NATIONAL RESEARCH UNIVERSITY
HIGHER SCHOOL OF ECONOMICS

Faculty of Computer Science
Bachelor's Programme "Applied Mathematics and Informatics"

Software Project Report on the Topic:

Digital Interview Analysis (Linguistics): Identifying Motivational Patterns: Approach – Avoidance in English

Submitted by the Student:

group #БПАД231, 2nd year of study   Shub Sergey Alekseevich

Approved by the Project Supervisor:

Perederin Denis Alexandrovich
Visiting Lecturer, Associate Professor
School of Oriental Studies, Faculty of World Economy and International Relations, HSE

Moscow 2025

Contents

Annotation

This project involves developing a software service for the automatic analysis of English-language interview transcripts to identify approach–avoidance motivational patterns. Its relevance stems from the need for efficient processing of textual data in HR, psychology, and marketing and from the ability to compare motivational trends in Eastern and Western cultures, highlighting the importance of cross-cultural analysis. The goal is to create a tool that converts unstructured texts into reports with highlighted segments showing motivation to achieve (approach) or to avoid failure (avoidance). The methodology is based on an interpretable NLP approach (tokenization, lemmatization, keyword lists, negation handling) and does not require large annotated datasets. The service is implemented in Python, accepts files (docx, txt, pdf), and generates CSV reports and visualizations, as well as a module for pairwise comparison of interviews to identify differences in motivational profiles.

Аннотация

Данный проект посвящен разработке программного сервиса для автоматического анализа текстовых транскриптов англоязычных интервью с целью выявления мотивационных моделей «подход–избегание». Актуальность обусловлена потребностью в эффективной обработке текстовых данных в HR, психологии и маркетинге и возможностью сравнивать мотивационные тенденции восточных и западных культур, что усиливает значимость кросс-культурного анализа. Цель – создание инструмента, преобразующего неструктурированные тексты в отчеты с выделением сегментов, демонстрирующих мотивацию достижения (подход) или избегания неудач (избегание). В основе лежит NLP-подход (токенезация, лемматизация, списки ключевых слов, учет отрицаний), ориентированный на интерпретируемость и не требующий больших размеченных датасетов. Сервис реализован на Python, принимает различные форматы текстов (docx, txt, pdf), генерирует CSV-отчеты и визуализации, а также реализуется модуль парного сравнения интервью для выявления различий в мотивационных профилях.

Keywords

Interview analysis, natural language processing, NLP, motivational patterns, approach–avoidance standardized PDF transcripts, rule-based systems, HR analytics, cross-cultural studies.

Introduction

Understanding human motivation is a fundamental task in psychology, sociology, and business. In personnel selection, special attention is paid not only to the objective assessment of professional competencies but also to uncovering deep motivational patterns. Among the dichotomies under study are Internal–External Reference, Proactivity–Reflectiveness, Goal-Oriented–Process-Oriented Thinking, Procedures–Possibilities, Similarity–Difference, General–Specific Thinking, and Time Orientation (Past–Present–Future).

This project is devoted to the development of a software service for analyzing the Approach–Avoidance motivation dichotomy [**1**, **2**] in English-language interview transcripts. This dichotomy divides motivation into the drive toward positive outcomes (approach) and the effort to avoid negative outcomes (avoidance). Identifying the dominant tendency in a candidate's speech enables HR professionals, psychologists, and researchers to better understand whether an individual is inclined to take initiative and embrace risk or prefers to avoid uncertainty and potential losses.

Automated text analysis of interviews is a promising direction, as manual annotation of large datasets is time-consuming and subjective. NLP tools [3] based on lemmatization, keyword patterns, and context and negation handling can efficiently classify statements along the Approach–Avoidance scale without requiring extensive labeled corpora, ensuring result interpretability and flexibility for further algorithm tuning.

The relevance and significance of this project lie in the service's narrow specialization on the Approach–Avoidance dichotomy, which allows for more accurate and valuable insights for HR practitioners, psychologists, and researchers. The interpretable approach, built on transparent text-processing methods, fosters trust in the service's outputs. Practically, the ready-to-deploy tool will enable rapid analysis of large numbers of interviews, uncovering motivational patterns and predicting candidate behavior under stress or uncertainty.

The project's goal is the development, implementation, and testing of a tool that accepts interview text files, extracts and preprocesses the text, classifies sentences along the Approach–Avoidance scale, and generates reports with visualizations. The novelty and practical significance of this work lie in creating a ready-to-use instrument for automating the analysis of motivational patterns, which will be highly useful for HR professionals, psychologists, and researchers.

Literature Review

There exists various commercial solutions use natural language processing (NLP) and machine learning techniques to analyze candidates' speech patterns. Below are some of the most well-known systems and their characteristics:

HireVue:

- Uses machine learning technologies to assess candidates based on video interviews.
- Analyzes both verbal (lexical choice, syntax) and non-verbal (intonation, facial expressions) aspects of speech.
- Evaluates a wide range of parameters, including emotional tone and candidate confidence.
- **Drawback:** Operates as a black-box system—HR specialists may not fully understand how a candidate's score is determined.

Pymetrics:

- Utilizes game-based assessments and neuropsychological models to evaluate candidates' cognitive and emotional traits.
- Analyzes behavioral tendencies such as risk-taking and motivation levels.
- Uses machine learning to match candidates with company culture.
- **Drawback:** Focuses on behavioral analysis rather than speech patterns and requires additional game-based testing.

TextRecruit:

- Automates candidate interactions through chatbots powered by NLP.
- Allows HR professionals to analyze text-based communication for emotional tone, lexical choices, and overall response sentiment.
- Aims to streamline communication and accelerate the hiring process.
- **Drawback:** Does not provide in-depth assessment of motivational speech patterns.

Key Limitations of Existing Solutions:

- **Lack of transparency** – HR specialists receive final assessments without insight into the decision-making process.
- **Broad focus** – Most systems analyze a wide range of characteristics but do not specifically target motivational speech patterns, such as the "Approach-Avoidance" dichotomy.
- **Dependence on large datasets** – Many solutions require extensive datasets for model training, making them less suitable for specialized applications.

Advantages of the Proposed Project:

- **Focus on motivational tendencies** – The analysis of "Approach-Avoidance" patterns provides a deeper understanding of candidates' motivations.
- **Transparent methodology** – A rule-based approach ensures clarity in decision-making.
- **Independence from large datasets** – The model does not require extensive pre-training, making it easier to implement.

