementing and testing the system's functionalities, including text recognition, information extraction, and document classification. Each document undergoes a preprocessing phase where text is extracted using specialized functions for PDF files and normalized to ensure consistency. This preprocessing includes steps such as converting text to lowercase, removing special characters, tokenizing, removing stop words, and lemmatizing.

2.2. Dataset Cleaning

To ensure the quality and consistency of the dataset, a cleaning process was undertaken. Due to the minimal file counts in the AUTOMOBILE (36 files) and BPO (22 files) classes, these categories were excluded as significant outliers. Including these outliers could have introduced bias and reducde the generalization ability of the trained machine learning models. To prevent bias and ensure robust model training, 22 classes were retained for analysis.

Outlier classes with significantly fewer instances can skew the performance of machine learning models, making them overly sensitive to those particular classes and potentially reducing the model's ability to generalize to more balanced datasets. By excluding these outliers, the model can focus on learning from a more representative distribution of classes, leading to improved overall performance.

The preprocessed dataset is then used to train and evaluate various machine learning models. A bar graph illustrating the distribution of classes in the dataset (Figure 1) is included to provide a clear visual representation of the dataset's composition and ensure a balanced approach to model training and evaluation.

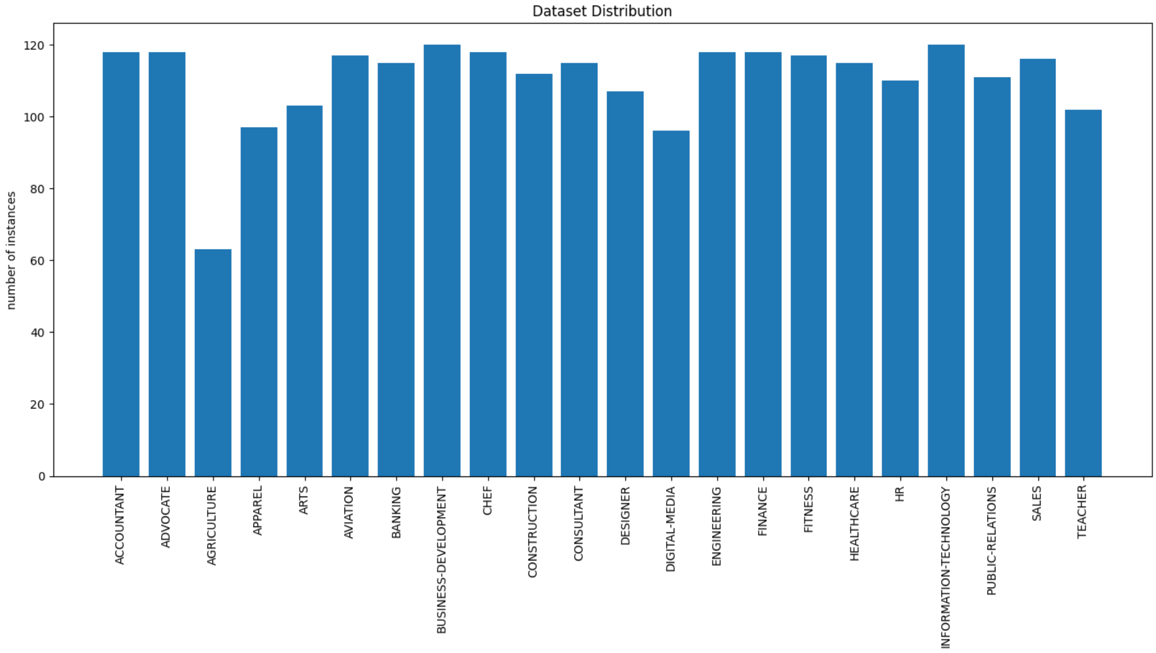


Figure 1: Class Distribution

2.3. Topics and Word Frequency

To gain deeper insights into the content and patterns within the documents, topic modeling and word frequency analysis were performed [14]. This section discusses some of the average topic distributions and the most frequent words across different professional classes in the dataset. The topics are determined by a topic modeling algorithm (LDA - Latent Dirichlet Allocation) and are labeled based on the three most frequent words that best describe the underlying theme. The following graphs depict the average topic distributions for selected classes.

