Movie Subtitles at the Service of Natural Language Processing

Jéssica Cristina Azevedo Veiga

Thesis to obtain the Master of Science Degree in

Information Systems and Computer Engineering

Supervisors: Prof. Maria Luísa Torres Ribeiro Marques da Silva Coheur
             Prof. Sandra Pereira Gama

Examination Committee

Chairperson: Prof. António Manuel Ferreira Rito da Silva
Supervisor: Prof. Maria Luisa Torres Ribeiro Marques da Silva Coheur
Member of the Committee: Prof. José Alberto Rodrigues Pereira Sardinha

August 2020
Acknowledgments

I would like to thank my parents for their unconditional kindness, friendliness, encouragement and support throughout all these years, without whom all battles would resemble endless wars and all great victories would be remembered as small achievements. I am dearly and eternally grateful for all their sacrifices in helping me be who and where I am now and making this project possible.

I would also like to thank my partner for his also unconditional kindness, loveliness, encouragement and perseverance throughout this past years, during a stress full time of constant studying, project implementations, presentation preparations, basically a lot of sweat, but which always ended in happy tears thanks to him.

I would also like to acknowledge my dissertation supervisors Prof. Luísa Coheur and Prof. Sandra Gama for their constant availability, reliability, comprehension and sharing of knowledge that has made this Thesis possible.

I would also like to acknowledge all university teachers with whom I had classes with for their much needed efforts in providing me with the fundamental theoretical concepts and programming abilities in the field of Computer Science.

I would also like to thank my employer for his comprehension and flexibility in granting me the opportunity to start my professional career in my field of study while simultaneously working on my Thesis.

Last, but farther from least, I would also like to thank my dog for always being there to remind me that life is not just hard work and that there has to be some playful moments too.

To each and every one of you – Thank you.
Abstract

The appearance of the *world wide web* allowed the creation and distribution of various extensive corpora. One such corpora are movie and TV show subtitles used in natural language processing to perform tasks such as statistical analysis, conversational agents, among others. However, majority of researchers first need to subject the subtitles to their own preprocessing steps to create a corpus suitable to their task requirements. B-Subtle is an open source framework including various preprocessing steps to build personalized dialogue corpora. In this work we have extended that framework to incorporate additional preprocessing steps considering the existence of subtitle duplicates, the frequency of dialogue turns and the presence of conversation topics. Our goal is to potentially help researchers avoid having to repeatedly implement preprocessing steps and to reduce their corpus to a more manageable size, requiring less computational power and storage capacity, while still being tailored to their system requirements. Furthermore, in this work we also developed from scratch B-Subtle App, a cross-platform desktop application executing the B-Subtle framework in the background and offering statistical analysis of the produced corpora in the form of a visual dashboard using state-of-the-art techniques in the field of information visualization. Our subsequent goals are to offer researchers a visualization that can potentially help in the incremental process of discovering the optimal sequence of preprocessing steps according to their task requirements through the iterative evaluation of the produced corpora and to attempt extending the usage of B-Subtle to a broader audience additionally including cinematographic enthusiasts.

Keywords

Subtitles; Preprocessing; Dialogue Corpora; Statistical Analysis; Visual Dashboard
Resumo

O aparecimento da world wide web permitiu a criação e distribuição de uma variedade de corpora extensa. Um exemplo de tal corpora são legendas de filmes e programas de TV usadas na área de processamento de linguagem natural para executar tarefas como análise estatística, agentes conversacionais, entre outras. No entanto, a maioria dos investigadores primeiramente precisa de sujeitar as legendas aos seus próprios pré-processamentos para criar um corpus mais adequado aos requisitos das suas tarefas. B-Subtle é uma framework open source que inclui vários pré-processamentos para criar corpora de diálogo personalizada. Neste trabalho estendemos essa framework para incorporar pré-processamentos adicionais que consideram a existência de legendas duplicadas, a frequência de diálogos, bem como a presença de tópicos de conversa. O nosso objectivo é tentar ajudar os investigadores a evitar a repetição de pré-processamento e reduzir o corpus a um tamanho administrável, exigindo menos poder computacional e capacidade de armazenamento, permanecendo customizado aos requisitos dos seus sistemas. Para além disso, neste trabalho também desenvolvemos de raiz a B-Subtle App, uma aplicação desktop multiplataforma que executa a B-Subtle framework em segundo plano e oferece análise estatística da corpora produzida na forma de um dashboard visual usando princípios e técnicas atuais na área de visualização de informação. O nosso objectivo seguinte é oferecer aos investigadores uma visualização que tente ajudar na avaliação iterativa da corpora produzida numa tentativa de criar corpora optimizada aos requisitos das suas tarefas, bem como possivelmente estender a utilização do B-subtle a entusiastas cinematográficos.

Palavras Chave

Legendas; Pré-processamento; Corpora de Diálogo; Análise Estatística; Dashboard
# Contents

1 Introduction ................................. 1  
  1.1 Motivation ................................... 2  
  1.2 Objectives ................................... 3  
  1.3 Contributions ............................... 4  
  1.4 Document Structure ......................... 4  

2 B-Subtle .................................. 5  
  2.1 Overview ................................... 6  
    2.1.1 Metadata Collectors ...................... 7  
    2.1.2 Metadata Filters ......................... 7  
    2.1.3 Producers ................................ 7  
    2.1.4 Transformers .............................. 8  
    2.1.5 Interaction Pair Filters .................. 8  
    2.1.6 Analytics ................................ 8  

3 Related Work ............................... 9  
  3.1 Script and Subtitle Corpora .................. 10  
  3.2 Script and Subtitle Applications ............. 13  
  3.3 Tailoring Corpora Tools ...................... 14  
  3.4 Visualizing Corpora Tools .................... 16  
  3.5 Discussion ................................ 23  

4 Solution ................................ 24  
  4.1 Pipeline Components ......................... 25  
    4.1.1 Subtitle Duplicates Removal .......... 25  
    4.1.2 Frequent Pairs Extraction .......... 26  
    4.1.3 Movie Topic Filtering .............. 26  
    4.1.4 Pair Conversation Topic Filtering ... 28  
  4.2 Visual Analytic Tool ......................... 29  
    4.2.1 Data Abstraction ....................... 29
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.2.2 Task Abstraction</td>
<td>29</td>
</tr>
<tr>
<td>4.2.3 Exploratory Work</td>
<td>30</td>
</tr>
<tr>
<td>5 Implementation</td>
<td>34</td>
</tr>
<tr>
<td>5.1 System Architecture</td>
<td>35</td>
</tr>
<tr>
<td>5.1.1 Overview</td>
<td>35</td>
</tr>
<tr>
<td>5.1.2 Specifics</td>
<td>36</td>
</tr>
<tr>
<td>5.2 Pipeline Components</td>
<td>38</td>
</tr>
<tr>
<td>5.2.1 Overview</td>
<td>39</td>
</tr>
<tr>
<td>5.2.2 Subtitle Duplicates Removal</td>
<td>40</td>
</tr>
<tr>
<td>5.2.3 Frequent Pairs Extraction</td>
<td>42</td>
</tr>
<tr>
<td>5.2.4 Movie Topic Filtering</td>
<td>44</td>
</tr>
<tr>
<td>5.2.5 Pair Topic Filtering</td>
<td>47</td>
</tr>
<tr>
<td>5.2.6 Conversation Filtering</td>
<td>49</td>
</tr>
<tr>
<td>5.3 Visual Analytic Tool</td>
<td>51</td>
</tr>
<tr>
<td>5.3.1 First Prototype Critique</td>
<td>53</td>
</tr>
<tr>
<td>5.3.2 Functional Prototype</td>
<td>61</td>
</tr>
<tr>
<td>6 Evaluation</td>
<td>71</td>
</tr>
<tr>
<td>6.1 Usability</td>
<td>72</td>
</tr>
<tr>
<td>6.2 Utility</td>
<td>74</td>
</tr>
<tr>
<td>6.3 Discussion</td>
<td>76</td>
</tr>
<tr>
<td>7 Conclusion</td>
<td>78</td>
</tr>
<tr>
<td>7.1 Conclusions</td>
<td>79</td>
</tr>
<tr>
<td>7.2 System Limitations and Future Work</td>
<td>80</td>
</tr>
</tbody>
</table>
List of Figures

2.1 Interaction pairs considered in a fragment of the subtitle file from the movie “The Lord of the Rings: The Fellowship of the Ring” ................................................................. 6
2.2 B-Subtle architecture. Adapted figure from the dissertation work that first introduced the B-Subtle framework [Angelo, 2018] ............................................................... 7

3.1 FinanViz dashboard described in Nguyen et al. [2019] ............................................. 16
3.2 Compare Clouds dashboard described in Diakopoulos et al. [2015] ........................ 17
3.3 Mitchell WordCloud dashboard described in Schwarz et al. [2017] ......................... 18
3.4 FinaVistory dashboard described in Chan and Qu [2016] ....................................... 19
3.5 PEARL dashboard described in Zhao et al. [2015] ................................................. 20
3.6 ParallelTopics dashboard described in Dou et al. [2011] ....................................... 21
3.7 LeadLine dashboard described in Dou et al. [2012] ................................................. 22