1. "Approach-Avoidance" motivation and its analysis using NLP: a theoretical overview

This chapter examines the theoretical foundations of Approach–Avoidance motivation, relevant natural language processing (NLP) methods for text analysis, and the positioning of this work among existing solutions.

1.1 Psychological foundations of motivational models

The "Approach–Avoidance" motivational concept posits that human behavior is driven either by the pursuit of positive outcomes (approach) or by the effort to avoid negative outcomes (avoidance) [**1**]. Andrew Elliot et al. expanded this theory, demonstrating links between these orientations and various cognitive and behavioral consequences [**2**]. Linguistic markers in speech (such as "achieve" or "grow" for approach, and "avoid" or "problem" for avoidance) can serve as indicators of a respondent's dominant motivational orientation [4]. Analyzing these markers enables an initial profiling of an individual's motivational tendencies.

1.2 NLP Methods for Text Analysis

Various natural language processing (NLP) methods are applied to identify motivational patterns in text [**3**, **5**]:

- **Tokenization** — splitting text into individual words or sentences.
- **Text cleaning** — removing punctuation, converting text to lowercase, and eliminating stop words.
- **Lemmatization** — reducing words to their base dictionary form (e.g., *running* $\rightarrow$ *run*). Preferred over stemming due to greater semantic accuracy.
- **Dictionary-based (lexicon) analysis** — searching the text for predefined keywords associated with the categories of approach and avoidance. The effectiveness of this method depends on the quality of the lexicons used.
- **Sentiment analysis** — determining the emotional tone of the text. This method does not always capture the nuances of approach–avoidance motivation, as avoidance can be expressed using neutral language.

- **Machine learning (ML) methods** — such as support vector machines (SVM) and neural network models like BERT [9]. These approaches can achieve high accuracy but require large annotated datasets and are often less interpretable.

This project adopts a hybrid approach that combines tokenization, cleaning, lemmatization, dictionary-based analysis, and explicit handling of negation. This allows for a balance between implementation simplicity, result transparency, and independence from large datasets.

1.3 Positioning of this project

There are numerous NLP libraries (such as NLTK [6], spaCy [7], and TextBlob [8]) as well as commercial platforms for text analysis. However, there are very few specialized solutions designed to detect approach–avoidance motivation using interpretable methods. Most existing tools either focus on general sentiment analysis or rely on machine learning models [9].

This project fills a specific niche by offering:

- **Specificity**: a clear focus on the approach–avoidance dichotomy.
- **Interpretability**: a keyword-based method that ensures transparency of results.
- **Low data requirements**: does not require training on large annotated corpora.
- **Ready-to-use service**: implemented as a command-line application that automates the entire analysis pipeline.

The work provides a practical solution to a well-defined problem, based on well-established NLP techniques but adapted and delivered as a convenient software product.

2. Motivation Analysis System Design

This chapter describes the key design decisions: the formal problem statement, rationale for the chosen approach, system architecture, and technology stack.

2.1 Problem Statement and Requirements

The objective is to develop a software service that supports two complementary modes of operation—single-file analysis and pairwise comparison—designed to extract and compare motivational patterns in interview texts.

Single-file analysis

For a given set of interview text files D and keyword lists $K_{\text{approach}}, K_{\text{avoidance}}, K_{\text{negation}}$, performs the following steps for each file: extracts the text T_i, segments it into sentences P_i, and for each sentence $p_{i,j}$, performs preprocessing (producing lemmas $L_{i,j}$), identifies detected approach and avoidance keywords $F_{\text{approach}}, F_{\text{avoidance}}$, detects negated keywords $N_{i,j}$, forms sets of non-negated keywords $F'_{\text{approach}}, F'_{\text{avoidance}}$, and assigns a label to the sentence $C_{i,j} \in \{\text{Approach, Avoidance, Mixed, Neutr}$ according to the following rules:

$$\begin{aligned}
&\text{Mixed} \quad \text{if} \quad F'_{\text{approach}} \neq \emptyset \quad \text{and} \quad F'_{\text{avoidance}} \neq \emptyset;\\
&\text{Approach} \quad \text{if} \quad F'_{\text{approach}} \neq \emptyset \quad \text{and} \quad F'_{\text{avoidance}} = \emptyset;\\
&\text{Avoidance} \quad \text{if} \quad F'_{\text{approach}} = \emptyset \quad \text{and} \quad F'_{\text{avoidance}} \neq \emptyset;\\
&\text{Neutral} \quad \text{if} \quad F'_{\text{approach}} = \emptyset \quad \text{and} \quad F'_{\text{avoidance}} = \emptyset.
\end{aligned}$$

The service must generate a CSV report R_i with the classified sentences and keywords, as well as optional visualizations V_i.

Pairwise comparison mode

Given an ordered pair of interview files D_1 and D_2, the system performs a comparative analysis to highlight differences in motivational orientation. Each file undergoes single-file analysis, producing individual CSV reports R_1 and R_2, along with optional visualizations V_1 and V_2. These visual materials are displayed side by side to facilitate intuitive, direct comparison of motivational trends and patterns across the two interviews.

2.2 Chosen Analysis Approach

A keyword-list-based approach with lemmatization, stop-word removal, and heuristics for negation handling was selected. Justification:

- **Interpretability**: results are easy to understand based on visible keywords.
- **Data requirements**: no need for large annotated training corpora.
- **Controllability**: keyword lists are easy to modify.
- **Efficiency**: less resource-intensive than ML models.
- **Alignment with goals**: relies on fundamental NLP methods.

Limitations: sensitivity to lexicon quality, lack of deep contextual understanding, and limited negation handling. However, for a basic and interpretable tool, this approach is optimal.

2.3 System Architecture and Technology Stack

The service is implemented as a command-line Python application with a pipeline-based architecture.