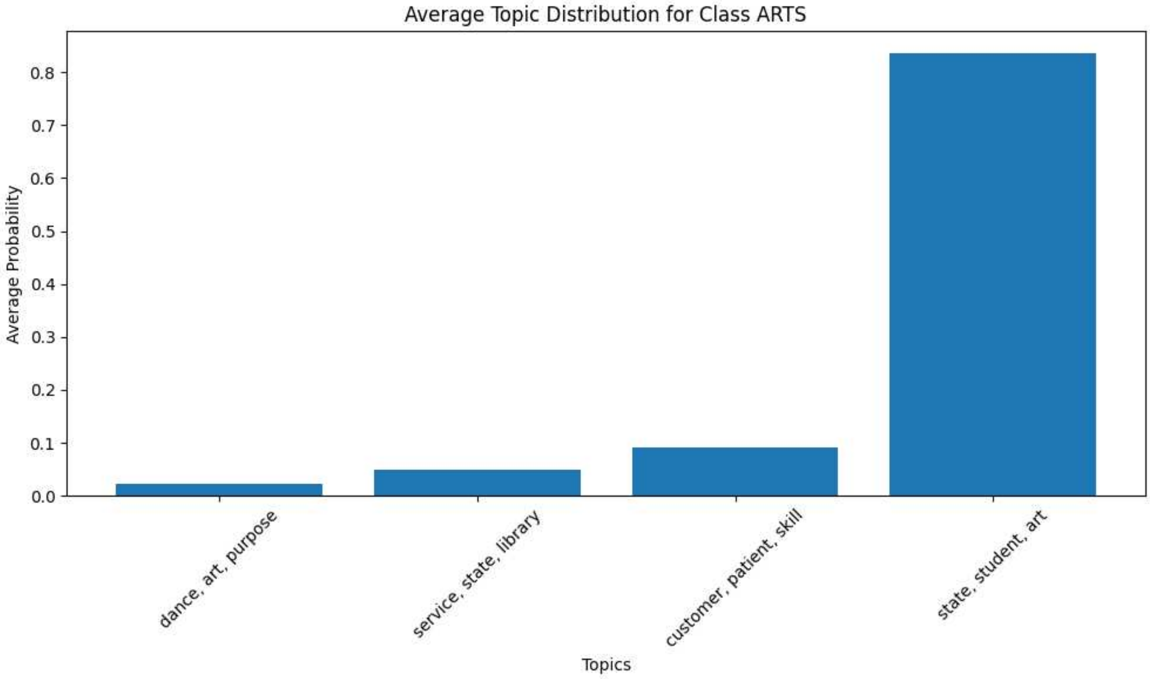


Figure 2 Average Topic Distribution for Class ARTS

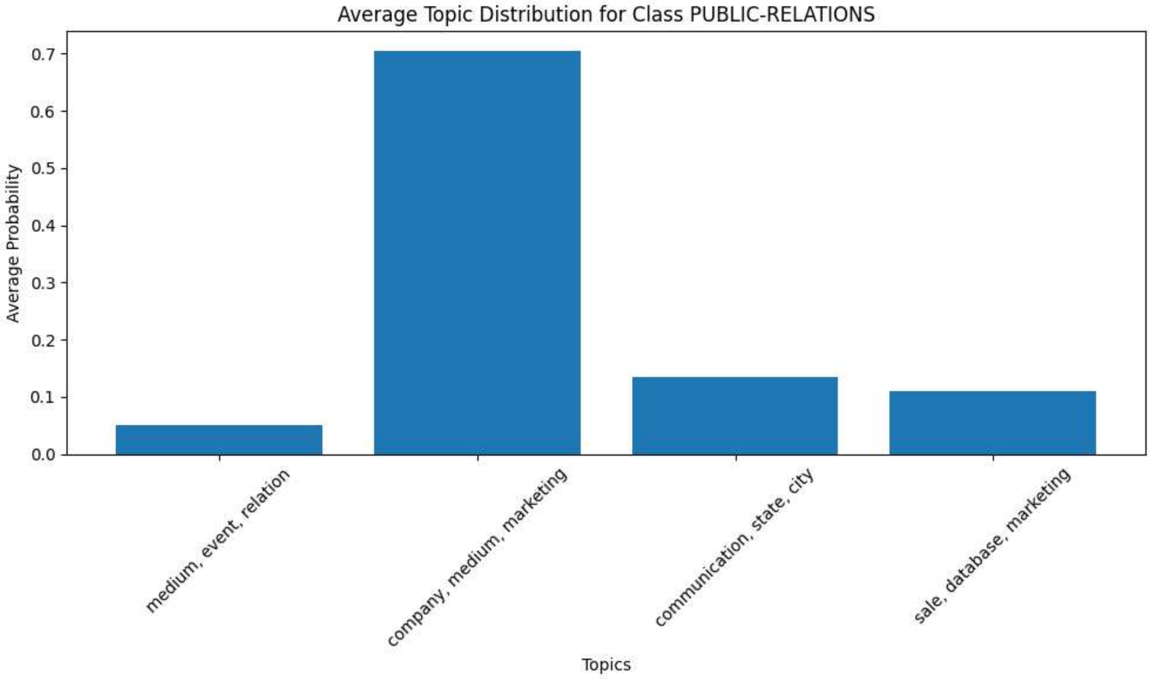


Figure 3 Average Topic Distribution for Class PUBLIC-RELATIONS

In the topic distributions for the ARTS and PUBLIC-RELATIONS classes (Figure 2 and Figure 3 respectively), one topic is visibly prevalent, with probabilities of 84% and 70%. This indicates a strong thematic focus within these classes, suggesting that certain key topics dominate the content in these professional fields. For the ARTS class, the prevalence of the topic labeled "state, student, art" makes sense as it captures the emphasis on artistic activities and education, which are central to this field. This stands in contrast to less frequent topics like "dance, art, purpose," which are also relevant but not as central to the overall theme. Similarly, in the PUBLIC-RELATIONS class, the dominant topic labeled "company, medium, marketing" aligns well with the primary activities in PR, focusing on corporate communication and marketing strategies.

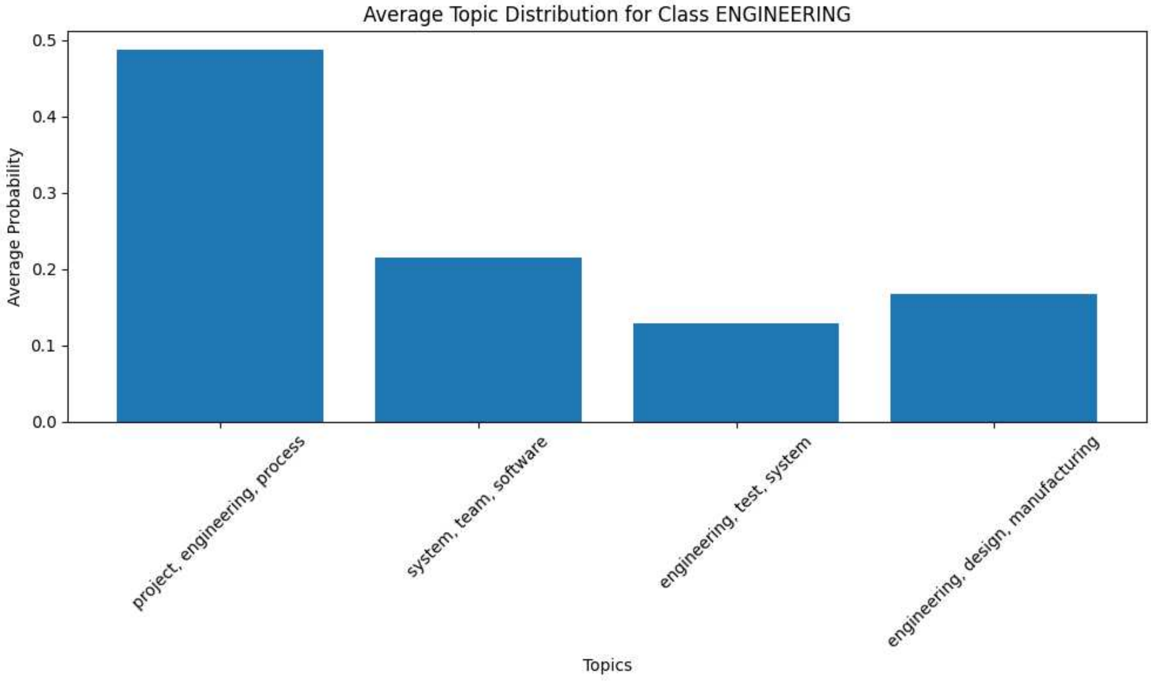


Figure 4 Average Topic Distribution for Class ENGINEERING

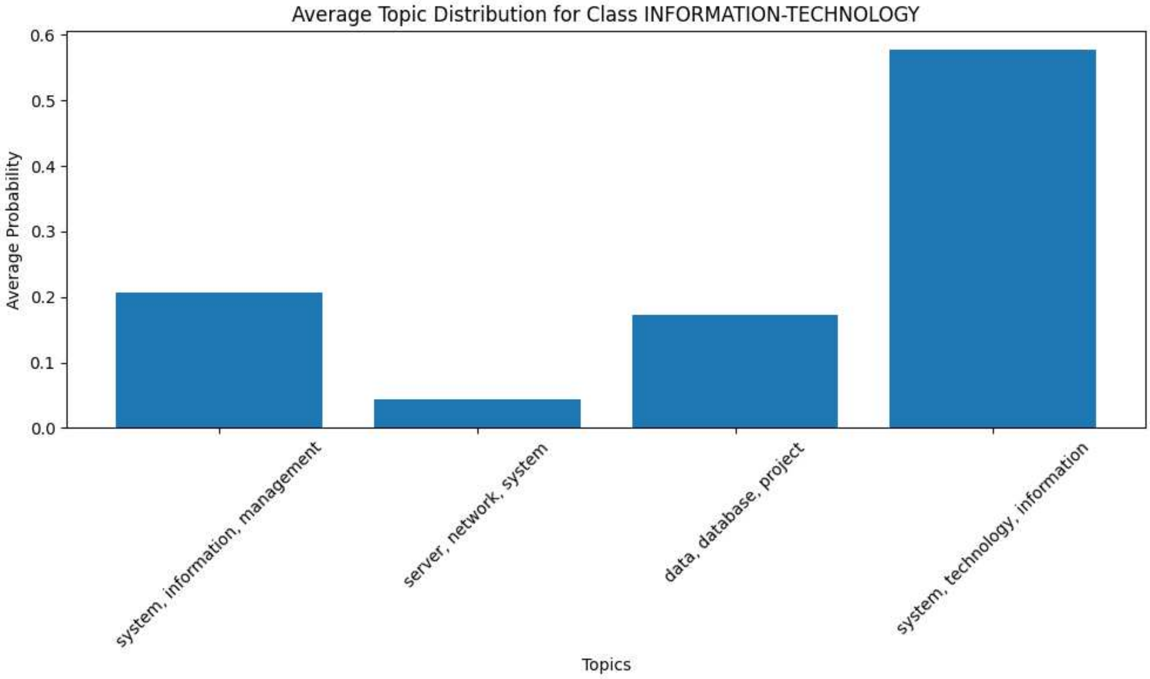


Figure 5 Average Topic Distribution for Class INFORMATION-TECHNOLOGY

In the ENGINEERING class (Figure 4) and INFORMATION-TECHNOLOGY class (Figure 5), the topic distribution is more evenly spread, with multiple topics having significant probabilities. For the ENGINEERING class, topics such as "project, engineering, process" and "system, team, software" reflect the diverse nature of engineering roles, which involve various aspects of project management, technical processes, and system design. This spread of topics indicates that engineering documents cover a broad range of activities and specializations. Similarly, in the INFORMATION-TECHNOLOGY class, topics like "system, technology, information" and "system, information, management" highlight the core responsibilities in IT roles, including system management and technological implementation. These topics emphasize the technical and operational focus inherent in IT professions.

The following heatmap shows the frequency of the top 10 words across all document classes in the dataset. It provides insights into common terms used in different professions. For instance, "customer" and "management" are highly frequent in sales and business development roles, while "project" and "engineering" are predominant in engineering roles. It is also evident that the words "state," "company," "city," and "name" are present in nearly all classes, which is logical given that the dataset consists of resumes, where such information is commonly included.