4.1 First non-functional prototype of B-Subtle visual analytic tool made in Adobe XD. .... 30
4.2 Performance widgets. ............................................................................................. 31
4.3 Words widgets. .................................................................................................... 32
4.4 Topics widget. ..................................................................................................... 32
4.5 Sentiments widget. ............................................................................................... 32

5.1 B-Subtle system architecture. ............................................................................... 35
5.2 Subtitle Duplicates Cleaner diagram. .................................................................... 42
5.3 Frequent Pairs Filter diagram. E.g. the starter subtitle corpora includes a subtitle with 3 dialogue turns each with distinct frequencies, through the application of the minPair-Frequency version of this Filter we are able to reduce the collection to the turns with frequency equal or greater than 10. .................................................. 43
5.4 Movie Topic Filter diagram. E.g. the starter subtitle corpora includes 2 subtitles with distinct set of plot-keywords, through the application of this filter we are able to reduce the collection to 1 subtitle that contains all compound, inflected and derived forms of the words “crime and” “computer” simultaneously. ................................................. 47
5.5 **Pair Topics Filter diagram.** E.g. the starter subtitle corpora includes a subtitle with 4 dialogue turns with distinct vocabulary sets, through the application of this filter we are able to reduce the collection to 2 turns that contain a derived form of the word “cook”. 48

5.6 **Conversation Filter diagram.** E.g. this filter is applied to the corpora resultant from the previous application of the Pair Topics Filter and with it we are able to extract 2 additional neighbouring interactions occurring immediately before and after the ones previously filtered. 50

5.7 **Summarization of survey suggestions.** 53

5.8 **Landing page** allowing to browse a starter configuration file or create a preprocessing pipeline on site. 55

5.9 **Landing pages** without the creation of a preprocessing pipeline on site. 55

5.10 **Parallel Tag Clouds.** Compare word frequencies between movie genres with selected words police and question. 57

5.11 **Wordle.** Words positioned horizontally and vertically for efficient usage of idiom area and to support big data sets. 57

5.12 **Tag Cloud.** 57

5.13 **Packed Bubble Chart.** 57

5.14 **Grouped Bar Chart** encodes words number in each year per genre. 59

5.15 **Grouped Bar Chart** encodes words number in each genre per year. 59

5.16 **Bar Chart.** 59

5.17 **Divergent Stacked Bar Chart.** 60

5.18 **Stacked Bar Chart.** 60

5.19 **Grouped Bar Chart.** 60

5.20 **Landing page** including configuration file and analytics folder input options. 62

5.21 **Loading page** when preprocessing pipeline includes analytics component. 62

5.22 **B-Subtle App analytics dashboard.** 63

5.23 **B-subtle App help guide documentation.** 64

5.24 **Execution Time** widget. 65

5.25 **Dialogue Turns** widget. 65

5.26 **Input Files.** 65

5.27 **Output Files.** 65

5.28 **Filtered Files.** 66

5.29 **Word Cloud.** 67

5.30 **Word Count.** 67

5.31 **Topics.** 68

5.32 **Filtered Files.** 68

5.33 **Toolips.** 69

5.34 **Fullscreen, print and download options.** 69
List of Tables

3.1 Summary of corpora built from movie/television scripts and movie/television subtitles. (*)
These values correspond to the number of utterances, not dialogue turns. 12

4.1 Examples of plot keywords from The Movie Database (TMDb). 27

4.2 Performance dataset description. 29

4.3 Subtitle dataset description. 29

4.4 Visual analytic tool tasks and questions. 30

6.1 Application usability evaluation questions definition and corresponding solutions and complexity. 73

6.2 Application evaluation questions definition and descriptive statistics. 73

6.3 Application evaluation on question complexity statistics. 74

6.4 Application evaluation System Usability Scale (SUS). For each question we present the answer in a scale of 1 (Strongly Disagree) to 5 (Strongly Agree) followed by the score contribution in brackets. The Individual Score column represents the usability score for each questionnaire. 74
Listings

4.1 Poker-related subtitle fragment from movie “Casino Royale” ........................ 28
<table>
<thead>
<tr>
<th>Acronym</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CA</td>
<td>Conversational Agent</td>
</tr>
<tr>
<td>IMSDb</td>
<td>Internet Movie Script Database</td>
</tr>
<tr>
<td>IGDB</td>
<td>Internet Game Database</td>
</tr>
<tr>
<td>IMDb</td>
<td>Internet Movie Database</td>
</tr>
<tr>
<td>JVM</td>
<td>Java Virtual Machine</td>
</tr>
<tr>
<td>MB</td>
<td>Megabytes</td>
</tr>
<tr>
<td>NLP</td>
<td>Natural Language Processing</td>
</tr>
<tr>
<td>NLTK</td>
<td>Natural Language Toolkit</td>
</tr>
<tr>
<td>QA</td>
<td>Question-Answering</td>
</tr>
<tr>
<td>SPA</td>
<td>Single Page Application</td>
</tr>
<tr>
<td>SUS</td>
<td>System Usability Scale</td>
</tr>
<tr>
<td>TMDb</td>
<td>The Movie Database</td>
</tr>
</tbody>
</table>
Introduction

Contents

1.1 Motivation .................................................. 2
1.2 Objectives .................................................. 3
1.3 Contributions ................................................. 4
1.4 Document Structure ........................................ 4
1.1 Motivation

The world wide web makes it possible for people all around the world to directly and indirectly contribute to the extension of an ever growing collection of publicly available data. There are almost unlimited sources of data from social media platforms such as Twitter, Goodreads and Reddit, to online databases such as Internet Movie Database (IMDb) and Internet Game Database (IGDB) and much more. In particular, the web provides a type of datasets that have been gaining more and more attention from Natural Language Processing (NLP) researchers, which consists in movie scripts and movie subtitles. There is an enormous amount of research works taking advantage of movie scripts and movie subtitles for performing a wide range of specific tasks: from providing more resources to the scientific community [Nio et al., 2014], to statistical analysis [Banchs, 2012], [Paetzold and Specia, 2016], machine translation [Lison et al., 2018], creation of knowledge-bases [Tandon et al., 2015], movie summarization [Gorinski and Lapata, 2018], violence prediction [Martinez et al., 2019], conversational agents [Banchs and Li, 2012], [Duplessis et al., 2016a], [Yu et al., 2016], [Ameixa and Coheur, 2013], among others.

In all of the research works mentioned above, the movie scripts and subtitles are subject to a preprocessing phase to build a custom and personalized dialogue corpora specific to accomplish the intended task. The preprocessing phase can include multiple steps such as tokenization, semantic similarity filtering, movie genre selection, removal of advertisements, named entity replacement, just to name a few. Still, it is legitimate to ask, why is it necessary to preprocess a dataset? There can be many reasons for the need to preprocess datasets, though we will only present two that are of most interest to this work in particular.

The first reason is due to the fact that movie scripts and movie subtitles constitute an example of an overly large corpora. As a frame of reference, a corpus of subtitles from movies and TV shows since 1880 up to 2018 written in one of the most commonly spoken language world wide, the English language, on its own can surpass 60GB. Processing overly large corpora requires multiple computers with high computational power and storage capacity. This means that researchers, when developing a small project running on their personal computer might find it hard to process such huge amounts of data. As a consequence, the implementation of preprocessing steps becomes almost imperative for processing data in computers with less computational power and storage capacity.

As for the second reason, every time researchers intend to implement a preprocessing phase to get the most suitable dialogue corpora to accomplish their specific task, they usually implement their own set of preprocessing steps that are later made public in their scientific articles and papers. However, after having read some of these, we have noticed that the implemented preprocessing steps tend to be similar and, in some cases, even the same. This alerted us to the fact that it would be helpful to merge a variety of useful and commonly implemented preprocessing steps in a single tool made available to
the scientific community. As a consequence, researchers would be able to minimize the time spent on building custom dialogue corpora and to maximize the time spent focusing on the key problems their research proposes to solve.

B-Subtle\(^1\) [Angelo, 2018] is an open source framework that allows to automatically build custom and personalized dialogue corpora containing dialogue turns (also referred as interaction pairs) extracted from subtitles belonging to the OpenSubtitles Corpus [Lison et al., 2018]. The decision to use the OpenSubtitles Corpus is justified with the fact that it is publicly available through OPUS\(^2\) and is an ever growing dataset of movie and TV show subtitles composed of 3.4 billion utterances covering more than 60 different languages and constituting “the world’s largest open collection of parallel corpora” [Lison et al., 2018]. As for the B-Subtle architecture, it relies on the existence of a pipeline that merges a variety of preprocessing steps to transform the starter subtitle corpora, originally enriched with metadata, into the final custom corpora of interactions. Additionally, the framework contains an analytics module that provides various information, such as a detailed overview of the performance of the pipeline, which includes the total execution time, the number of accepted subtitle files and interaction pairs per filter, the number of output files, among others. However, it is important to note that B-Subtle is supposed to be a framework with long-term maintenance and improvement to accommodate the most recent preprocessing steps being used by NLP researchers, as a means to increase its value in the scientific community. Therefore our work focuses on providing such maintenance and improvement as we will explain in more detail in the following section.