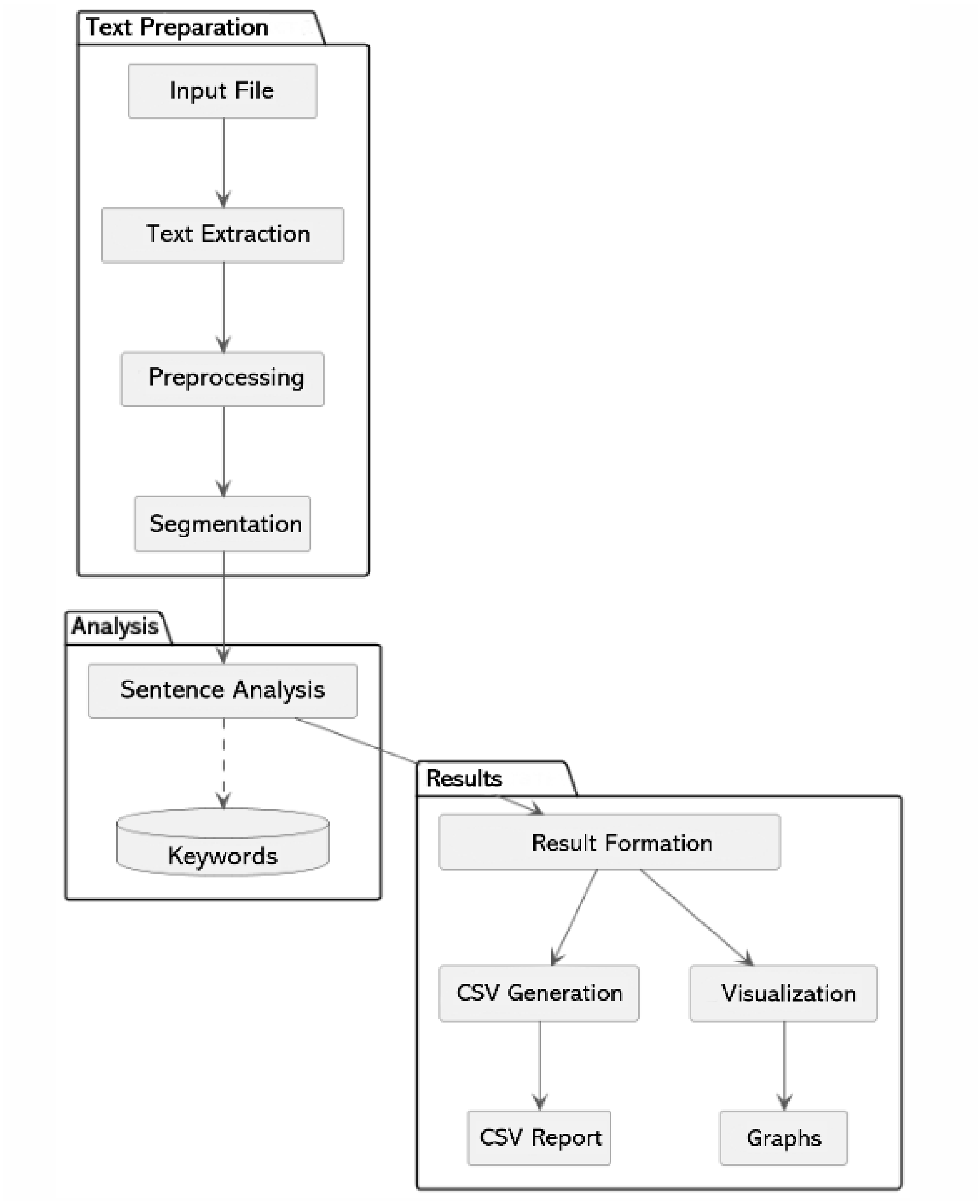

Figure 2.1: System architecture of the motivation analysis service for a single file

2.4 Main Components and Library Selection

The system includes the following main components: a text extraction module (python-docx, PyMuPDF), a preprocessing module (NLTK: tokenization, lemmatization, stop-word removal), a segmentation module (NLTK), a sentence analysis module (keyword matching using `set`, negation heuristics), a result aggregation module (pandas), a CSV report generation modules (pandas), and a visualization module (Matplotlib, Seaborn).

Library selection:

- **Text extraction:** `python-docx` [10] for `.docx`, PyMuPDF (`fitz`) [11] for `.pdf`, built-in functions for `.txt`.

- **Preprocessing:** NLTK [6] (tokenizers: `punkt`, `word_tokenize`; `stopwords`; `WordNetLemmatizer`
- **Classification:** keyword sets implemented via `set`; sliding window heuristics for negation detection.
- **Reporting and visualization:** `pandas` [12] for CSV export; `Matplotlib` [13] and `Seaborn` [14] for graphs; `collections.Counter` for frequency counting.

3. Service Implementation and User Interaction

This chapter presents a detailed overview of the practical implementation of the developed software service. We describe the general structure of its source code, outline the key software components and functions that implement the core logic of the analysis, and provide a typical usage scenario from the preparation of input data to the interpretation of the results.

3.1 Code Structure and Description of Key Components

The software service for automated analysis of motivational patterns in interview texts is implemented as a single main script written in Python. Although the entire code is contained in one file, it has a clear logical structure, divided into several functional blocks. This organization ensures not only the integrity of the solution but also contributes to better readability, understanding, and potential modifiability of the code. Structurally, the script includes the following main logical blocks:

1 **Library import block**: at the beginning of the script, all external libraries are imported. In addition, standard Python modules are also included, such as `os` for interacting with the file system (working with paths, checking for file existence), `re` for regular expressions (e.g., during text cleaning), `string` for working with punctuation sets, and `collections` (specifically, the `Counter` class for efficient frequency counting).

2 **NLTK resource initialization block**: before the main execution begins, the script initializes required resources from the NLTK library. It checks whether necessary corpora and models are available and downloads them if missing. These resources include the sentence tokenizer `punkt` (for proper sentence segmentation), the `stopwords` list (for removing frequent non-informative words), the lexical database `wordnet` (used by the lemmatizer to reduce words to their base form), and the multilingual package `omw-1.4` (Open Multilingual Wordnet), which is also needed for proper lemmatizer operation across languages, although the focus of this project is on English texts.

3 **Definition of global constants and keyword lists**: this section defines all main parameters that control the behavior and logic of the system. This includes the keyword lemma lists that characterize "approach" motivation (stored in the variable `APPROACH_KEYWORDS`) and "avoidance" motivation (variable `AVOIDANCE_KEYWORDS`). A list of negation markers is also defined (variable `NEGATION_WORDS`). Other important constants may be defined here as

well, such as the size of the NEGATION_WINDOW, within which the system searches for negation markers preceding a detected keyword. At this stage, main NLP tools are also initialized, such as creating an instance of the WordNetLemmatizer and building an up-to-date set of stopwords, which can be extended with filler words or interjections specific to the interview domain.