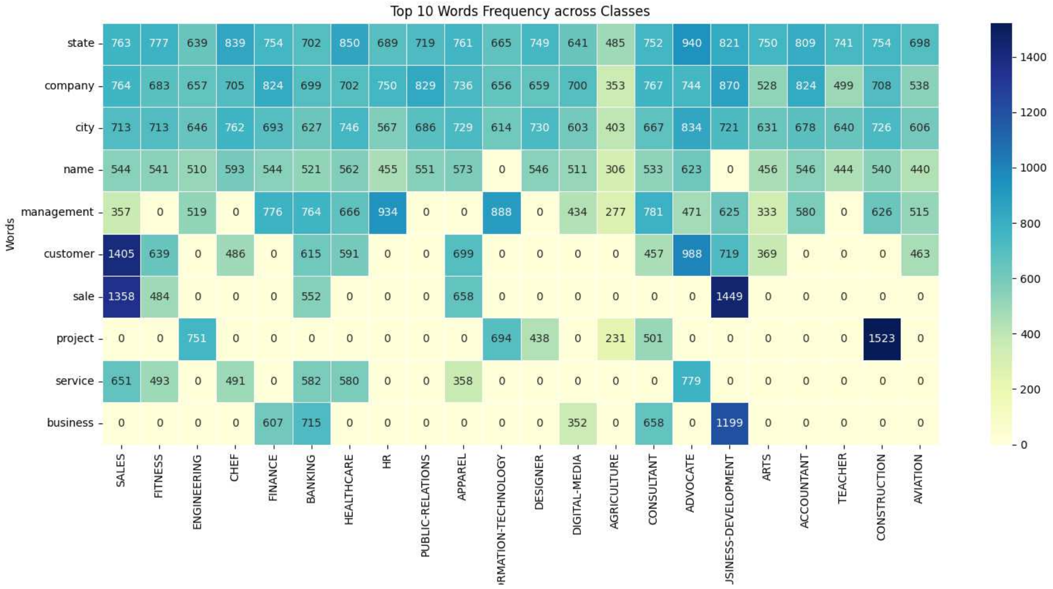


Figure 6 Words Frequency across Classes

3. Methodology

The proposed system for document analysis and comparison consists of several key components: text extraction, preprocessing, embedding generation, model training and evaluation, and document analysis. Each component is responsible for a specific function, and together they form a comprehensive workflow for analyzing and classifying documents.

3.1. Data Preprocessing

Data preprocessing is essential for any data-driven analysis. Effective preprocessing transforms raw data into a clean and usable format. The preprocessing steps are as follows:

3.1.1. Text Extraction

Text extraction is a critical initial step in the preprocessing phase, involving the conversion of textual content from various document formats into machine-readable text. This system specifically handles PDF and DOCX files, using tailored functions to accurately extract the embedded text. The specialized functions ensure that text is accurately extracted from each document format, providing a reliable foundation for subsequent preprocessing steps. The extracted text can then be used for further preprocessing, ensuring consistency and accuracy across the dataset.

3.1.2. Text Preprocessing

The extracted text undergoes several preprocessing steps to normalize and prepare it for analysis. These steps are implemented using a custom preprocessing script and include the following processes:

Lowercasing: All characters in the text are converted to lowercase to ensure uniformity and reduce the complexity of text analysis.

Removing Special Characters: Punctuation, symbols, and other non-alphanumeric characters are eliminated to clean the text and focus on the meaningful content.

Tokenizing: The text is split into individual words or tokens, which serve as the basic units for further processing and analysis.

Removing Stop Words: Commonly used words such as "and," "the," and "is" are removed, as they do not contribute significantly to the semantic meaning of the text. Additionally, two specific tokens ("ï¼" and "â") were included in the stop word list due to their frequent occurrence in the dataset, where they were used to anonymize private information in the original CVs.

Lemmatizing: Words are reduced to their base or root form to standardize different morphological variations, ensuring that words with similar meanings are treated as a single term.

The purpose of these preprocessing steps is enhancing the quality and consistency of the text data. By transforming the raw text into a cleaner and more structured format, the subsequent stages of text analysis and machine learning are made more efficient and accurate.

3.2. Embeddings

Text embeddings are crucial for transforming textual data into numerical vectors that machine learning models can process. This thesis explores several embedding techniques to capture the semantic relationships between words and documents.

GloVe (Global Vectors for Word Representation) generates dense vector representations based on the co-occurrence of words in large corpora. By analyzing the global word-word co-occurrence matrix, GloVe captures the semantic relationships between words, allowing for meaningful vector representations.

Word2Vec produces word embeddings using either the Continuous Bag of Words (CBOW) or Skip-Gram model. Both models are trained on word co-occurrence data but differ in their approach: CBOW predicts a target word from its context, while Skip-Gram predicts context words from a target word. These embeddings capture semantic relationships and similarities between words.

TF-IDF (Term Frequency-Inverse Document Frequency) is a statistical measure that evaluates the importance of a word in a document relative to a corpus. It generates sparse vector representations by multiplying the term frequency (how often a word appears in a document) by the inverse document frequency (how common or rare the word is across all documents). This method highlights important terms and reduces the impact of frequently occurring but less informative words.

Doc2Vec extends the Word2Vec model to generate vector representations for entire documents. It captures the context of the document as a whole, providing a more comprehensive representation of the document's content. This method is particularly useful for tasks that require understanding the broader context and structure of the text.

These embedding techniques are evaluated to determine the most effective method for representing text data in the context of document classification. By transforming textual information into numerical vectors, embeddings facilitate the application of machine learning models, enhancing the system's ability to analyze and classify documents accurately.

3.3. Model Training and Evaluation

3.3.1. Machine Learning Models

Several machine learning models are trained and evaluated to determine their effectiveness in document classification. The following models are employed in this thesis: Support Vector Machines (SVM), K-Nearest Neighbors (KNN), and Naive Bayes.

Support Vector Machines (SVM) is a supervised learning model that is effective for high-dimensional spaces, making it particularly useful for text classification due to its robustness and ability to handle a large number of features. It works by finding the hyperplane that best separates the data into different classes. The objective function for a linear SVM can be represented as:

$$\min_{w,b} \frac{1}{2} \|\mathbf{w}\|^2 \qquad (1)$$

subject to:

$$y_i(\mathbf{w} \cdot \mathbf{x}_i + b) \geq 1 \ \ \forall i \qquad (2)$$

where $\boldsymbol{w}$ is the weight vector, b is the bias term, y_i is the class label for the i-th sample and $\mathbf{x}_i$ is the feature vector for the i-th sample. This constrained optimization problem can be solved using quadratic programming methods. An illustration (Figure 7) showing the hyperplane separating two classes, with support vectors lying on the margin boundaries, helps visualize how SVMs find the optimal separating hyperplane.

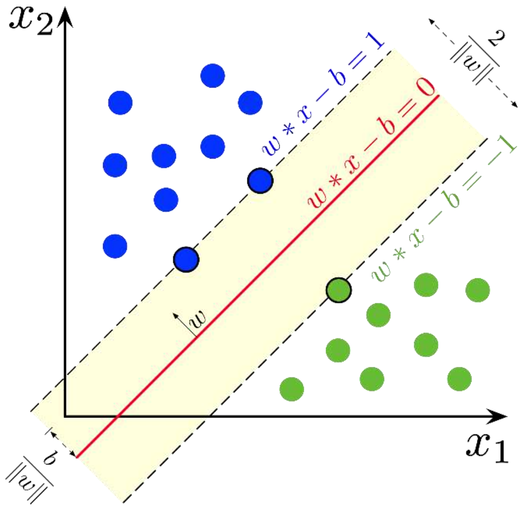


Figure 7: Maximum-Margin Hyperplane and Margins for an SVM Trained with Samples from Two Classes [11]

K-Nearest Neighbors (KNN) is an instance-based learning algorithm that classifies documents based on the majority class among the nearest neighbors in the feature space. It typically uses the Euclidean distance to measure the similarity between instances. The Euclidean distance between two points $\mathbf{x}_i$ and $\mathbf{x}_j$ is given by:

$$d(x_i, x_j) = \sqrt{\sum_{k=1}^{n} (x_{ik} - x_{jk})^2} \quad (3)$$

where x_{ik} and x_{jk} are the k-th features of $\mathbf{x}_i$ and $\mathbf{x}_j$, respectively, and n is the number of features. The boundary created by KNN for different values of k boundary becomes smoother or more complex as k changes, as shown in (Figure 8).