1.2 Objectives

In this work we propose to extend the original version of B-Subtle to include additional preprocessing steps that might be useful in the creation of personalized dialogue corpora for the accomplishment of a broader set of research tasks. Some of these are inspired in preprocessing steps being described in recent research works and others in requirements by the HLT\(^3\) community. Furthermore, because the original version of the framework displays the information gathered by its analytics module in the system console, in this work we will also be using state-of-the-art techniques in the field of Information Visualization to develop a more user-friendly interface. This interface consists in a cross-platform desktop application that not only can be used by researchers in the incremental process of discovering the sequence of preprocessing steps that results in the ideal dialogue corpora according to their research tasks, but that also can extend the usage of the framework to a broader audience additionally including cinematographic enthusiasts with interest in the analysis of movie and

\(^1\)B-Subtle version 1.0 is publicly available at https://gitlab.hlt.inesc-id.pt/miguelv/B-Subtle.
\(^2\)Collection of enriched corpora extracted from multiple external data sources publicly available at http://opus.nlpl.eu/
\(^3\)https://www.hlt.inesc-id.pt/
TV show corpora for tasks outside the scope of Computational Linguistics.

With that said, we propose to add extensions such as the removal of alternative incomplete subtitles, the selection of the most frequent interaction pairs, the selection of subtitles and interaction pairs based on topics, as well as the further extraction of conversations. As for the visualization improvements we propose to create from scratch an application exposing not only the analytical data which is collected already in the original version of the framework, but also additional analytical data such as frequency of words, topics and sentiments, among others.

1.3 Contributions

This work list of contributions include:

- **Extension of B-Subtle framework** with the improvement of existent and creation of new pipeline components as well as analytics modules.

- **Creation from scratch of a statistical analysis visualization** as a desktop cross-platform application for the subsequent exploration of produced corpora.


All code implemented during the course of this work is publicly available on GitLab.

1.4 Document Structure

As for the structure of the remaining document, in Chapter 2 we will provide a thorough description for the architecture of the original version of the B-Subtle framework. In Chapter 3 we will present multiple research works, each proposing to solve a different problem: from building movie dialogue corpora, to movie dialogue corpora applications, to corpora tailoring frameworks and, last but not least, to the implementation of different corpora visualization techniques. In Chapter 4 we will describe briefly the proposed extensions to the original version of B-Subtle, followed by a through description of such extensions in Chapter 5. In Chapter 6 we will describe the approach adopted as well as the corresponding results achieved in the evaluation process of the developed application. Finally, in Chapter 7 we will present our conclusions.
B-Subtle

Contents

2.1 Overview ................................................................. 6
B-Subtle is an open source framework that allows researchers to automatically personalize and customize corpora containing dialogue turns extracted from subtitles belonging to the OpenSubtitles Corpus [Lison et al., 2018]. Subsequently, this corpora can be used in the accomplishment of various tasks such as the implementation of data-driven dialogue systems.

2.1 Overview

The framework functionality relies on the existence of a pipeline in which the input subtitles will be subject to a sequence of preprocessing steps leading to the final corpora of dialogue turns. It is important to note that from this moment on, throughout this work we will use interchangeably the terms dialogue turns and interaction pairs to refer to every two consecutive lines within a dialogue excerpt, composed of both a trigger and an answer, as shown in Figure 2.1.

![Figure 2.1: Interaction pairs considered in a fragment of the subtitle file from the movie “The Lord of the Rings: The Fellowship of the Ring”](image)

To generate the final corpus, researchers should provide as input both a configuration file specifying the overall pipeline, including their need to collect analytical data, as well as the subtitles from the OpenSubtitles Corpus previously extracted through OPUS². The configuration file follows the YAML formatting and should contain an user-defined arrangement of the available pipeline components, each responsible for a preprocessing step. The pipeline components can be abstracted into five categories: metadata collectors, metadata filters, producers, transformers and interaction pair filters. Additionally, the pipeline includes an analytics module for the collection of analytical data regarding the execution of the pipeline. The pipeline components and the analytics module will be further described in the following subsections. An overview of the framework architecture is presented in Figure 2.2.
2.1.1 Metadata Collectors

Most subtitles provided by OpenSubtitles originally include various metadata such as the movie country, duration, genre, rating, year, among others. Nonetheless, to better suit the researchers requirements it is possible to enrich the original subtitles with additional metadata from external data sources through the creation of metadata collectors. The framework provides an instance of a metadata collector based on The Movie Database (TMDb) called themoviedb, which enriches subtitles with movie title and audience certifications.

2.1.2 Metadata Filters

This pipeline component is responsible for filtering the original or previously enriched subtitles based on a specific metadata field-value pair. Most metadata fields can be filtered providing a minimum, maximum, range or exact quantity for their values. The framework enables the researcher to combine one or more of these filters from a total of eleven metadata filters including Audience, Encoding, Genre, Movie and Subtitle Rating, Movie Year, among others.

2.1.3 Producers

As shown in Figure 2.2, following the application of the specified metadata collectors and filters, the framework proceeds to extract interaction pairs to which can be applied the remaining pipeline components, starting with the producers. Producers are responsible for generating tokens, stems, part-of-speech, as well as sentiment analysis annotations to the interaction pairs. So, the framework offers a total of four different producers, which can be combined according to the researchers requirements.

---

1. https://www.themoviedb.org/
2.1.4 Transformers

Transformers are responsible to lowercase or uppercase the interaction pairs, as well as to define the term separator character in an interaction pair. This transformations can be applied to either or both the trigger and answer in an interaction pair. So, the framework offers a total of four different transformers, which, once more, can be combined according to the researchers requirements.

2.1.5 Interaction Pair Filters

This pipeline component is responsible for filtering the interaction pairs based on their characteristics. For most interaction pairs this can be accomplished by providing a minimum, maximum, range or exact quantity for the values of their characteristics. The framework enables the researcher to combine one or more of these filters from a total of eight interaction filters including Trigger and Answer Text Length, Text Content, Token Length, as well as Interaction Interval, among others.

2.1.6 Analytics

In addition to the previously described pipeline components, B-Subtle implements an analytics module to gather analytical data related with the overall execution of a pipeline, such as the total number of input subtitles, the total number of invalid input subtitles, the number of accepted subtitles and interaction pairs per filter, the total number of output files, the total execution time, among others.
Related Work

Contents

3.1 Script and Subtitle Corpora ................................................. 10
3.2 Script and Subtitle Applications .......................................... 13
3.3 Tailoring Corpora Tools ..................................................... 14
3.4 Visualizing Corpora Tools .................................................. 16
3.5 Discussion ................................................................. 23
In this chapter, we describe multiple research works proposing to solve different problems including building script and subtitle dialogue corpora in Section 3.1, using script and subtitle corpora in the development of various systems such as conversational agents in Section 3.2, as well as developing tools for the customization and visualization of corpora in Section 3.3 and Section 3.4 respectively. It is important to note that the presented works are similar to B-Subtle in the extent that both deal with movie scripts and subtitles, however they differ in their main goals. For this reason and because it is of most interest to us, we will mainly be focusing our attention on the preprocessing phases being used by researchers to tailor scripts and subtitles in those works.

### 3.1 Script and Subtitle Corpora

As described in the previous chapter, B-Subtle’s main purpose is to offer a framework capable of tailoring publicly available dialogue corpora consisting in movies and TV show subtitles extracted from the OpenSubtitles Corpus [Lison et al., 2018]. There are two types of dialogue corpora: the natural dialogues which are rich in spontaneous interactions and the unnatural dialogues which are rich in predetermined interactions. Movie scripts and subtitles are considered to be an unnatural dialogue corpora due to being composed of predetermined and fictional interactions, however both are written to resemble spontaneous interactions. They constitute an unique dialogue corpora because being predetermined means “dialogues are more compact, follow a steady rhythm, and contain less garbling and repetition” [Serban et al., 2015a] and being fictional means wider diversity of topics. Furthermore, they avoid having to deal with privacy constraints inherent in most true spontaneous interactions due to the frequent presence of personal information. It is important to note that movie scripts and subtitles differ in the existence and non-existence of a mapping between utterances and speakers, respectively. Besides the aforementioned, the work of Serban et al. [2015a] also presents various dialogue corpora based on movie scripts, such as the Movie-DiC Corpus [Banchs, 2012], the Cornell Movie-Dialogue Corpus [Danescu-Niculescu-Mizil and Lee, 2011], the MovieTriples Corpus [Serban et al., 2015b] and the Filtered Movie Script Corpus [Nio et al., 2014], as well as dialogue corpora based on movie subtitles, such as the SubTle Corpus [Ameixa and Coheur, 2013] and the OpenSubtitles Corpus [Lison et al., 2018]. Next we will provide a general description of each these corpus.