4 **A set of functions for reading files in various formats:** To ensure flexibility when working with different types of input data, a set of specialized functions was implemented (read_text_file, read_docx_file, read_pdf_file). Each of these functions is responsible for correctly extracting the textual content from files of the corresponding format, taking into account format-specific features. To simplify usage and automate the selection of the appropriate function, an auxiliary dispatcher function called get_text_from_file was also created. This function determines the file type based on its extension and calls the appropriate reader function.

5 **The main text analysis function (analyze_motivation):** This function is the core and most complex component of the entire system. It implements the main pipeline for linguistic analysis. It accepts as input the full interview text (as a single string), segments it into individual sentences, and then performs a multi-stage preprocessing cycle on each sentence. This cycle includes tokenization (splitting into words), conversion to lowercase, lemmatization (reducing each word to its base form), and the removal of stop words and punctuation. After preprocessing, the function searches the cleaned sentence for keywords from predefined lists (APPROACH_KEYWORDS and AVOIDANCE_KEYWORDS). A custom heuristic is applied to detect and correctly handle negation constructions that can alter or invert the meaning of detected keywords. Based on the collected information (presence and type of keywords, presence of negations), the function classifies each sentence into one of four possible motivational categories. The results for each relevant (i.e., non-neutral) sentence are accumulated for the report generation stage.

6 **Report generation function (generate_report):** This function takes as input the structured analysis results produced by analyze_motivation and converts them into a tabular format using a DataFrame object from the popular pandas library. The resulting table is then saved as a CSV file at the path specified by the user.

7 **Visualization function (visualize_analysis_results):** To provide a clear and interpretable presentation of the analysis results, a visualization function was implemented. It

reads the previously generated CSV report (or, if needed, works directly with the output of the analysis function) and produces a set of informative plots and diagrams. Visualizations are created using the `Matplotlib` and `Seaborn` libraries. This step is optional for the user—the decision to generate visualizations is made at runtime.

8 **Extended visualisation function for comparative analysis: (`if __name__ == "__main__"`):** To facilitate visual comparison of the analysis results of two interviews, the system supports a mode for visual comparison of analysis results for two interviews. In this mode, paired diagrams are generated, allowing for a clear side-by-side comparison of key parameters. All charts are displayed in two columns — one for each analyzed file—facilitating interpretation of differences in motivational patterns. Each visualization is accompanied by a brief textual summary with numerical characteristics, aiding interpretation. If one of the reports is empty or lacks sufficient data, the corresponding visual elements are omitted.

9 **Main script execution block (`if __name__ == "__main__"`):** This standard Python code block serves as the main entry point when the program is executed as a standalone script. During execution, the user is prompted to enter the name of the input file (or a directory path in the case of batch processing). If desired, a second file can also be specified for comparison, in which case the system performs a parallel analysis of both interviews. Also user can choose, whether visualizations should be generated or not. The output CSV file name is generated automatically, but can be changed manually if needed. Based on the input parameters, the program then sequentially performs all key stages: text extraction, linguistic analysis, report generation, and—if selected—visualization of the results. If two files are provided, the visualization is presented in a comparative format, with the results of each interview shown in a separate column. Basic error handling is implemented to inform the user if any of the specified files are missing or unreadable.

Such a clear and logically modular code structure not only significantly simplifies the process of understanding and debugging the system, but also provides the necessary flexibility for potential future modifications, extensions, or integration with other systems. For example, it would be easy to add support for a new input file format, improve or replace the current lemmatization algorithm with a more advanced one, or even implement a completely new module for additional types of analysis—without substantially affecting other well-tested components of the system.

3.2 Key Components Implementing the Core Program Logic

Key program components that implement the core functionality and logic of the developed service are the following functions:

- `get_text_from_file(filepath)`: This utility function, taking a string parameter `filepath` that contains the full path to the file to be analyzed, first determines the file's extension (for example, `.docx`, `.pdf`, `.txt`). Depending on the determined file type, it calls the corresponding specialized reader function responsible for correctly extracting the content from that specific format. Upon successful execution, the function returns the entire extracted text as a single string. If any error occurs during reading (for example, if the file is corrupted or has an incorrect format) or if the file format is not supported by the current version of the system, the function returns `None`, signaling that the file cannot be processed.
- **analyze_motivation(text):** This function serves as the core and most complex intellectual component of the entire developed analytical process. It implements a multi-step logic for processing text in order to detect motivational patterns.

  1 The function takes a string parameter `text`, which contains the full transcript of the interview to be analyzed.

  2 As a first step, the raw text is segmented into individual sentences using the well-established sentence tokenizer `nltk.sent_tokenize` from the NLTK library.

  3 The system then iterates over each of the obtained sentences, executing the following sequence of operations:

     I. The sentence is tokenized into individual words (or word forms).

     II. The resulting tokens are lemmatized using a `WordNetLemmatizer` instance. Lemmatization reduces various inflected forms of a word to its dictionary (base) form. At the same time, stop-words (frequent, semantically uninformative words like articles, prepositions, conjunctions) and all punctuation marks are removed from the token list.

     III. To ensure proper functioning of the subsequent negation-handling mechanism, the system retains both the list of cleaned lemmas and the original (unprocessed) lowercase tokens along with the mapping between their positions.

     IV. Each significant lemma is checked for presence in one of the two predefined keyword lists: `APPROACH_KEYWORDS` (words indicative of "approach" motivation) and

`AVOIDANCE_KEYWORDS` (words linked to "avoidance" motivation).

V. If a lemma is recognized as a keyword, a negation check is immediately triggered. The mechanism scans the original (non-lemmatized) tokens preceding the keyword within a predefined contextual window (determined by the `NEGATION_WINDOW` parameter). If any negation marker (e.g., "not", "no", "never") is detected in this window, the keyword is marked as *negated* and excluded from motivation classification.

VI. Based on this analysis, two keyword lists are created for each sentence: one containing *non-negated* keywords and the other containing *negated* ones.

VII. Using the presence or absence of non-negated approach/avoidance keywords, the sentence is classified into one of four categories:

   - **Approach** – only approach keywords detected.
   - **Avoidance** – only avoidance keywords detected.
   - **Mixed** – both types of keywords detected.
   - **Neutral** – no relevant non-negated keywords found.

VIII. If the assigned class is anything other than `Neutral`, all relevant information about the sentence (its full original text, assigned label, and lists of non-negated and negated keywords) is added to the overall results list.