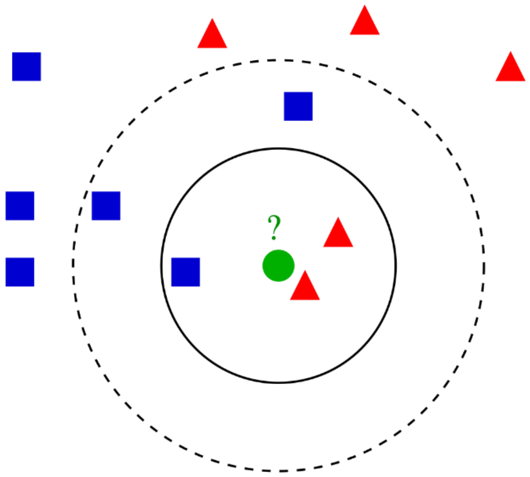


Figure 8: Example of k-NN Classification [12]

The provided figure illustrates the K-Nearest Neighbors (KNN) algorithm. Blue squares and red triangles represent two different classes of data points, while the green circle represents a new data point to be classified. The figure shows the green circle's k nearest neighbors, comprising two red triangles and one blue square. The majority class among these neighbors determines the class of the green circle, highlighting the voting mechanism of KNN. This figure highlights how the parameter k affects the decision boundary, with smaller k values creating more complex boundaries and larger k values smoothing the boundaries, enhancing generalization but risking oversimplification.

Naive Bayes is a probabilistic classifier based on Bayes' theorem. It assumes strong independence between features, making it computationally efficient. The probability of a class C_k given a feature vector $\mathbf{x}$ is calculated as:

$$P(C_k|\mathbf{x}) = \frac{P(C_k)P(\mathbf{x}|C_k)}{P(\mathbf{x})} \tag{4}$$

Since $P(\mathbf{x})$ is constant for all classes, the classification rule can be simplified to:

$$P(C_k|\mathbf{x}) \propto P(C_k)\prod_{i=1}^{n} P(x_i|C_k) \tag{5}$$

where $P(C_k)$ is the prior probability of class C_k and $P(x_i|C_k)$ is the likelyhood of feature x_i given class C_k. For Multinomial Naive Bayes, the likelihood $P(x_i|C_k)$ is calculated using:

$$P(x_i|C_k) = \frac{N_{ik} + \alpha}{N_k + \alpha n} \tag{6}$$

where N_{ik} is the count of feature x_i in class C_k, N_k is the total count of all features in class C_k, α is the smoothing parameter (Laplace smoothing), and n is the number of unique features. Multinomial Naive Bayes offers several advantages, however, the assumption of conditional independence between features is often unrealistic, which can affect the classifier's performance. Furthermore, Multinomial Naive Bayes can assign zero probability to unseen words in the training data, though this issue can be mitigated using techniques such as Laplace smoothing.

3.3.2. Performance Metrics

The evaluation of machine learning models involves splitting the dataset into training and test sets to ensure a fair assessment of the model's generalization ability. Several performance metrics are used to evaluate the models.

Accuracy is the proportion of correctly classified documents out of the total number of documents and is calculated as:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{7}$$

where TP is True Positives, TN is True Negatives, FP is False Positives, and FN is False Negatives.

Precision is the proportion of true positive predictions out of all positive predictions made by the model and is calculated as:

$$Precision = \frac{TP}{TP + FP} \tag{8}$$

Recall is the proportion of true positive predictions out of all actual positive instances in the dataset and is calculated as:

$$Recall = \frac{TP}{TP + FN} \tag{9}$$

F1-Score is the harmonic mean of precision and recall, providing a single metric that balances both aspects, and is calculated as:

$$F1\text{-}Score = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{10}$$

These metrics provide a comprehensive evaluation of the model's performance, highlighting their strengths and weaknesses in different scenarios. The goal is to identify the model that offers the best balance of accuracy, precision, recall, and F1-score for document classification tasks.

3.4. Model Selection and Visualization

The process of selecting the best model involves processing documents, extracting text, and preprocessing it. Various text embeddings, TF-IDF, Word2Vec, Doc2Vec, and GloVe, are applied to transform the text data. The text data is then split into training and test sets. Multiple machine learning models, such as Support Vector Machines (SVM), K-Nearest Neighbors (KNN), and Multinomial Naive Bayes, are trained and evaluated using those embeddings. This section outlines the steps taken to select the best-performing combination of text embedding and model, the evaluation criteria used, and the visualizations generated to present the results.

Several performance metrics are used to evaluate the models, including accuracy, precision, recall, and F1-score. The F1-score is a single metric that balances both aspects of precision and recall.

Visualizing the performance of the models is crucial for understanding their strengths and weaknesses. Bar graphs are used to compare the performance metrics (accuracy, precision, recall, and F1-score) of each text embedding-model combination, providing a clear visual representation of how each combination performs across different metrics. This comprehensive evaluation ensures that the chosen model is not only accurate but also robust and reliable for document classification tasks.

3.5. Document Similarity

Document similarity measures provide a mathematical means to express the degree of similarity between two documents [13]. Two common methods for calculating text similarity between documents are cosine similarity and Jaccard similarity.

Cosine similarity measures the cosine of the angle between two non-zero vectors in a multidimensional space. In the context of text analysis, the vectors represent the term frequencies of words in the documents. Cosine similarity is calculated as follows:

$$Cosine\ Similarity = \frac{\vec{\boldsymbol{A}} \cdot \vec{\boldsymbol{B}}}{\| \vec{\boldsymbol{A}} \| \| \vec{\boldsymbol{B}} \|} \qquad (\ 11\ )$$

where $\vec{\boldsymbol{A}}$ and $\vec{\boldsymbol{B}}$ are the term frequency vectors of the two documents. The cosine similarity value ranges from 0 to 1, where 0 indicates no similarity and 1 indicates complete similarity.

Jaccard similarity measures the similarity between two sets by dividing the size of the intersection by the size of the union of the sets. For text analysis, the sets are the unique words in each document. Jaccard similarity is calculated as:

$$Cosine\ Similarity = \frac{|\ \boldsymbol{A} \cap \boldsymbol{B}\ |}{|\ \mathbf{A} \cup \mathbf{B}\ |} \qquad (\ 12\ )$$

where $\boldsymbol{A}$ and $\boldsymbol{B}$ are the sets of unique words in the two documents. The Jaccard similarity value ranges from 0 to 1, where 0 indicates no similarity and 1 indicates complete similarity.

Both methods provide a numerical value indicating the level of similarity between two documents, which can be used for various applications such as document clustering, duplicate detection, and information retrieval.

4. Implementation

4.1. Tools and Libraries Used

This section lists the tools and libraries employed in the implementation of the document analysis and comparison system. Each tool and library was chosen based on its relevance and effectiveness in handling specific tasks within the system.

- **Python**: The primary programming language used for its simplicity and extensive libraries.
- **scikit-learn**: Used for implementing machine learning models, providing tools for model training, evaluation, and selection.
- **NLTK (Natural Language Toolkit)**: Utilized for text preprocessing tasks, such as tokenization, stop word removal, and lemmatization.
- **spaCy**: Employed for advanced text preprocessing and NLP tasks.
- **PyMuPDF (fitz)**: Used for extracting text from PDF documents due to its efficient and reliable parsing capabilities.
- **python-docx**: Utilized for extracting text from DOCX files, facilitating the handling of this specific document format.
- **joblib**: Used for saving and loading machine learning models and vectorizers, enabling efficient model deployment and reuse.
- **Matplotlib**: Employed for generating visualizations, such as bar graphs, to compare model performance metrics.
- **NumPy**: Used for numerical operations, especially for handling arrays and matrices which are crucial in machine learning and data processing tasks.
- **Pandas**: Utilized for data manipulation and analysis, providing data structures and functions needed to work with structured data seamlessly.
- **SciPy**: Used for scientific and technical computing, particularly in tasks involving optimization and integration which may support machine learning algorithms.
- **Gensim**: Employed for text embedding, specifically for implementing Word2Vec and Doc2Vec.
- **sklearn-pandas**: Simplifies the integration of pandas with scikit-learn, allowing for seamless data preparation and model training.

These tools and libraries were chosen for their robustness, ease of use, and ability to efficiently handle the specific requirements of the document analysis and comparison tasks.

4.2. Document Analysis and Comparison

The document analysis and comparison process involves utilizing a pre-trained machine learning model and custom scripts to classify and compare new documents. This section outlines the methodology employed to load and use the pre-trained model for analyzing and comparing new documents, detailing the steps taken to ensure accurate predictions and comprehensive comparison capabilities.

4.2.1. Document Analysis

The initial step in the document analysis process is to load the pre-trained machine learning model and the corresponding vectorizer. These components are essential for transforming the text data from new documents into a format that the model can process.

Text is extracted from new documents using specialized functions tailored to different file formats, specifically DOCX and PDF files. This ensures that the textual content from various document formats is accurately extracted and prepared for further analysis. At this step, the document is also analyzed to determine the page count, word count, character count, and paragraph count, while tables and images are extracted as well.