The Movie-DiC Corpus [Banchs, 2012] is a dialogue corpus of 132,229 dialogue turns, all in English, built from movie scripts available at the Internet Movie Script Database (IMDb)\(^1\). The corpus comprises speakers, utterances and context information. It was built to be subject to further statistical analysis, which concluded that each movie had on average 175 dialogues, around 38% of the dialogues had two utterances and around 45% involved two speakers.

\(^1\)https://www.imdb.com/
The main goal of the work described in Danescu-Niculescu-Mizil and Lee [2011] is to investigate the level of coordination between neighbouring utterances in human communication, with a particular interest in function words. To achieve that goal the Cornell Movie Dialogue Corpus [Danescu-Niculescu-Mizil and Lee, 2011] was created. This corpus is composed of 220,579 dialogue turns, all in English, built from movie scripts. The main difference when compared with the previous dialogue corpus is the addition of metadata extracted from the IMDb\(^2\), which include genre, release year, rating, votes, character gender and credits position.

The MovieTriples Corpus [Serban et al., 2015b] was created to compare the performance of generative models used for non-task-oriented dialogue systems. This dialogue corpus consists in the expansion of the Movie-DiC Corpus, subject to further tokenization, named entity recognition and normalization. The result was a dialogue corpus composed of 245,296 utterance triples, all in English.

The Filtered Movie Script Corpus presented in Nio et al. [2014] consists in a dialogue corpus of 86,719 dialogue turns, all in English, built through cross-referencing data from the Friends TV Show, the IMSDb\(^3\) and the Daily Script\(^3\). This corpus was created through processing and filtering an initial dialogue corpus of 1,042,288 dialogue turns. A semantic similarity filter using Dice Similarity Coefficient and WordNet\(^4\) synsets was implemented to assess the correlation of utterances within each dialogue turn and to further dispose of uncorrelated turns. The dialogue corpus is rich in correlated dialogue turns, but is smaller than the previous ones and still restricted to a single language. This work is of particular interest because if we extend B-Subtle to support a similar semantic similarity filter we could also build dialogue corpora rich in correlated dialogue turns, but for over 60 different languages.

The SubTle Corpus [Ameixa and Coheur, 2013] is a dialogue corpus of around 3.35 million dialogue turns, both in English and Portuguese, built from movie subtitles covering Horror, SciFi, Western and Romance genres. This corpus was built to be used by Edgar Smith [Fialho et al., 2013], a domain-oriented Conversational Agent (CA), to cope with out-of-domain interactions.

As the name suggests, the OpenSubtitles Corpus [Lison et al., 2018] is an ever growing database of movie and TV show subtitles available at OPUS\(^2\), which in the year of 2018 contained a total of 3.98 million subtitles covering more than 60 different languages. As a reflection, the OpenSubtitles Corpus has been growing non-stop since its early days in the year 2011 [Tiedemann, 2012], with an increase in volume of almost 40% in the year 2016 [Lison and Tiedemann, 2016] and of around 30% in the year 2018 [Lison et al., 2018]. The current version of OpenSubtitles Corpus is “the world’s largest open collection of parallel corpora” [Lison et al., 2018], composed of 3.4 billion utterances.

OpenSubtitles Corpus is a parallel corpora containing both cross-lingual alignment, i.e. sentence alignment in subtitles across multiple languages, and intra-lingual alignment, i.e. sentence alignment

\(^2\)https://www.imdb.com/
\(^3\)http://dailyscript.com/
\(^4\)https://wordnet.princeton.edu/
for alternative subtitles of the same language. As a result, this corpora constitutes a great contribution to machine translation, as well as alternative translations, paraphrase extraction, just to name a few. However, OpenSubtitles Corpus is more than a parallel corpora mainly built to be used in statistical machine translation. In fact, OpenSubtitles Corpus is an heavily annotated corpora making it also a great contribution in many other fields of study, such as data-driven dialogue systems. The annotations comprise metadata extracted from external sources, such as IMDb\(^2\) and OpenSubtitles\(^5\), and metadata collected during the conversion process used to create the final corpus (for more detailed information on the annotations see Lison and Tiedemann [2016] and Lison et al. [2018]).

Other corpus released one year after the publication of Serban et al. [2015a] is the SubIMDB [Paetzold and Specia, 2016]. The SubIMDB is a structured subtitle corpus composed of around 38 million utterances, all in English, built through cross-referencing data from OpenSubtitles\(^5\) and IMDb\(^2\). This corpus was created to be representative of everyday language usage and provide effective frequency analysis studies, with applications in psycholinguistics, complex word identification and lexical simplification. A preprocessing phase was necessary to refine the subtitles to everyday language. This preprocessing phase included: selection of genres, removal of unwanted information, removal of utterances over 80 characters long, removal of utterances containing words over 15 characters long and removal of advertisements. This work is of particular interest to us, because its preprocessing phase points out the need of some filters that are not supported in the current version of B-Subtle. Those filters consist in both the removal of utterances containing overly long words and the removal of advertisements.

In Table 3.1 we summarize some important features of each of the previously mentioned corpus. Most corpus are small and restricted to a single language, especially when compared with the OpenSubtitles Corpus. Notice the OpenSubtitles Corpus leads the presented corpora with 3.4 billion utterances covering more than 60 different languages. For that reason and considering the goal of this work to tailor corpus for multiple languages, B-Subtle was developed to support subtitles extracted from the OpenSubtitles Corpus (as described in Chapter 2).

<table>
<thead>
<tr>
<th>Corpus</th>
<th>Topic</th>
<th>Dialogue Turns</th>
<th>Languages</th>
</tr>
</thead>
<tbody>
<tr>
<td>Movie-DIC</td>
<td>Scripts</td>
<td>132K</td>
<td>English</td>
</tr>
<tr>
<td>Cornell Movie</td>
<td>Scripts</td>
<td>220K</td>
<td>English</td>
</tr>
<tr>
<td>Dialogue Corpus</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MovieTriples</td>
<td>Scripts</td>
<td>245K</td>
<td>English</td>
</tr>
<tr>
<td>Filtered Movie</td>
<td>Scripts</td>
<td>86K</td>
<td>English</td>
</tr>
<tr>
<td>Script Corpus</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SubTle</td>
<td>Subtitles</td>
<td>3.35M</td>
<td>English and Portuguese</td>
</tr>
<tr>
<td>OpenSubtitles</td>
<td>Subtitles</td>
<td>3.4B*</td>
<td>More than 60 languages</td>
</tr>
<tr>
<td>SubIMDB</td>
<td>Subtitles</td>
<td>38M*</td>
<td>English</td>
</tr>
</tbody>
</table>

Table 3.1: Summary of corpora built from movie/television scripts and movie/television subtitles.

(*) These values correspond to the number of utterances, not dialogue turns.

\(^2\)https://www.imdb.com/
3.2 Script andSubtitle Applications

The corpora described above prove the increasing use of movie scripts and subtitles to create corpus for a wide range of specific tasks: from providing more resources to the scientific community, to performing statistical analysis, to providing multilingual corpora for statistical machine translation and to training CAs. Still there are many more tasks such as the creation of knowledge-bases [Tandon et al., 2015], movie summarization [Gorinski and Lapata, 2018], prediction of violent content through movie scripts [Martinez et al., 2019], among others. However, because training CAs is arguably one of the most trendier tasks, in this section we will describe some CAs trained with movie and television dialogue corpora that present interesting preprocessing steps. Note that our main goal in this section is not to describe in depth the architecture of each of the presented CAs, but rather to point out the need to extend the current version of B-Subtle to support some of the preprocessing steps being used in the creation of their corpora.

The work of Banchs and Li [2012] presents IRIS a data-driven and non-task-oriented dialogue system. IRIS functionality relies on the use of a large movie dialogue corpora represented into the vector space model for further use of the cosine similarity metric. This work uses a “dual search strategy” [Banchs and Li, 2012], which finds the closest match for both the user utterance and the current dialogue history between the user and IRIS, using the cosine similarity metric. Once that is done, the corresponding subsequent utterance is provided as a response to the user (for more detailed information on the IRIS architecture see Banchs and Li [2012]). Concerning the IRIS movie dialogue corpora, it consists of a subset of the Movie-DiC Corpus. This subset is achieved with the implementation of a preprocessing phase, which includes: the extraction and storage of the dialogues in a data structure, removal of bad formatting and misspelling terms, among other features. An interesting feature of this preprocessing phase is the replacement of the names of speakers participating in the dialogue with custom tags, such as <self-name> and <other-name>. This feature points out the need to extend B-Subtle framework to allow the identification of person names and further replacement with custom strings.