4 Once all sentences have been processed, the function returns the accumulated results list, which contains a dictionary for each non-neutral sentence with complete analysis information.

3.3 User Scenario and Interface

The motivational analysis service developed in this project is implemented as a command-line (CLI) application. This means that all user interaction with the system occurs through the terminal interface. A typical user scenario when working with the service generally includes the following sequential steps:

1 **Configuring analysis parameters:** the main parameters that control the analysis process are set by the user directly in the source code of the script.

2 **Launching the text analysis process:** once all necessary configurations have been completed and the parameters are set, the user starts the script execution from the command line (terminal). This is done using the standard command for running Python scripts:

```
python NLP.py
```

3 **Script execution:** during the script execution, the user is prompted to provide several parameters via the console. First, the system requests the name of the file to be analyzed — it can be entered either as a relative path (e.g., a filename from the `Text files` directory) or as an absolute path. If no filename is provided, the script terminates.The user may also optionally enter the name of a second file, in which case both files will be analyzed and compared. Next, the user is asked whether visualizations should be generated, by entering `y` or `n` (or the full forms `yes`/`no`); if confirmed, the system will automatically produce charts based on the analysis results. Once the input is received, the script sequentially performs all processing steps: file existence check, text extraction, analysis, report generation, and visualization (if requested). Throughout the process, detailed execution progress is printed to the console, including the number of sentences analyzed and classified, as well as key metrics. The output CSV filenames are generated automatically based on the input filename, which helps avoid naming conflicts and simplifies report organization.

4 **Receiving and interpreting the results.**

Throughout its execution, the script continuously outputs informative messages to the console. These messages reflect the current progress of the analysis: which file is being processed, whether the analysis has started or completed, and the status of the report generation and visualizations. Additionally, as previously mentioned, the console displays key statistical information for user convenience — such as the total number of analyzed sentences, the number of classified sentences, class distribution, and lists of top keywords — which can be directly used in reporting and interpreting the analysis results.

Upon completion of the analysis, a CSV file is created and saved to disk at a path generated. This file contains detailed results of the linguistic analysis for each sentence classified by the system as non-neutral.

This structured and detailed report format allows the user to easily import the results into widely used spreadsheet applications (such as Microsoft Excel, Google Sheets, or LibreOffice Calc) or other specialized tools and packages for data analysis. It enables extensive opportunities for further in-depth processing, examination, interpretation, and presentation of the results.

The decision to not implement a complex graphical user interface (GUI) at this stage helped simplify development and allowed the focus to remain on the core analytical functionality.

3.4 Visual Representation of Analysis Results

The visualization module, implemented as the `visualize_analysis_results` function, leverages the rich capabilities of popular Python libraries—Matplotlib and Seaborn. Its primary goal is to generate clear, informative, and aesthetically pleasing graphics that reflect the statistical output of the linguistic analysis. When two interview reports are provided, the module automatically switches to a *comparative mode* implemented as the `visualize_analysis_results _dual`: every figure is drawn twice (left / right), enabling convenient side-by-side inspection of both interviews.

The service currently supports four key plot types:

I. Overall Classification Distribution (Figure 4.1) A standard bar chart (built with `seaborn ċountplot`) shows the number of sentences assigned to each motivation category: `Approach`, `Avoidance`, and `Mixed`.

II. Top N Non-Negated Keywords (Figure 4.2) A horizontal bar chart (via `seaborn.barplot`) displays the most frequent non-negated keywords. These words contributed directly to the classification and were unaffected by negation, highlighting dominant motivational themes.

III. Top N Negated Keywords (Figure 4.3) Similar to the previous chart, this bar plot visualizes which keywords were most often negated. Data are taken from the *Negated Keywords Found* column, enabling the researcher to see how frequently particular motivational terms appear in negative contexts.

IV. Classification Dynamics Over Interview Progression (Figure 4.4) To illustrate motivation shifts over time, a stacked bar chart is generated (e.g. via `pandas.DataFrame.plot(..., stacked=True)`). The interview is divided into `N` equal-length segments (default: 5), and for each segment the relative frequencies of `Approach`, `Avoidance`, and `Mixed` sentences are computed. This plot reveals whether motivational trends change as the interview progresses.

Each figure opens in a separate window and is supplied with a clear title (including its figure number) and labelled axes. Seaborn's high-level styling yields a modern, readable appearance compared with default Matplotlib plots, facilitating interpretation of the results. If desired, the entire visual module can be skipped at runtime by answering `n`/`no` to the console prompt.

4. Experimental Testing and Analysis of System Results

This chapter describes the process of testing the developed service, analyzes its results on real data, and discusses limitations.

4.1 Testing Methodology and Data

The purpose of testing is to evaluate the correctness and interpretability of motivational pattern classification, as well as to verify the system's robustness on various types of input texts.

Testing was carried out on a set of 20 real text files containing English-language interview transcripts in `.docx`, `.txt`, `.pdf` formats, which allowed us to verify correct text extraction from different sources. The overall data volume proved sufficient for a comprehensive assessment of system performance.

Testing methodology:

- **Functional testing:** verify correct file reading, error-free script execution, CSV report generation, and graph creation for all test files.
- **Qualitative classification analysis:** since no "gold standard" (manually annotated dataset) exists, the focus was on reviewing generated CSV reports, evaluating the appropriateness of assigned labels (Approach, Avoidance, Mixed), and checking the keyword detection and negation-handling mechanisms.
- **Iterative refinement:** after each test run, keyword lists and negation heuristics were updated based on the analysis.
- **Visualization analysis:** evaluate the informativeness and clarity of the automatically generated graphs.

Using one example file—`1st. Ahmad SohaibWasiq interview.docx`—we can illustrate system output in detail:

- Processed **1** `.docx` file.
- Extracted and initially segmented **483** sentences.
- Analyzed all **483** nonempty sentences as potentially relevant.
- Classified **198** of those as non-neutral (Approach, Avoidance, or Mixed).

- Processing a single file takes **a few seconds**.

Aggregated numeric metrics across all 20 files:

- **Total sentences processed:** 5,739
- **Number classified as non-neutral:** 2,030 (35.4 %)
- **Average non-neutral sentences per file:** 186
- **Average processing time per file:** 1.44 s

These figures confirm that the system consistently handles large, heterogeneous texts: extraction, segmentation, classification, and graph generation run without errors and complete in acceptable time.