Once the text is extracted, it undergoes preprocessing to ensure consistency and compatibility with the trained model. This preprocessing includes converting text to lowercase, removing special characters, tokenizing, removing stop words, and lemmatizing.

The preprocessed text is then transformed using the loaded vectorizer and fed into the trained model to make predictions. The model outputs the predicted class label for the document, indicating its category.

4.2.2. Document Comparison

In addition to classifying individual documents, the system can compare two documents to identify similarities and differences. This comparison involves analyzing various aspects of the documents to provide a comprehensive comparison.

Text is extracted from both documents using the same methods mentioned above and preprocessed to ensure consistency. The comparison process includes analyzing and comparing page count, word count, character count, and paragraph count. Additionally, tables and images from the documents are extracted. The preprocessed text from both documents is transformed into vectors using the loaded vectorizer. Both documents are classified using the loaded model. The system identifies and compares the top 10 repeating words in each document, providing insights into the content and focus of each document. Document similarity is also calculated using cosine similarity and Jaccard similarity.

The output of the document comparison script includes the detailed analysis of both documents and their similarity score. Below is an example of an output.

```
File 1 (11797122.pdf):
Page Count: 2
Word Count: 614
Character Count: 4113
Paragraph Count: 77

File 2 (13964744.pdf):
Page Count: 2
Word Count: 702
Character Count (excluding spaces): 4547
Paragraph Count: 73

Top 10 common words in file 1 (11797122.pdf) and file 2 (13964744.pdf):
Word                 Count
------------------------------
project              25
business             19
management           19
process              16
customer             16
team                 14
requirement          13
company              12
skill                11
city                 10

Classification of file 1 (11797122.pdf): ['AVIATION']
Classification of file 2 (13964744.pdf): ['INFORMATION-TECHNOLOGY']

Cosine Similarity of files 1 and 2:  0.09108626689613666
Jaccard similarity of files 1 and 2:  0.10401188707280833
```

Code 1: Example of Document Comparison Output

5. Results and Discussion

5.1. Performance Evaluation

In this section, the evaluation results of various model-embedding combinations used for document classification are presented. The performance metrics considered include accuracy (Figure 9), precision (Figure 10), recall (Figure 11), and F1-score (Figure 12). These metrics provide a comprehensive assessment of each model's effectiveness in classifying documents. The models and embeddings evaluated are Support Vector Machines (SVM), K-Nearest Neighbors (KNN), and Multinomial Naive Bayes as the models. The embeddings utilized were TF-IDF, Word2Vec, Doc2Vec, and GloVe.