Another data-driven non-task-oriented dialogue system is the Joker Chatterbot [Duplessis et al., 2016a], whose main goal is to have daily humorous conversations with the users using as knowledge-base a movie dialogue corpora. The followed strategy also represents dialogue utterances into the vector space model and uses a similarity metric to find the dialogue turn in which the trigger is the closest match to the user utterance, such that it can return as response the corresponding answer (for more detailed information on the Joker Chatterbot architecture see Duplessis et al. [2016b]). Concerning the Joker Chatterbot movie dialogue corpora, it consists in a set of subtitles from comedy and sci-fi television series, which, just like in the work of IRIS, were further subject to a preprocessing phase. This
The preprocessing phase includes various steps, one of which is the removal of text that do not belong to the dialogue, called *subber annotations* [Duplessis et al., 2016b] or *advertisements* [Paetzold and Specia, 2016]. These subber annotations consist in annotations generated by the creators of the subtitles. This feature of the preprocessing phase is of interest to us because if we extended B-Subtle to support a similar subber annotation removal feature, we could help researchers build an even less noisy dialogue corpora.

The *TickTock* [Yu et al., 2016] CA is a data-driven and non-task-oriented Question-Answering (QA) system. TickTock is somewhat similar to the previous CAs due to also using cosine similarity metric for question retrieval. However, TickTock novelty consists in being able to deal with unpredictable user interactions such as sentences composed of a single word, unknown vocabulary, anaphoras, unknown named entities, among others (for more detailed information on the TickTock functionality see Yu et al. [2016]). Focusing on the most relevant part for the purposes of our work, TickTock knowledge-base consists in question-answer pairs extracted from CNN Interview Transcripts of the “Piers Morgan Tonight” TV show. The extraction consisted in searching the original dialogue corpus for question patterns, such as the occurrence of a *question mark* or any *question word*. Note that the current version of B-Subtle allows to filter interaction pairs where the trigger is a question by using either the Trigger Regular Expression or Trigger Text Content filters. However, this approach is not direct and the researcher has to explicitly type every single question pattern he requires for matching. Furthermore, in the work of Yu et al. [2016], the resulting question-answer corpora was preprocessed by removing stop-words, overly long utterances and parentheses/square brackets content, among other steps. The current version of B-Subtle supports all the preprocessing steps explicitly mentioned above, except the removal of stop-words. So, this work is of particular interest because it highlights the need to not only provide a more direct filter for the selection of dialogue turns in which the trigger is a question, but also the need to provide a transformer for the removal of stop-words occurring in dialogue utterances.

### 3.3 Tailoring Corpora Tools

As we have seen in previous sections, usually movie scripts and subtitles are not used in their raw form, but rather subject to a diversity of preprocessing phases that vary according to the researchers specific tasks. In this section we present a similar framework to B-Subtle that attempts to merge a variety of preprocessing steps in a single tool made available to the scientific community. Note that this framework is similar to B-Subtle in the extent that both are used to preprocess corpora, however they deal with different types of corpora. After thorough investigation we were not able to find any framework targeted to preprocess movie subtitles just as B-Subtle.

The work of Bazo et al. [2013] presents the TWORPUS tool which allows the creation of tailored
Twitter corpora according to the researchers specific requirements. Our main interest in this work, despite the fact that TWORPUS supports a different corpora from B-Subtle, is due to the fact that both TWORPUS and B-Subtle aim at solving the same problem consisting in the “lack of customization and personalization of such corpora” [Bazo et al., 2013]. For this reason, we describe some of the most interesting features of the TWORPUS tool in more detail.

Although Twitter is not the main focus of this work, a brief description is necessary for further comprehension of the TWORPUS tool. Twitter is a social media platform mostly known as a microblogging service, in which users post short text messages, called tweets, that might include or not tweet keywords, called hashtags. These tweets are supposed to be shared with the users respective subscribers. Consequently, Twitter has become a widely popular online source of user-generated data allowing for various scientific applications: from the creation of different corpus [Petrovic et al., 2010], [Liu and Homan, 2019], to sentiment analysis [Joshi and Deshpande, 2018], [Tarimer et al., 2019], to event detection [Saeed et al., 2019], to identification of fake news [Agrawal et al., 2019], to cyberbullying detection [Rosa et al., 2018] among others.

The first important thing to mention about the TWORPUS tool is that it is implemented as a web application. This differs from B-Subtle, which has to meet the prerequisite of being available as an open-source framework that can be downloaded and further installed locally on a given computer. The second important thing is that TWORPUS tool connects with the Twitter Streaming API to collect and store tweets metadata in a MySQL database. Note that both TWORPUS and B-Subtle deal with the same problem of big data, one in the form of tweets metadata and the other in the form of subtitles. To solve this problem TWORPUS uses a MySQL database in contrast with B-Subtle current version which filters one subtitle at a time without the help of any database. This approach points out the need to use an optimized database in B-Subtle, such that it can achieve higher speeds in the application of the user-defined filters. Lastly, another important feature of the TWORPUS tool is that, once the tweet metadata has been stored in the database, the researchers can use the web interface to filter the tweets that match specific metadata values. These metadata include IDs of both the tweet and user, word and character counts, hashtags, among others (for more detailed information on the stored metadata see Bazo et al. [2013]). Regarding the hashtags of a given tweet, these consist in a set of user-defined keywords that reflect the topics being approached in the user tweet. This metadata is of particular interest to us because in the movie realm there is a similar metadata consisting in the plot-keywords, which also provides information regarding the topics approached in a given movie. The fact that Bazo et al. [2013] has set as future goal the implementation of filters based on these hashtags, points the need to extend B-Subtle to also allow to filter subtitles based on its underlying topics.
3.4 Visualizing Corpora Tools

As described in Chapter 2, B-Subtle framework enables researchers to tailor dialogue corpora, by having subtitles from the OpenSubtitles Corpus [Lison et al., 2018] being subject to a pipeline composed of an user-defined arrangement of preprocessing steps. Additionally, B-subtle offers an analytics module which is responsible for the extraction of analytical data regarding the overall performance of that user-defined pipeline. In the original version of B-Subtle, this data is being displayed as a simple text print on the system console. However, when dealing with any type of text corpora, it becomes indispensable to use state-of-the-art visualization techniques to help researchers improve their efficacy and efficiency in the analysis of that corpora. To the best of our knowledge, there is no previous work that provides the visualization we require for B-Subtle. For this reason, in this section we will present works which focus on other types of text corpora and in a specific text analytics task either word frequency and context, sentiment and emotion or topic analysis.

![Figure 3.1: FinanViz dashboard described in Nguyen et al. [2019].](https://vixlab305.github.io/FinanViz/)

The work described in Nguyen et al. [2019] presents the FinanViz tool that enables entrepreneurs to search for emerging trends in the marketing and financial world through the analysis and visualization of financial news published in social media platforms. Its dashboard is presented in Figure 3.1 and contains a *small multiples tag cloud timeline*, which includes a tag cloud representing the evolution in time of the most frequent words for each news topic, and a *node-link graph*, in which the nodes represent the words with sudden frequency variations in time and the links represent the co-occurrence of two words in the same news. In terms of interactions it makes use of tooltip, highlighting, time granularity options, keyword search box, among others.

---

6Available at https://vixlab305.github.io/FinanViz/
FinanViz uses multiple coordinated views and is useful to perform various word frequency analysis tasks. However, it has two major disadvantages: the topic clouds position is fixed which hinders the task of comparing topics farther away from each other according to Weber’s Law [Fechner, 1966]; and, its responsiveness and scalability, because using the Google Chrome web browser with an internet speed of around 60Mbps it took 15 seconds to render the entire visualization. It is important to note, this visualization works with a small corpus of 27K news when compared with B-Subtle, which in the worst case scenario, just for the English version of the OpenSubtitles Corpus, has to work with more than 370K subtitles. Therefore, this visualization might not scale for larger datasets.

Figure 3.2: Compare Clouds dashboard described in Diakopoulos et al. [2015].

Another visual analytic tool is Compare Clouds [Diakopoulos et al., 2015] which instead of providing a visual interface for the comparison of word frequency between topics like FinanViz, does so for the comparison of word frequency and context between two distinct corpus. Moreover, it was developed to compare media frames between mainstream news media and online blogs related with the Snowden case.

Compare Clouds’ dashboard, presented in Figure 3.2, includes: a tag cloud which encodes the corpus with the color hue, the word frequency in both corpus with the font size and its degree of exclusiveness in respect to each corpus with both the horizontal position and color saturation; a grouped bar chart to inform on the frequency count of the typed keyword in each corpus; and, a list of 100 samples of raw text which contains both the typed keyword and the words that occur in the same sentence, meaning the context words, for each corpus. These views are coordinated with the use of both a search box or simple selection, to highlight context words, and a z-score filter, to show words with a frequency above and bellow average.

First and foremost, it is important to note that Compare Clouds does not show word frequency evo-
olution throughout time unlike FinanViz, which ultimately makes it a more scalable solution, however lacking answer to a potentially interesting task. Besides the mentioned, Compare Clouds also solves the problem of FinanViz related with the difficulty in comparing tag clouds that are distanced by limiting the comparison to at most two clouds. This can be a disadvantage, because it does not allow to compare more than two clouds at the same time, but on the other hand, not only it reduces the quantity of displayed words in the screen, allowing for a better scalability, but it also allows a more preattentive comparison, with the clear progression from words exclusive to the first corpus, to words common in both corpus and, finally, to words exclusive to the second corpus. Furthermore, this work follows the progressive disclosure design pattern [Chuang et al., 2012], also known as drill-down approach, by including samples of raw text in its visualization.