4.2 Analysis Results and Interpretation

The qualitative analysis was conducted based on the following criteria: classification adequacy, completeness of detected keywords, correctness of negation handling, and informativeness of the reports and visualizations. Quantitative metrics were extracted directly from the system's output.

4.2.1 Detailed Example for a Single File

For the file `1st.  Ahmad SohaibWasiq interview.docx`, the quantitative outcomes were: Class distribution among those 198 non-neutral sentences (Figure 4.1):

- **Approach** – 117 (59.09 %)
- **Avoidance** – 66 (33.33 %)
- **Mixed** – 15 (7.58 %)

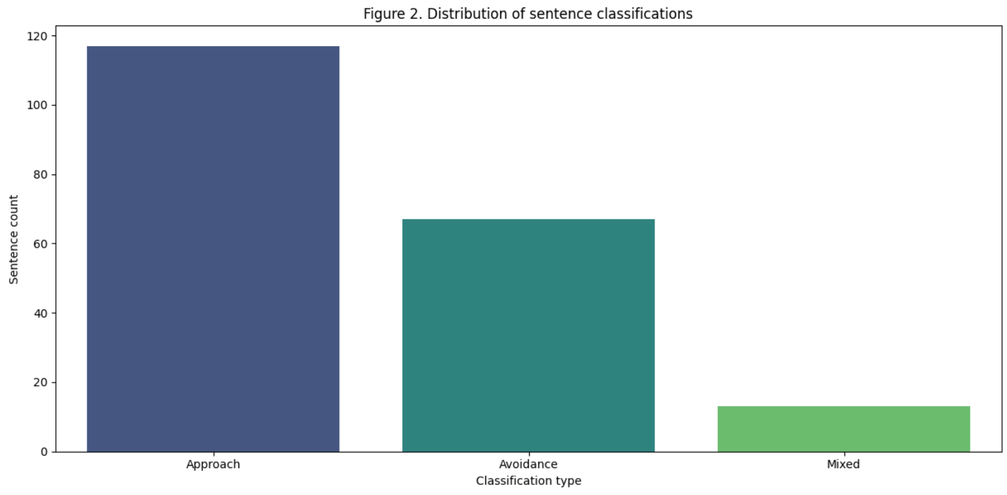

Figure 4.1: Example of sentence classification distribution in an interview

Top most frequent **non-negated** keywords (Figure 4.1):

- *good* (37), *need* (19), *new* (13), *learn* (13), etc.

It indicates that the conversation focused on quality, needs, and new opportunities.

Top most frequent **negated** keywords (Figure 4.2):

- *train* (3), *good* (2), *hard* (1)

This confirms the importance of handling negation.

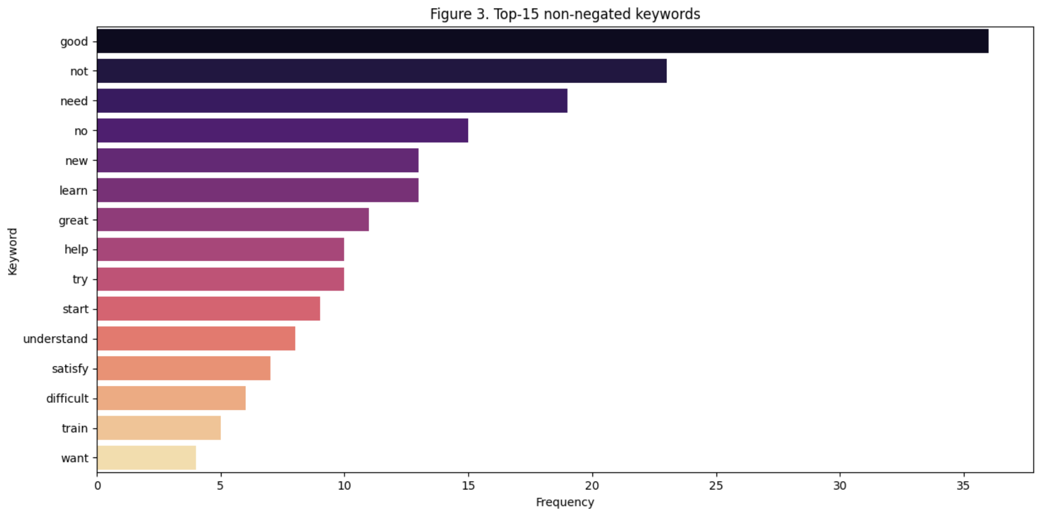

Figure 4.2: Example of the top 15 non-negated keywords

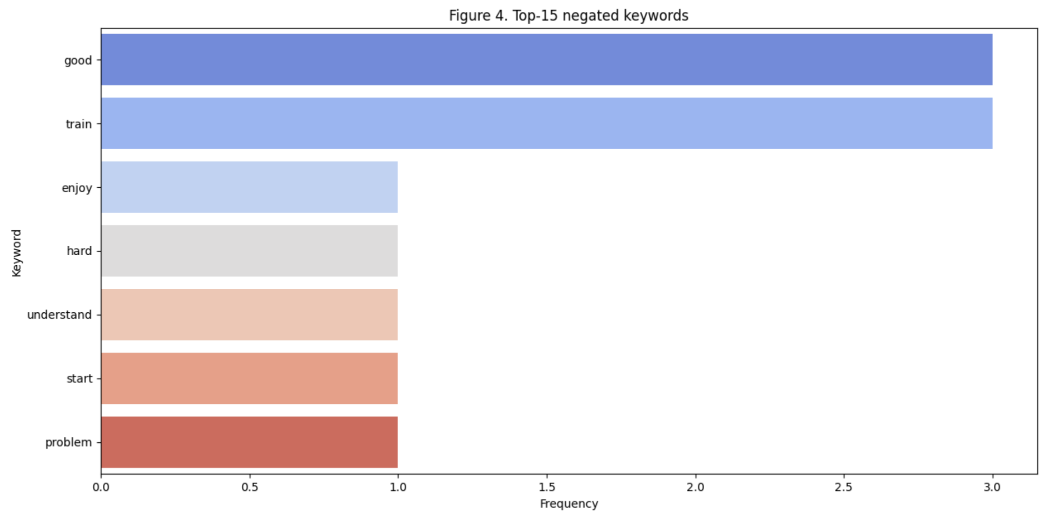

Figure 4.3: Example of the top 15 negated keywords

Negation-handling performance:

- The system successfully processes basic forms such as “not X” or “no Y,” properly tagging the keyword as negated. The mechanism also successfully handled cases like “not very hard,” correctly marking “hard” as negated.
- **Limitations:** Complex or distant negations (e.g., “I might not want to . . . ”) may be handled incorrectly; modality (“might”) and nested negations are not always recognized.