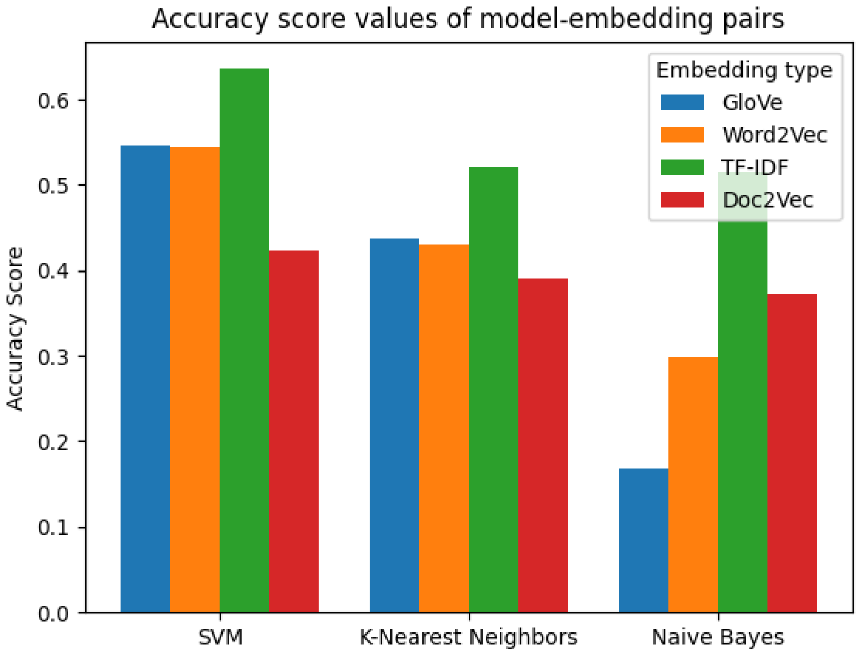


Figure 9: Accuracy Score Values of Model – Embedding Pairs

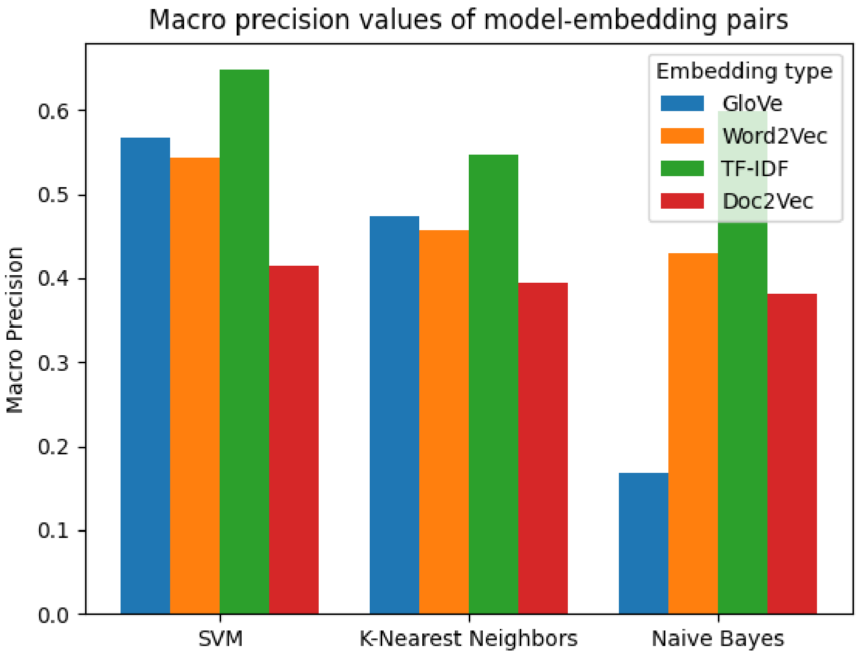


Figure 10: Macro Precision Values of Model – Embedding Pairs

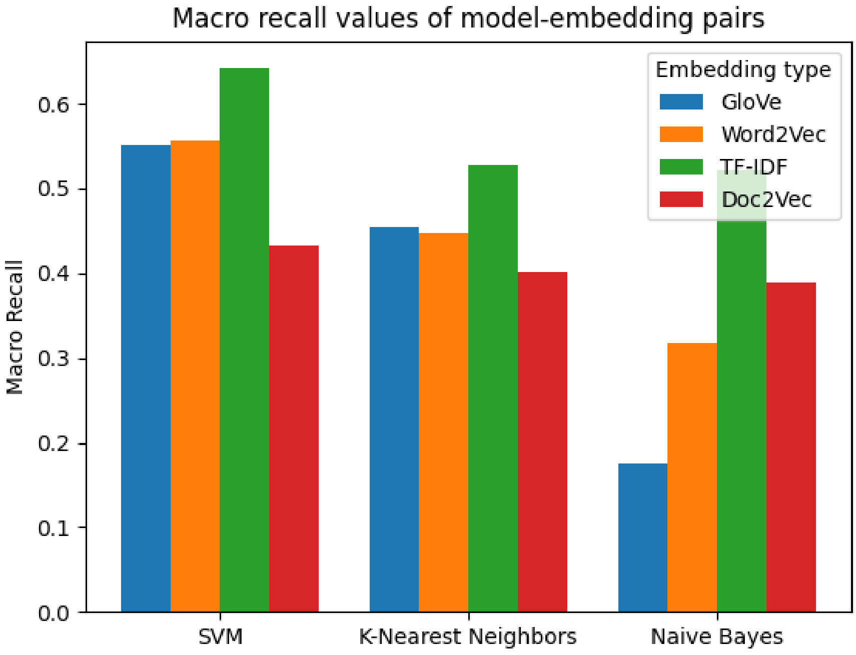


Figure 11: Macro Recall Values of Model – Embedding Pairs

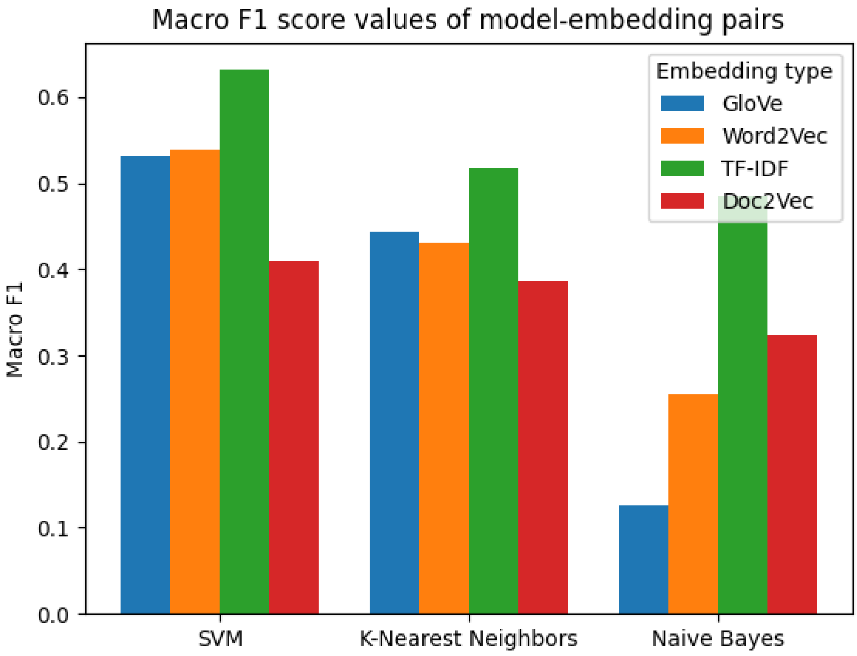


Figure 12: Macro F1-Score Values of Model – Embedding Pairs

In addition to the visual representation, the table below provides the exact numerical values for each performance metric, allowing for detailed reference and analysis.

Model	Embedding	Accuracy	Precision	Recall	F1-Score
Linear SVM	TF-IDF	**0.635802**	**0.647919**	**0.641813**	**0.63165**
Linear SVM	Word2Vec	0.54321	0.542795	0.557268	0.538864
Linear SVM	Doc2Vec	0.423868	0.414886	0.431807	0.408888
Linear SVM	GloVe	0.545267	0.567632	0.551279	0.531971
KNN (k=5)	TF-IDF	0.520576	0.547476	0.527511	0.516946
KNN (k=5)	Word2Vec	0.430041	0.456592	0.44647	0.430842
KNN (k=5)	Doc2Vec	0.390947	0.394836	0.401276	0.385853
KNN (k=5)	GloVe	0.438272	0.473993	0.455036	0.444058
Multinomial Naive Bayes	TF-IDF	0.514403	0.597739	0.521421	0.484213

Multinomial Naive Bayes	Word2Vec	0.298354	0.428828	0.316919	0.255162
Multinomial Naive Bayes	Doc2Vec	0.372428	0.381196	0.389138	0.324203
Multinomial Naive Bayes	GloVe	0.168724	0.167814	0.175714	0.125685

Table 1: Performance Metrics

The performance of the best-performing model-embedding combination (SVM with TF-IDF) was evaluated across various metrics. The class-wise F1 scores, precision, and recall for this combination are presented in the following figures.