![Mitchell WordCloud dashboard](image)

**Figure 3.3:** Mitchell WordCloud dashboard described in [Schwarz et al. 2017](#).

The work of [Schwarz et al. 2017](#) proposes a visual analytic tool which, just like the previous works of FinanViz and Compare Clouds, can be considered to fall on the same task of word frequency analysis. This tool is called **Mitchell WordCloud** due to being an alternative approach to the standard search-based interfaces for virtual libraries with a real-life application for the David Scott Mitchell collection, available at the State Library of New South Wales. To accomplish the goal of resembling a traditional book-browsing experience in a physical library, this visualization, presented in Figure 3.3, includes, besides the traditional Dewey bar, to search books in the collection, a tag cloud, to represent the frequency of words occurring in the selected set of books, a bar chart, to represent the number of books throughout time, an ordered list of book titles, and many more features.

Both Mitchell WordCloud and FinanViz allow to analyse the evolution of word frequency in a collection throughout time, unlike Compare Clouds. However, on one hand Mitchell WordCloud provides answer to that task with a more scalable solution for big data visualizations by restricting the tag cloud content to a maximum of 150 words and to a specific time range selected by the user. But on the other hand, this
makes it harder for the user to compare clouds between years, decades or any other time granularity, by relying on recall to do so, which opposes the Nielsen usability heuristic of “Recognition rather than recall” [Nielsen, 1994]. Furthermore, this work provides other options to restrict the cloud content to just locations or person names through the “More Locations” and “More Persons” buttons, respectively. Also, it has multiple coordinated views, follows the drill-down approach by including links to the full books and it is one of the few described works that includes reset, help and about options to ease the users navigation through the visualization.

As stated before, the previous described works are related with word frequency and context analysis. However, the next two works we will describe, even though they can still provide an incomplete answer to that same task, they mainly fall into the category of sentiment analysis works.

Figure 3.4: FinaVistory dashboard described in Chan and Qu [2016].

FinaVistory is described in the work of Chan and Qu [2016] and it was developed to provide a narrative visualization of economic news that allows the exploration of news polarity and impacts of economic indexes variations. Its dashboard, presented in Figure 3.4, revolves around an user-selected index and displays various elements including a stacked bar chart, to represent the evolution in time of the number of positive and negative news related with that index, a wordle, to display the frequency of words occurring in the positive and negative news, an heatmap to represent the evolution in time of other indexes polarity, among others.

First and foremost, compared with the previous works, the use of a wordle instead of a tag cloud makes a more efficient use of the available space by placing words in different orientations, however this not only makes it harder to read, but can also induce the user in thinking the orientation encodes other attribute violating the Expressiveness Principle [Mackinlay, 1986], which states that an idiom should encode all and only the information in the dataset attributes. Also, this work has other disadvantages:
first, regarding the stacked bar chart not only its y-axis scale is not legible opposing the Tuft design principle “Labeling should be clear and detailed” [Tuft, 1986], but also one of the main problems inherent to this visual encoding is the difficulty in comparing the length of the bar segments given they are not aligned, according to Weber’s Law [Fechner, 1966]; lastly, the inconsistent use of red and green to not only encode positive and negative polarity of news, but also some economic indexes. However, it has its advantages: multiple coordinated views, follows the drill-down approach and it includes an help button following the Nielsen usability heuristic “Provide help and documentation” [Nielsen, 1994].

Figure 3.5: PEARL dashboard described in Zhao et al. [2015].

The previous work of FinaVistory includes a simple sentiment analysis of positive and negative economic news based on the increase and decrease of the corresponding economic index respectively. Another interesting work that delves deeper into a more detailed level of sentiment analysis is PEARL [Zhao et al., 2015], presented in Figure 3.5. PEARL is a visual analytics tool focused on the analysis of a person emotional style based on his tweet history and aimed at personal self-reflection and third-party profiling. This work uses both the Discrete Categorical and Continuous Dimensional emotion models to define a mood as a distribution over the eight emotion components of anger, fear, anticipation, surprise, joy, sadness, trust and distrust, each associated with a value of valence, arousal and dominance.

PEARL’s main visual components are: a detail view with an area chart, to represent the number of tweets over time, a tag cloud, to represent the word frequency for tweets of the selected time range, and a streamgraph, to represent the moods over time; a mood view with a pie glyph-based scatterplot, to represent the tweets emotional keywords used to infer the selected mood; and, a raw tweets view with a list of highlighted emotional keywords.

This visualization presents advantages such as the use of the drill-down approach, allowing to transition between overview of moods and details of raw tweets, as well as multiple coordinated views, with
brushing, linking and filtering interactions. Moreover, the choice of colors for the emotion components of moods is not random, instead it fits the emotion analysts mental model by following the color scheme used in the Discrete Categorical model, as means to increase recognition and decrease recall as much as possible. However, it also presents some disadvantages. The streamgraph used in this work offers the same problem as the stacked bar chart used in the previous described work. Moreover, almost all used views can scale for larger static tweet datasets with the exception of the scatterplot provided it uses pie glyphs, which in larger quantities even with the help of opacity can still cause occlusion problems.

So far, we have described works focusing on word frequency and context analysis, as well as sentiment and emotion analysis. However, another popular text analytics task among the scientific community is topic analysis, which will be our focus in the following works.

**Figure 3.6:** ParallelTopics dashboard described in Dou et al. [2011].

**Parallel Topics** [Dou et al., 2011], presented in Figure 3.6, is one of multiple visual analytic tools that allow to analyse the evolution of topics over time encoded in a ThemeRiver [Havre et al., 2000]. However its main contribution is the further analysis of the distribution of documents across a set of topics encoded in a parallel coordinates, with an axis for each topic, representing its proportion in each document, and a line for each document, representing the subset of topics it covers. Furthermore, it includes a small multiples tag cloud, in which the word vertical position and font size represent its frequency in the topic and in the collection respectively, and also a pie glyph-based scatterplot, in which the pie glyph represents a document with its slices and positions representing the topics and topic number respectively.

This visualization has various positive aspects worth mention besides implementing coordinated views and following a drill-down approach, such as the careful implementation of the parallel coordinates with axis ordered based on topic similarity and normalized scales, as well as brushing support to reduce potential visual clutter. However, it has its downsides: the views are presented in separate windows; the overall visualization does not scale properly for larger datasets, because, for instance, it repeats the...
same issue mentioned in PEARL related with the use of glyphs causing occlusion problems; and, the topic numbered labels not only it is not clear the meaning of them violating the Tufte design principle “Labeling should be clear and detailed”, but also they do not reflect the categorical nature of topics violating the Expressiveness Principle [Mackinlay, 1986].

Figure 3.7: LeadLine dashboard described in Dou et al. [2012].

Unlike ParallelTopics, which aims to analyse the evolution of topics over time, LeadLine [Dou et al., 2012] aims to further analyse the cause-effect relationship between events and trending topics. Through the combination of topic modelling, named entity recognition and early event detection techniques, they created a topic-based summarization of news and microblog messages further displayed in a visual analytic tool for event exploration, presented in Figure 3.7.

This visualization uses: a topic view with a splitted streamgraph, in which events are represented with highlighted and contoured segments of the streams and the corresponding keywords are displayed next to them in small tag clouds, overlaying another streamgraph faded into the background, which represents an overview of the evolution of all topics; an entity view with a node-link graph, to represent the co-occurrence of entities in the text; and, a geospatial view with a dot map, to represent locations related to the selected entity.

In the previous work of Parallel Topics the user had to recall a topic color to analyse both the streamgraph and small multiples tag cloud, this is solved in LeadLine by having the tag cloud overlying the corresponding stream, as means to follow the Nielsen usability heuristic of “Recognition rather than recall”. Moreover, this visualization is presented in a single window with coordinated views and follows a drill-down approach with the possibility to visualize raw text. Other of its advantages are the improvement of preattentive processing with the use of color to highlight events and, also, the display in its node-link graph of a subset of entities according to the user selected topic, time or entity type, which scales for larger datasets. On the other hand, according to Weber’s Law, its main disadvantage is that both visual encodings used in the topic view do not allow to easily compare topics that are farther apart.
3.5 Discussion

In this chapter we have presented various works preprocessing movie scripts and subtitles to perform a wide range of tasks including building dialogue corpora, frequency analysis, machine translation, conversational agents and others. However, these works seem to be “reinventing the wheel” provided they have to re-implement preprocessing steps from previous works, instead of focusing solely on the main problem they propose to solve. For this reason, with this work we propose to extend the B-Subtle framework to include some of the preprocessing steps being used in the scientific community, such as subber annotation removal, as well as others required by the HLT community7, such as topic filtering.