Motivational dynamics (Figure 4.4) across five equal segments of the interview (20 % of text each):

- In the **first segment**, Approach = 71.8 %, Avoidance = 15.4 %
- In the **fifth segment**, Approach = 54.8 %, Avoidance = 42.9 %.

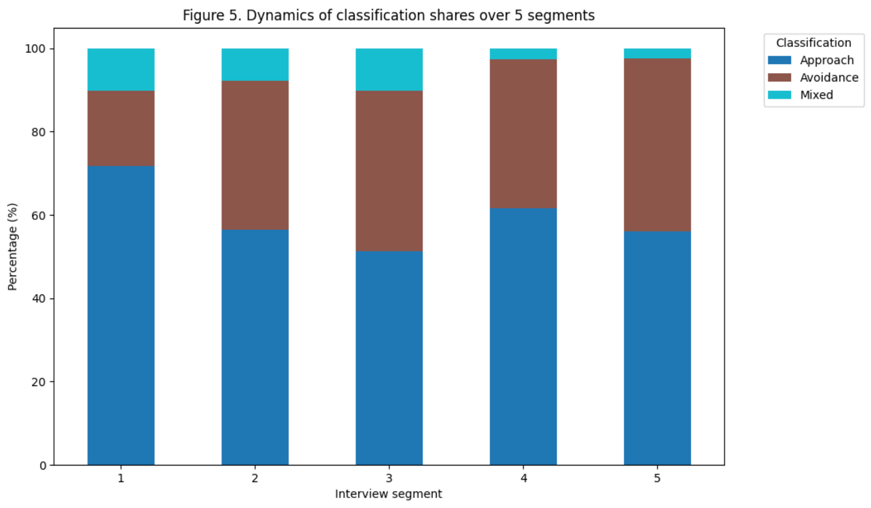

Figure 4.4: Example of a chart showing the dynamics of classification shares across 5 interview segments

This suggests a decrease in optimistic (Approach) motivation toward the end of the interview, with an increase in cautious or negative (Avoidance) expressions.

4.2.2 Aggregated Analysis Across All Files

Aggregated Analysis Across All Test Files

In addition to the single-file example, we aggregated results over all 20 interviews:

- A total of **9,483** sentences were analyzed, of which **3,720** (39.2 %) were classified as non-neutral.
- The average breakdown of classes per file:
  - **Approach:** 73.5 % (on average)
  - **Avoidance:** 16.5 %
  - **Mixed:** 10 %
- The most common **non-negated** terms overall were: *good*, *want*, *try*, *need*, *start.* The most common **negated** terms were: *good*, *understand*, *want*, *problem.*

4.3 Limitations and Future Directions

1 **Dependence on keyword quality.**

- If the keyword list omits rare or domain-specific terms, those will not be captured.

2 **Limited contextual sensitivity.**

- The system does not detect sarcasm, irony, or complex figurative language.
- Constructions requiring external context are processed as individual words.

3 **Lack of a "gold standard" for quantitative accuracy evaluation.**

- Without a manually annotated dataset, metrics such as precision, recall, or F1-score cannot be calculated.
- Qualitative analysis relies exclusively on subjective expert review of CSV reports.

Future directions:

- Integrate machine learning (e.g., train a binary or multi-class classifier) to improve contextual awareness.

- Add more sophisticated rules for handling negations and modality, including nested constructs.
- Develop a full front-end with interactive dashboards, allowing users to view results in real time.

Conclusion

During this course project, a software service for automatic analysis of English-language interviews to detect “approach–avoidance” motivational patterns was successfully developed and tested. The relevance of this work is confirmed by the demand for effective and interpretable text-analysis tools in HR, psychology, and marketing.

Main Results

- **Developed software service:** a command-line Python application was developed, implementing the complete analysis pipeline. It supports reading textual data from various formats (DOCX, TXT, PDF), performing text preprocessing (including tokenization, lemmatization, and stop-word removal), sentence classification based on keyword lists with negation handling and generating structured CSV reports. In addition, the system offers optional visualizations to support interpretation of the results. For enhanced analytical insight, it also includes a comparative mode that enables side-by-side analysis of two interview transcripts, facilitating the identification of motivational differences across speakers or contexts.
- **Interpretable approach:** a rule-and-keyword–based analysis ensures transparency of results.
- **Negation handling implemented:** a heuristic for detecting and accounting for explicit negations improves accuracy.
- **Testing performed:** the system was tested on 20 real interviews, demonstrating functionality. A qualitative analysis and quantitative metrics (e.g., for one test file with 483 sentences, 198 were classified as non-neutral with an Approach share of 59.09%) confirmed the effectiveness of the method.

Limitations

Despite its strengths, the developed service has the following limitations:

- **Dependence on keyword lists:** the accuracy heavily relies on the comprehensiveness of the “approach” and “avoidance” lexicons.
- **Limited contextual understanding:** complex linguistic phenomena such as irony, sarcasm, or nuanced modal expressions are not fully captured by simple keyword heuristics.

- **Partial negation detection:** some negation constructions—especially those using implicit or multi-word negators—may be missed, leading to potential misclassification of sentences.

Future Work

Potential directions for further development include:

- **Expanding the keyword lexicons:** including domain-specific vocabulary (e.g., business, clinical psychology) and colloquial expressions to improve coverage.
- **Enhancing contextual processing:** integrating syntactic parsing and modality detection (e.g., using dependency trees or transformer-based embeddings) to better interpret complex sentence structures.
- **Developing a graphical user interface (GUI):** creating a user-friendly front end so that HR specialists and researchers without programming skills can run analyses and view results interactively.
- **Supporting additional languages:** extending lemmatizers, stop-word lists, and negation heuristics for other languages (e.g., Spanish, French) to broaden applicability.
- **Integrating with a transcription module:** enabling automatic import of interview transcripts from audio, thereby streamlining the workflow from audio recording to motivational analysis.