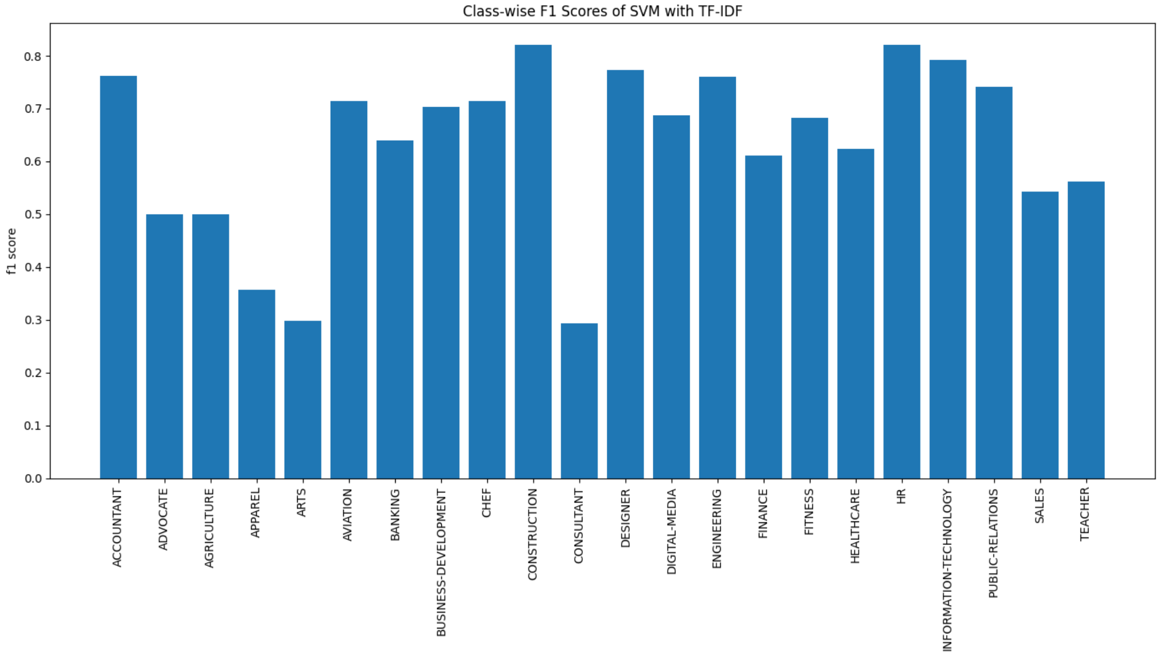


Figure 13: Class-wise F1 Scores of SVM with TF-IDF

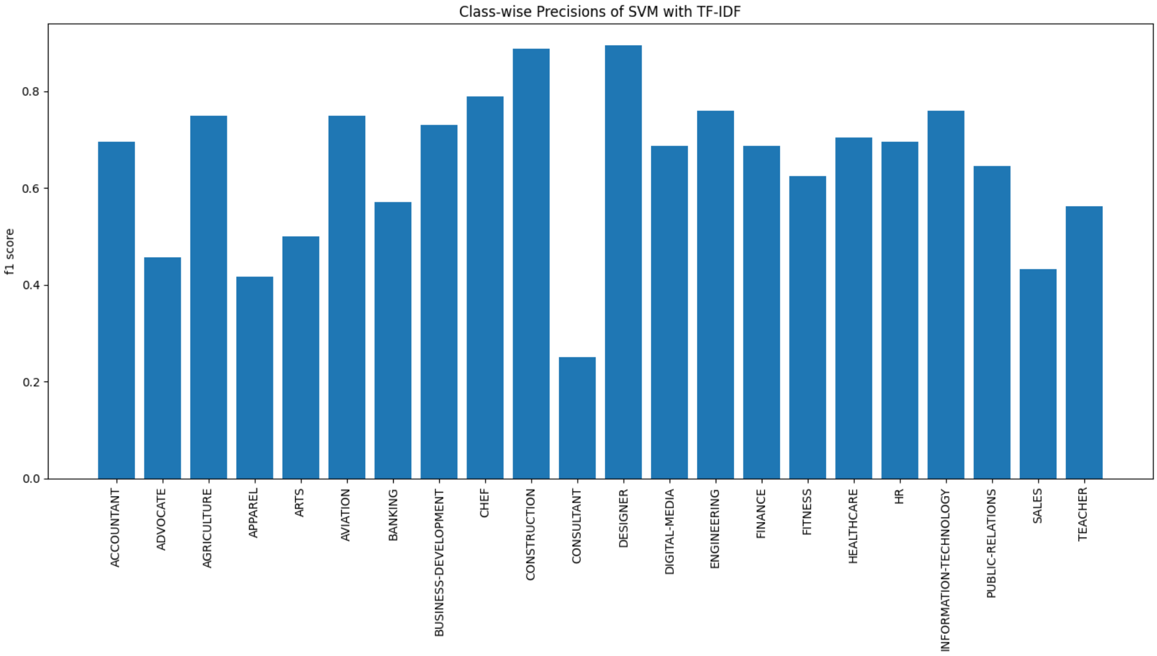


Figure 14: Class-wise Precisions of SVM with TF-IDF

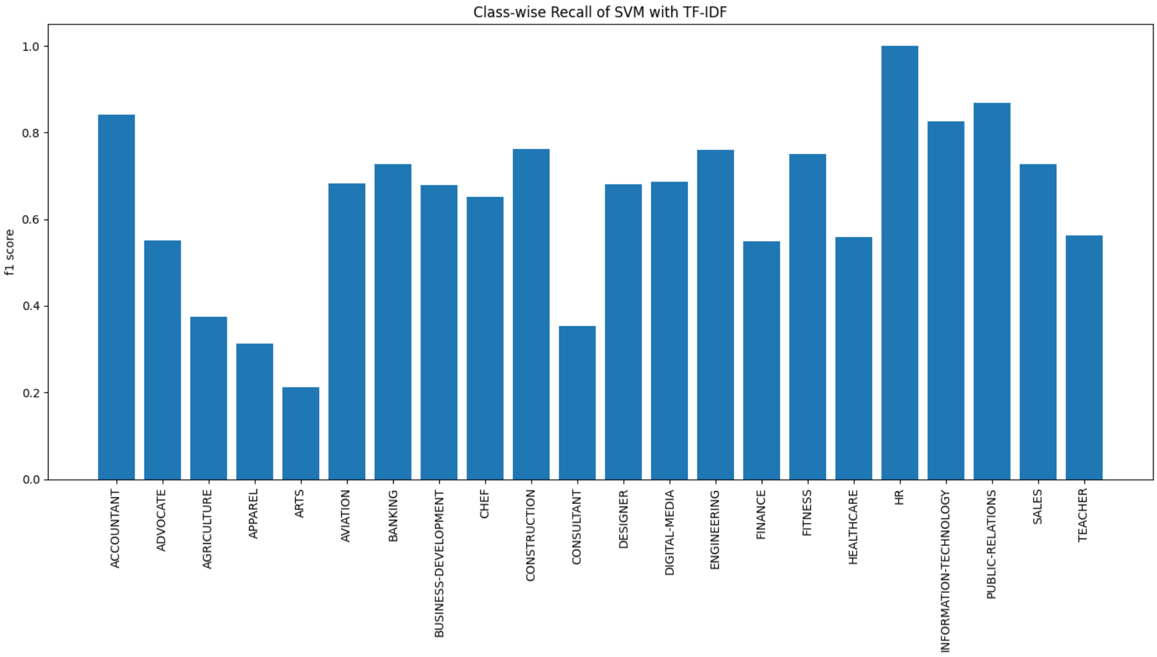


Figure 15: Class-wise Recall of SVM with TF-IDF

These graphs illustrate the model's performance across different document classes (job categories), providing insights into its strengths and weaknesses in specific areas. For instance, the model achieved the highest F1 scores in categories such as CONSTRUCTION, DESIGNER, HR and INFORMATION-TECHNOLOGY, indicating robust performance.

However, in categories such as APPAREL, ARTS and CONSULTANT, the performance was relatively lower, suggesting areas for further improvement.

5.2. Comparative Analysis Results

In this section, the performance of various models and embeddings is compared, discussing why certain models performed better than others.

5.2.1. Comparison of Model Performance

The SVM models, particularly with the linear kernel, consistently outperformed other models. This is likely due to the ability of SVM to handle high-dimensional data effectively. It is indicated that linear decision boundaries are effective for this classification task.

KNN models, which classify documents based on the majority class among the nearest neighbors, showed lower performance. This suggests that the local neighborhood structure of the data is less informative for this classification task, possibly due to the high dimensionality of the text embeddings.

Multinomial Naive Bayes, while generally effective for text classification, performed lower than SVM models. This could be due to the independence assumptions made by Naive Bayes, which may not hold true for the text data in this particular dataset.

5.2.2. Comparison of Embedding Techniques

TF-IDF consistently resulted in higher performance across all models. This is likely because TF-IDF captures the importance of words in the documents relative to the corpus, providing a good balance between term frequency and document frequency.

Word2Vec and GloVe, which generate dense vector representations capturing semantic relationships between words, also performed well, particularly with SVM models. These embeddings are effective in capturing the contextual meaning of words, which is beneficial for document classification.

Doc2Vec, which extends Word2Vec to generate document-level embeddings, showed slightly lower performance. This could be due to the complexity of capturing the overall context of a document, which may require more training data or different hyperparameters for optimal performance.

5.2.3. Discussion

The superior performance of SVM models, particularly with TF-IDF and linear kernel, indicates that these models are well-suited for high-dimensional text classification tasks. The ability of SVM to create effective decision boundaries likely contributes to its effectiveness.

The lower performance of KNN models suggests that they may struggle with high-dimensional data, where the notion of distance becomes less meaningful. This is a known challenge with KNN in high-dimensional spaces.

Multinomial Naive Bayes' lower performance highlights the limitations of the independence assumption in text data. While it is computationally efficient and works well with large vocabularies, it may not capture the dependencies between words as effectively as other models.

Overall, the comparative analysis demonstrates that the choice of model and embedding technique significantly impacts the performance of document classification systems. SVM models with TF-IDF and semantic embeddings (Doc2Vec, GloVe) provide a robust solution for this task, balancing precision, recall, and overall accuracy.

Conclusion

This thesis has explored the implementation and evaluation of a comprehensive system for document analysis and comparison. The primary findings of this research can be summarized as follows: A robust system was developed to handle various document formats, including PDF and DOCX, for text extraction, preprocessing, and analysis. The system incorporated different text embedding techniques—TF-IDF, Word2Vec, Doc2Vec, and GloVe—and machine learning models—Linear SVM, K-Nearest Neighbors (KNN), and Multinomial Naive Bayes. Among the various model-embedding combinations evaluated, the Linear SVM model with TF-IDF embedding achieved the highest performance metrics, indicating its effectiveness for document classification tasks. The detailed evaluation metrics were presented in terms of accuracy, precision, recall, and F1-score. The system also provided functionalities for comparing documents based on various metrics such as word count, page count, and identifying the top 10 repeating words. This feature demonstrated the system's capability to handle practical document analysis scenarios.

This thesis contributes significantly to the field of document analysis by developing a system that automates the extraction, preprocessing, and analysis of text from various document formats, addressing the inefficiencies and inaccuracies associated with manual document analysis. The systematic comparison of different text embedding techniques and machine learning models provides valuable insights into their performance and applicability. This analysis highlights the strengths and limitations of each approach, guiding future research and application development in document classification. The versatility of the developed system, capable of handling multiple document formats and providing robust classification and comparison features, represents a significant advancement in document analysis technology.

While this thesis has achieved its primary objectives, several areas for future research and improvement have been identified. Future work could involve the optimization of hyperparameters for the machine learning models to further improve classification accuracy and performance. Exploring the use of advanced deep learning models, such as transformer-based architectures (e.g., BERT, GPT), could enhance the system's ability to understand and process complex textual data. Extending the system to support additional document formats, such as HTML and scanned images, would broaden its applicability and usefulness.

Developing more sophisticated document comparison features, including semantic similarity and content summarization, could provide deeper insights and more detailed analyses.

In conclusion, this thesis has laid a strong foundation for automated document analysis and comparison. The findings and contributions presented herein offer valuable insights and pave the way for future advancements in this critical field. The proposed areas for future work suggest promising directions for further research and development, aimed at enhancing the system's capabilities and expanding its application scope.

References

[1] Bhawal, S. Resume Dataset. *Kaggle*. Available at: https://www.kaggle.com/datasets/snehaanbhawal/resume-dataset. Accessed June 5th, 2024

[2] Jurafsky, D., Martin, J. H. *Speech and Language Processing*. 3rd ed. London: Pearson, 2019.

[3] Manning, C. D., Raghavan, P., Schütze, H. *Introduction to Information Retrieval*. Cambridge: Cambridge University Press, 2008.

[4] Scikit-learn: Machine Learning in Python. Available at: https://scikit-learn.org/stable/

[5] Natural Language Toolkit (NLTK). Available at: https://www.nltk.org/

[6] SpaCy: Industrial-strength Natural Language Processing. Available at: https://spacy.io/

[7] Bird, S., Klein, E., Loper, E. *Natural Language Processing with Python*. Sebastopol: O'Reilly Media, 2009.

[8] Honnibal, M., Montani, I. spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing, 2017. Available at: https://spacy.io

[9] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... & Duchesnay, É. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12 (2011), pp. 2825-2830.

[10] PyMuPDF Documentation. Available at: https://pymupdf.readthedocs.io/en/latest/

[11] Larhmam, Maximum-Margin Hyperplane and Margins for an SVM Trained with Samples from Two Classes, CC BY-SA 4.0, Available at: https://commons.wikimedia.org/w/index.php?curid=73710028

[12] Antti Ajanki AnAj, Example of k-NN Classification, CC BY-SA 3.0, Available at: https://commons.wikimedia.org/w/index.php?curid=2170282

[13] Kocaman, A. *How to Measure Text Similarity: A Comprehensive Guide*. Medium, (2020, August). Available at: https://medium.com/@ahmetmnirkocaman/how-to-measure-text-similarity-a-comprehensive-guide-6c6f24fc01fe [Accessed June 2024].