Also, in this chapter, we have presented works whose goal is to provide a visual analytic tool to help researchers investigate specific types of corpora, as means to get answers to a specific text analytics task. As we have seen, not only most tools are mainly focused on analysing the evolution of a given text characteristic over time, but also most tools follow a drill-down approach, offering the possibility to transition between overview and details of raw text, and use multiple coordinated views, with the use of various techniques including brushing, highlighting, selecting, searching and filtering. Moreover, most tools focused on word frequency analysis tend to use tag clouds and on topic analysis tend to use streamgraphs. However, most of those works have scalability issues, mainly due to the possibility of occlusion for larger datasets, and violate the Weber’s Law, mainly due to the placement of data to be compared farther apart from each other. Other commonly violated design principles include: Expressiveness, Tufte “Labeling should be clear and detailed” and Nielsen “Recognition rather than recall”. Furthermore, it is important to note that from all the presented works only one offers both an help and reset options. So, each described work presents its own disadvantages and we were not able to find any work that deals with movie subtitles and combines word frequency, sentiment and topic analysis in the same visualization. For this reason, to accompany B-Subtle framework, we also propose to develop from scratch a visual analytic tool that not only scales well for larger datasets and follows the mentioned design principles, but also can provide answer to questions related with all those tasks, as well as general information regarding movies and TV shows.

7https://www.hlt.inesc-id.pt/
4
Solution

Contents

4.1 Pipeline Components .................................................. 25
4.2 Visual Analytic Tool ..................................................... 29
In Chapter 2, we have presented the B-Subtle framework which offers various pipeline components that enable researchers to refine their final collection of dialogue turns according to their specific research tasks. In Chapter 3, we emphasized the need to extend the current version of B-Subtle pipeline, to support additional preprocessing steps that have been used in the scientific community, and to develop from scratch a B-Subtle visual analytic tool, for further analysis of the produced corpora. In this chapter, we describe the extensions we propose to perform to the current version of B-Subtle. These extensions are divided into the addition of new pipeline components in Section 4.1 and the addition of an entire visual analytic tool in Section 4.2.

4.1 Pipeline Components

In this section we propose to extend B-Subtle framework with additional pipeline components inspired from some preprocessing steps of the works described in Chapter 3 and others required by the HLT\(^1\) community. Because those works merely report the use of such preprocessing steps with no full explanation on why and how they were implemented, in the present section we will briefly present the motives and proposals on our own solutions for the implementation of such preprocessing steps. However, a more thorough explanation will be provided in Section 5.2.

4.1.1 Subtitle Duplicates Removal

As described in Chapter 2, B-subtle receives as input movie and TV show subtitles from the OpenSubtitles Corpus [Lison et al., 2018]. Despite all the efforts during the preprocessing phase, the OpenSubtitles Corpus continues to be somewhat noisy. One of the main reasons for that is the occurrence of subtitle duplicates, which consist in alternative subtitles in the same language for the same movie that in most cases are incomplete. With being incomplete we mean that those subtitles only contain a fragment of the total movie or TV show script. As an example, consider the case of the movie “Wonder Woman”, for which the English version of the OpenSubtitles Corpus presents a total of twelve subtitles. The first official spoken sentence of the movie is “I used to want to save the world.” which agrees with seven out of twelve of those subtitles, however the other five subtitles start with a different sentence “Hello, Diana.”, which is officially the seventeenth spoken sentence of the movie.

To identify subtitle duplicates we will use some metadata attributes included in the subtitles from the OpenSubtitles Corpus such as real user subtitle ratings (<rating>), as well as subtitle total number of sentences, corrected misspelled words and unknown words (<sentences>, <corrected_words> and <unknown_words> respectively). We will define two algorithms for the identification of subtitle duplicates, one which will consider duplicates all the subtitles with the lowest user ratings and other

\(^1\)https://www.hlt.inesc-id.pt/
which will calculate a machine rating that will consider duplicates all subtitles with lowest number of sentences and corrected misspelled words and highest number of unknown words. To accomplish this we will propose the addition of a new pipeline component type called Cleaners. A Cleaner is a special filter that is executed at the beginning of the pipeline to reduce the initial noisiness of the starter subtitle corpora. Lastly, we propose to add a Cleaner specific for the resolution of the presented problem which we will call Subtitle Duplicates.

4.1.2 Frequent Pairs Extraction

The wide diversity of scenarios covered by movies, makes the corresponding subtitles be composed of a wide variety of interactions. Some interactions, besides occurring in specific and improbable/unrealistic scenarios, are also too closely related with those same scenarios, and for that reason they occur a limited number of times throughout the movie subtitle corpora. In some applications of a movie subtitle corpora the occurrence of such rare interaction pairs might not be a concern, such as statistical analysis, machine translation and others. However the same does not apply to all applications and an example of that are CAs. Consider the interaction “Ogres are like onions.” followed by “They stink?” in the English subtitle of the movie “Shrek”, 2001. In rare situations a dialogue system needs to consider such interaction pair as a possible dialogue between the system and the user. On the other end, despite the diversity of scenarios, some of them are common throughout movies for example greetings, such as “Hello!” and “Goodbye!”. This implies that there is also a redundancy inherent to the movie subtitles that somehow reflects the redundancy inherent to our daily conversations.

To take advantage of this, we will allow the user to extract dialogue turns based on their frequency of occurrence in the starter dialogue corpora per subtitle, through the use of an interaction pair frequency list. This list would store for each interaction pair the total number of times it occurs in the movie dialogue corpora. To accomplish this we will propose to extend B-Subtle to include an additional Interaction Pair Filter called Interaction Frequency.

4.1.3 Movie Topic Filtering

Consider the case where a researcher intends to develop a CA whose target domain consists in a card game such as poker. In this case, the researcher could be interested in gathering a corpus rich in interactions related with poker and, ideally, during poker games. The vast topic diversity of movies and TV shows makes it possible for the researcher to select subtitles from a collection of poker-related movies as a knowledge-base for his CA.

In order to refine a subtitle corpora to poker-related movies, we would first need to enrich that corpora with metadata regarding the corresponding movie topics. In a movie description the attribute that best
reflects the list of topics covered in the movie are the plot-keywords (see Table 4.1). So, we first need to extend the themoviedb Metadata Collector to additionally extract plot-keywords metadata from the TMDb API. Afterwards, we would still require to filter the subtitles based on that metadata. So, lastly, we need to create a Topic Metadata Filter to select subtitles in which the plot-keywords metadata includes the topic or list of topics provided by the user.

It is important to address the fact that TMDb is a community-built database that offers the contributors the freedom to assign any relevant plot-keywords to a given movie or TV show. Consequently, contributors can use different plot-keywords to refer to the same topic (see highlighted topics in Table 4.1). Most variations consist in the addition of more information to the topic, known as a form of hyponyms (e.g. “terrorist” and “cyber terrorist”). However, other variations consist in different inflected forms of the same word (e.g. “bank” and “banker”). To solve this, we will propose to use some custom solutions based on tokenization and stemming to consider every semantic and lexical related forms of the plot-keywords. Thus, if the researcher applied this filter providing as topics the terms “terrorism” and “financial institution”, the result would not only include the movie “Casino Royale”, but also the movies “Now You See Me” and “Blackhat”.

<table>
<thead>
<tr>
<th>Movie Title</th>
<th>Plot-Keywords</th>
</tr>
</thead>
<tbody>
<tr>
<td>Casino Royale</td>
<td>italy, banker, poker, casino, money, torture, terrorist, [...]</td>
</tr>
<tr>
<td>Now You See Me</td>
<td>bank, fbi, vault, magic, investigation, heist, money, escape [...]</td>
</tr>
<tr>
<td>Blackhat</td>
<td>technology, hacker, computer virus, fugitive, cyber terrorist, [...]</td>
</tr>
</tbody>
</table>

Table 4.1: Examples of plot keywords from TMDb.

However, it is important to note that a possible disadvantage of this entire solution is its dependency on the TMDb API, which at the time of our proposal restricted usage up to 40 HTTP requests every 10 seconds. As a frame of reference, the biggest corpora a user can input to B-Subtle is the English version of the OpenSubtitles Corpus composed of more than 370K subtitles, which just to be enriched with plot-keywords would require approximately 22 hours. Nonetheless, this solution would still continue to be advantageous due to its diversity of topics, besides we estimate that with the implementation of the Subtitle Duplicates cleaner (described in Section 4.1.1) the starter subtitle corpora will be severely reduced, meaning that by the time the Topic filter is applied we would have a much smaller corpora to deal with. However, fortunately, at the time of our implementation since December 16 2019, TMDb has disabled that request rate limit as reported in their official announcement on the TMDb developers website.