Overall Conclusion

In summary, the project objectives have been achieved: the developed service demonstrated its effectiveness in initial testing, offers transparency through an interpretable pipeline, and has significant potential for practical applications in real-world HR, social, and marketing researches. There remains room for improvement, but the current implementation already represents a valuable tool for accelerating and objectifying motivation analysis in interview data.

References

[1] Elliot, A. J. The Hierarchical Model of Approach-Avoidance Motivation // *Motivation and Emotion*. – 2006. – Vol. 30. – P. 111–116. – DOI: 10.1007/s11031-006-9028-7.

[2] Elliot, A. J., Church, M. A. A Hierarchical Model of Approach and Avoidance Achievement Motivation // *Journal of Personality and Social Psychology*. – 1997. – Vol. 72, No. 1. – P. 218–232. – DOI: 10.1037/0022-3514.72.1.218.

[3] Dogra, V., Verma, S., Kavita, Chatterjee, P., Shafi, J., Choi, J., Ijaz, M. F. A Complete Process of Text Classification System Using State-of-the-Art NLP Models // *Computational Intelligence and Neuroscience*. – 2022. – Vol. 2022. – Article ID 1883698. – DOI: 10.1155/2022/1883698.

[4] Li, M. Application of Sentence-Level Text Analysis: The Role of Emotion in an Experimental Learning Intervention // *Journal of Experimental Social Psychology*. – 2022. – Vol. 99. – Article ID 104278. – DOI: 10.1016/j.jesp.2021.104278.

[5] De Camillis, P. Analysing Natural Language Processing Techniques: A Comparative Study of NLTK, spaCy, BERT, and DistilBERT on Customer Query Datasets // *MSc in Data Analytics*. – 2022. – No. 1. – URL: `https://arc.cct.ie/cgi/viewcontent.cgi?article=1000&context=msc_da` (Accessed: 20 May 2024).

[6] Bird, S., Klein, E., Loper, E. *Natural Language Processing with Python*. – O'Reilly Media, 2009. – URL: `https://www.nltk.org/book/` (Accessed: 20 May 2024).

[7] Honnibal, M., Montani, I. spaCy 2: Natural Language Understanding with Bloom Embeddings, Convolutional Neural Networks and Incremental Parsing. – 2017. – URL: `https://spacy.io/` (Accessed: 20 May 2024).

[8] Loria, S. TextBlob: Simplified Text Processing. – 2018. – URL: `https://textblob.readthedocs.io/en/dev/` (Accessed: 20 May 2024).

[9] Devlin, J., Chang, M. W., Lee, K., Toutanova, K. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding // *arXiv preprint arXiv:1810.04805*. – 2018. – DOI: 10.48550/arXiv.1810.04805.

[10] python-docx Documentation. – URL: `https://python-docx.readthedocs.io/` (Accessed: 20 May 2024).

[11] PyMuPDF Documentation. – URL: https://pymupdf.readthedocs.io/ (Accessed: 20 May 2024).

[12] Pandas: Powerful Python Data Analysis Toolkit. – URL: https://pandas.pydata.org/pandas-docs/stable/ (Accessed: 20 May 2024).

[13] Matplotlib: Visualization with Python. – URL: https://matplotlib.org/stable/contents.html (Accessed: 20 May 2024).

[14] Seaborn: Statistical Data Visualization. – URL: https://seaborn.pydata.org/api.html (Accessed: 20 May 2024).

[15] Voximplant. Basics of Natural Language Processing for Text. – URL: https://habr.com/ru/companies/Voximplant/articles/446738/ (Accessed: 16 May 2024).

[16] Codex1. Lemmatize It Faster (PyMorphy2, PyMystem3, and a Bit of Magic). – URL: https://habr.com/ru/articles/503420/ (Accessed: 20 May 2024).

Appendix A. Keyword Lists

A.1. Approach Motivation Keywords

{'achieve', 'achievement', 'advance', 'advantage', 'aspire', 'benefit', 'best', 'build', 'challenge', 'contribute', 'create', 'curious', 'desire', 'develop', 'eager', 'effective', 'efficient', 'encourage', 'enjoy', 'enthusiasm', 'excited', 'explore', 'fascinating', 'fast', 'forward', 'gain', 'goal', 'good', 'great', 'grow', 'growth', 'help', 'hope', 'idea', 'improve', 'improvement', 'increase', 'initiative', 'innovate', 'innovation', 'interest', 'learn', 'learning', 'love', 'master', 'motivation', 'motivated', 'new', 'opportunity', 'optimize', 'passion', 'positive', 'potential', 'progress', 'promote', 'pursue', 'ready', 'satisfaction', 'satisfied', 'satisfy', 'seek', 'skill', 'solution', 'start', 'strength', 'succeed', 'success', 'support', 'towards', 'train', 'try', 'understand', 'value', 'want', 'welcome', 'win', 'wish'}

A.2. Avoidance Motivation Keywords

{'afraid', 'anxiety', 'avoid', 'bad', 'blame', 'block', 'burden', 'burnout', 'complain', 'concern', 'conflict', 'confused', 'constraint', 'criticism', 'danger', 'defend', 'delay', 'deny', 'difficult', 'difficulty', 'disadvantage', 'disappointed', 'dislike', 'dissatisfaction', 'dissatisfied', 'dissatisfy', 'doubt', 'drag', 'dread', 'escape', 'fail', 'failure', 'fear', 'fine', 'force', 'frustrated', 'hard', 'hardship', 'hate', 'hesitate', 'hurdle', 'ignore', 'impossible', 'inadequate', 'inhibit', 'insecure', 'issue', 'lack', 'limit', 'limitation', 'lose', 'loss', 'mandatory', 'minimize', 'mistake', 'must', 'need', 'negative', 'neglect', 'nervous', 'never', 'oblige', 'obligation', 'obstacle', 'pain', 'panic', 'penalty', 'prevent', 'problem', 'protect', 'quit', 'reduce', 'reject', 'reluctant', 'resist', 'risk', 'risky', 'stagnant', 'stop', 'stress', 'struggle', 'stuck', 'suffer', 'threat', 'tired', 'trouble', 'unable', 'uncertain', 'uncomfortable', 'unhappy', 'unpleasant', 'unsafe', 'unsure', 'urgent', 'weakness', 'without', 'worry', 'worried', 'wrong'}

A.3. Negation Marker Words

{'not', 'never', 'no', 'without', "n't", 'hardly', 'rarely', 'seldom'}