[14] *Topic Modeling with Gensim (Python)*. Machine Learning Plus, (2018, March). Available at: https://www.machinelearningplus.com/nlp/topic-modeling-gensim-python/ [Accessed June 2024].

Summary

Implementation of Document Analysis and Comparison Functionalities

This thesis develops an automated system for text recognition, information extraction, document classification and document comparison. The system handles various document formats and uses text embeddings (TF-IDF, Word2Vec, Doc2Vec, GloVe) and machine learning models (Linear SVM, K-Nearest Neighbors, Multinomial Naive Bayes) to enhance classification accuracy. Evaluated on a Kaggle-sourced resume dataset with 22 classes, the Linear SVM with TF-IDF achieved the best performance. This work offers insights for future advancements in automated document analysis.

Keywords

Document Analysis, Text Recognition, Information Extraction, Document Classification, Python, Text Embeddings, Machine Learning, TF-IDF, Word2Vec, Doc2Vec, GloVe, Linear SVM, K-Nearest Neighbors, Multinomial Naive Bayes, Automated Document Processing

Sažetak

Implementacija funkcionalnosti analize i usporedbe dokumenata

Ovaj rad bavi se razvojom automatiziranog sustava za prepoznavanje teksta, ekstrakciju informacija, klasifikaciju i usporedbu dokumenata. Sustav obrađuje različite formate dokumenata i koristi tekstualne ugradnje (TF-IDF, Word2Vec, Doc2Vec, GloVe) i modele strojnog učenja (Linear SVM, K-Nearest Neighbors, Multinomial Naive Bayes) za poboljšanje točnosti klasifikacije. Evaluiran na skupu podataka sačinjenog od životopisa s Kaggle-a koji sadrži 22 klase, linearni SVM s TF-IDF-om postigao je najbolje rezultate. Ovaj rad nudi uvid u buduća unapređenja automatizirane analize dokumenata.

Ključne riječi

analiza dokumenata, prepoznavanje teksta, ekstrakcija informacija, klasifikacija dokumenata, tekstualne ugradnje, strojno učenje, TF-IDF, Word2Vec, Doc2Vec, GloVe, linearni stroj s potpornim vektorima, K-najbliži susjedi, multinomijalni naivni Bajesov klasifikator, automatizirana obrada dokumenata

Appendices

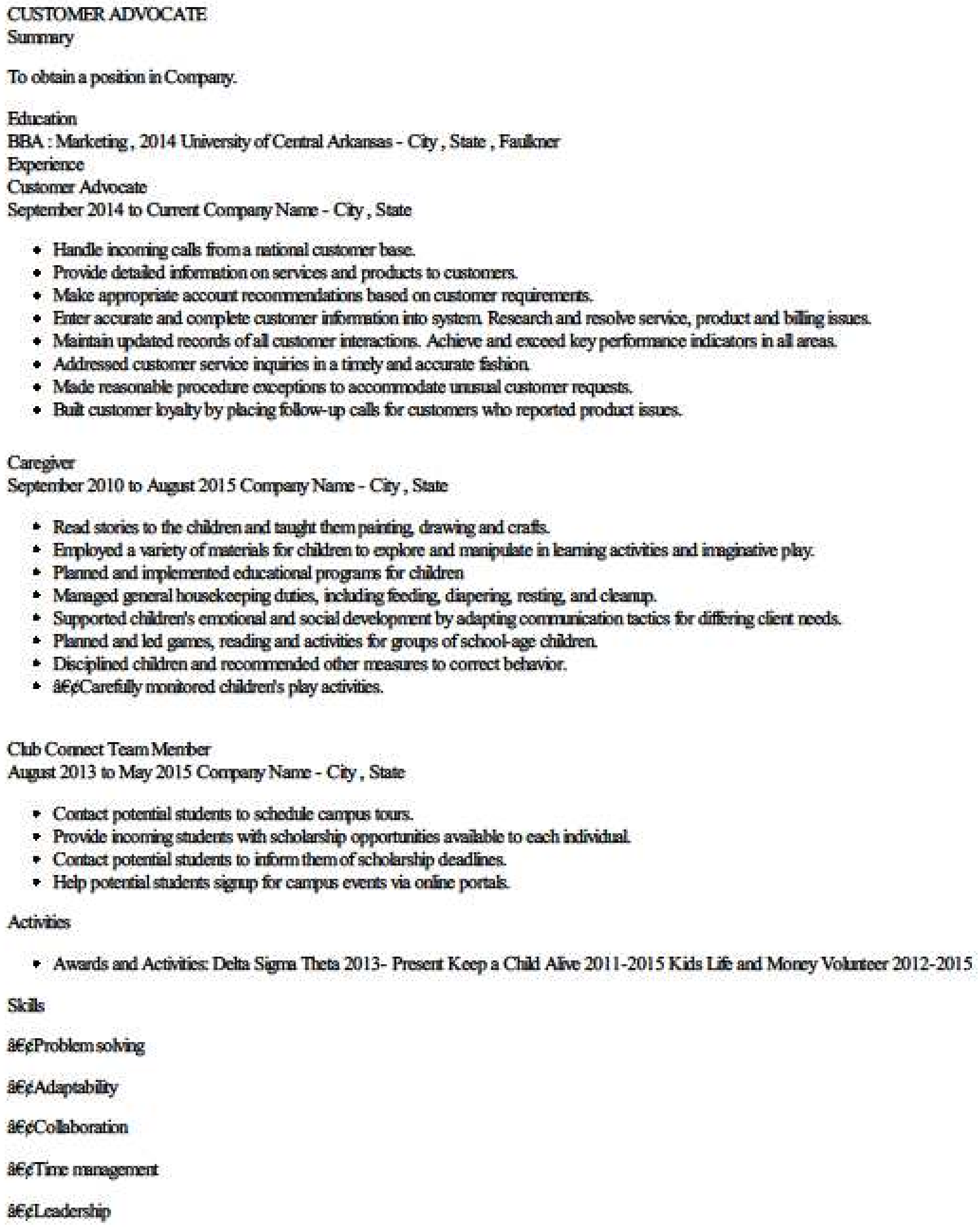
Appendix A: GitHub Repository

The code used for the implementation of the document analysis and comparison system can be found in the following GitHub repository:

https://github.com/zagajski/DocumentAnalysis

Appendix B: Document Example

This is an example of a document from



CUSTOMER ADVOCATE
Summary

To obtain a position in Company.

Education
BBA : Marketing , 2014 University of Central Arkansas - City , State , Faulkner
Experience
Customer Advocate
September 2014 to Current Company Name - City , State

- Handle incoming calls from a national customer base.
- Provide detailed information on services and products to customers.
- Make appropriate account recommendations based on customer requirements.
- Enter accurate and complete customer information into system. Research and resolve service, product and billing issues.
- Maintain updated records of all customer interactions. Achieve and exceed key performance indicators in all areas.
- Addressed customer service inquiries in a timely and accurate fashion.
- Made reasonable procedure exceptions to accommodate unusual customer requests.
- Built customer loyalty by placing follow-up calls for customers who reported product issues.

Caregiver
September 2010 to August 2015 Company Name - City , State

- Read stories to the children and taught them painting, drawing and crafts.
- Employed a variety of materials for children to explore and manipulate in learning activities and imaginative play.
- Planned and implemented educational programs for children
- Managed general housekeeping duties, including feeding, diapering, resting, and cleanup.
- Supported children's emotional and social development by adapting communication tactics for differing client needs.
- Planned and led games, reading and activities for groups of school-age children.
- Disciplined children and recommended other measures to correct behavior.
- â€¢Carefully monitored children's play activities.

Club Connect Team Member
August 2013 to May 2015 Company Name - City , State

- Contact potential students to schedule campus tours.
- Provide incoming students with scholarship opportunities available to each individual.
- Contact potential students to inform them of scholarship deadlines.
- Help potential students signup for campus events via online portals.

Activities

- Awards and Activities: Delta Sigma Theta 2013- Present Keep a Child Alive 2011-2015 Kids Life and Money Volunteer 2012-2015

Skills

â€¢Problem solving

â€¢Adaptability

â€¢Collaboration

â€¢Time management

â€¢Leadership