5https://developers.themoviedb.org/3/getting-started/request-rate-limiting
4.1.4 Pair Conversation Topic Filtering

With the extensions described in the previous subsection, a researcher can more easily refine his collection of subtitles to movies that address a specific topic or set of topics provided by him. Once again, consider the case in which a researcher wants to develop a CA to chat with its adversaries while playing poker. Because movies address various topics simultaneously, a simple movie topic selection would not be enough to ensure all extracted interactions are poker-related. This means the researcher would additionally need to refine his collection to only poker-related interaction pairs. In order to accomplish this, we will implement a new Interaction Pair Filter, called Interaction Topic. This Interaction Topic filter will extract topic pairs, meaning interaction pairs in which occur a single term or list of terms provided by the user. However, the user cannot provide a complete list of poker terms, so once again we will propose to use custom solutions based on tokenization and stemming to consider every semantic and lexical related forms of the user provided terms.

Furthermore to avoid the loss of various interaction pairs that even though they are not topic pairs, they occur in the same dialogue and might belong to the same conversation topic, we will need to extract not only the topic pairs, but also neighbouring interaction pairs within a user-defined conversation length. So, for instance, considering Listing 4.1, if the researcher applied the filter with term “game” and defined a conversation length of 1, the result would not only include the topic pair in line 7, but also the interaction pairs in line 3 and 10. Note that with a conversation length of 1, line 15 would not be included in the result. To accomplish this we will propose the creation of a new type of filter called Conversation Filter which will allow to extract conversations, meaning a sequence of interaction pairs neighbouring not only topic pairs but any pairs previously filtered with any Interaction Filter (defined in section 2.1.5).

**Listing 4.1:** Poker-related subtitle fragment from movie “Casino Royale”

```
1 <s id="444"> <time id="T151S" value="01:07:49,162" /
2     Ladies and gentlemen, welcome.
3 </s>
4 <s id="445"> <time id="T152S" value="01:07:51,428" /
5     As you know, the game is “no limit”, Hold’em poker.
6 </s>
7 <s id="446"> <time id="T152E" value="01:07:56,292" />
8     Five communal cards and two on the hold.
9 </s>
10 <s id="447"> <time id="T153S" value="01:07:56,292" />
11     Monsieur Mendel represents the Basel banks in Switzerland holding the sticks.
12 </s>
```
4.2 Visual Analytic Tool

In this section we initiate our proposal to improve the B-Subtle framework console-based interface with an entire visual analytic tool that can provide answer to various research tasks. We will present the tool dataset in Section 4.2.1, the tasks and respective questions it aims to answer in Section 4.2.2, as well as the tool first non-functional prototype in Section 4.2.3. However, the description of the tool architecture, the explanation on some limitations of the underlying framework that were not considered when sketching the first non-functional prototype, as well as the description of the final application developed in this work, will be provided in Section 5.3.

4.2.1 Data Abstraction

As described in Chapter 2, the current version of B-Subtle analytics module extracts various data related with the overall performance of its pipeline, which we will refer as the performance dataset. However, in this work we propose to extend the analytics module to extract further data related with each individual movie and TV show subtitle, which we will refer as the subtitle dataset. The latter dataset includes subtitle metadata and derived data such as word frequency, sentiments, topics, and others. A thorough description of both datasets is provided in Table 4.2 and Table 4.3, respectively.

<table>
<thead>
<tr>
<th>Attributes</th>
<th>Type</th>
<th>Ordering</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number and Size of Processed Input Files</td>
<td></td>
<td>-</td>
</tr>
<tr>
<td>Number and Size of Invalid Input Files</td>
<td></td>
<td>-</td>
</tr>
<tr>
<td>Number and Size of Output Files per Type</td>
<td></td>
<td>-</td>
</tr>
<tr>
<td>Size of Largest Input File</td>
<td></td>
<td>-</td>
</tr>
<tr>
<td>Size of Input File with More Interaction Pairs</td>
<td></td>
<td>-</td>
</tr>
<tr>
<td>Size of Largest Output File per Type</td>
<td></td>
<td>-</td>
</tr>
<tr>
<td>Total Time Processing All Files</td>
<td></td>
<td>-</td>
</tr>
<tr>
<td>Avg. Time Processing Each File</td>
<td></td>
<td>-</td>
</tr>
<tr>
<td>Avg. Number of Interaction Pairs per File</td>
<td></td>
<td>-</td>
</tr>
<tr>
<td>Avg. Time between Trigger/Answer</td>
<td></td>
<td>-</td>
</tr>
<tr>
<td>Number of Files Accepted per Filter</td>
<td></td>
<td>-</td>
</tr>
</tbody>
</table>

Table 4.2: Performance dataset description.

<table>
<thead>
<tr>
<th>Attributes</th>
<th>Type</th>
<th>Ordering</th>
</tr>
</thead>
<tbody>
<tr>
<td>Title</td>
<td>Nominal</td>
<td>-</td>
</tr>
<tr>
<td>Genre</td>
<td>Nominal</td>
<td>-</td>
</tr>
<tr>
<td>Production Company Country</td>
<td>Nominal</td>
<td>-</td>
</tr>
<tr>
<td>Topics</td>
<td>Nominal</td>
<td>-</td>
</tr>
<tr>
<td>Spoken Languages</td>
<td>Nominal</td>
<td>-</td>
</tr>
<tr>
<td>Words</td>
<td>Nominal</td>
<td>-</td>
</tr>
<tr>
<td>Words Frequency</td>
<td>Ratio</td>
<td>Sequential</td>
</tr>
<tr>
<td>Number of Tokens</td>
<td>Ratio</td>
<td>Sequential</td>
</tr>
<tr>
<td>Number of Sentences</td>
<td>Ratio</td>
<td>Sequential</td>
</tr>
<tr>
<td>Duration</td>
<td>Ratio</td>
<td>Sequential</td>
</tr>
<tr>
<td>Release Date</td>
<td>Continuous</td>
<td>Sequential</td>
</tr>
<tr>
<td>Trigger/Answer Sentiment</td>
<td>Ordered</td>
<td>Diverging</td>
</tr>
</tbody>
</table>

Table 4.3: Subtitle dataset description.

4.2.2 Task Abstraction

Our visual analytic tool will make use of both datasets presented in the previous subsection to provide answer to various tasks. In Table 4.4 we present the set of tasks and example questions our tool proposes to answer. The questions aim to provide maximum coverage of each corresponding task.

<table>
<thead>
<tr>
<th>ID</th>
<th>Tasks</th>
<th>Example Questions</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Summarize pipeline performance metrics</td>
<td>What is the total pipeline execution time?</td>
</tr>
</tbody>
</table>

Continue on the next page
<table>
<thead>
<tr>
<th>ID</th>
<th>Tasks</th>
<th>Example Questions</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>Compare pipeline performance metrics</td>
<td>Have metadata filters discarded more subtitles than interaction pair filters?</td>
</tr>
<tr>
<td>3</td>
<td>Summarize words/topics occurrence in screenplays</td>
<td>Is the topic “alien invasion” more frequent than “terrorism” in screenplays?</td>
</tr>
<tr>
<td>4</td>
<td>Compare words/topics occurrence in screenplays between groups</td>
<td>How different are the vocabularies used in horror and comedy movies? And in the 90’s and recent movies?</td>
</tr>
<tr>
<td>5</td>
<td>Identify occurrence of special screenplay words</td>
<td>What are popular character names used in screenplays?</td>
</tr>
<tr>
<td>6</td>
<td>Summarize sentiments occurrence in screenplays</td>
<td>In general, are movies more positive or negative?</td>
</tr>
<tr>
<td>7</td>
<td>Compare sentiments occurrence in screenplays between groups</td>
<td>Are crime movies more negative than horror movies?</td>
</tr>
</tbody>
</table>

Table 4.4: Visual analytic tool tasks and questions.

4.2.3 Exploratory Work

To provide answer to the tasks described in the previous subsection we have sketched a prototype for the B-Subtle visual analytic tool shown in Figure 4.1. It is important to emphasize that it is merely a first non-functional prototype that was subject to further changes and improvements through the course of our work as will be thoroughly explained in Section 5.3.

![Figure 4.1: First non-functional prototype of B-Subtle visual analytic tool made in Adobe XD.](image)

In our prototype we have adopted the concept of a customizable dashboard. The dashboard is composed of widgets that are draggable and detachable. Because our tool targets a wide public which
within the scope of movies and TV shows might have different interests, we made an effort to offer our users the possibility to customize their dashboards by being able to place each widget in the desired area as well as removing the ones undesired. However, to initially provide an overview of the data and show the users the potential of our tool, the default state of the dashboard includes all featured widgets.

The control panel, in the left side of the dashboard, combines all the options to control the behaviour of the dashboard and is composed of 4 sections: a search section, which includes a search box for the user to search specific widgets, genres, years or even specific words and topics; a widgets section, which includes a list of widgets that can be added or removed from the dashboard; a configuration section, which includes both global and local configurations; and an help section, which includes a reset button, to recover the initial state of the dashboard, an help button, to trigger the start of a tutorial on how to use the tool, and an about button, to render a new page with information on the development context for the tool. The configuration section is divided into two subsections corresponding to two types of configurations: global configuration, which offers three alternatives to aggregate the displayed data including corpora, for a general overview of the entire set of subtitles, genre, for a view per movie genre, and year, for a view per movie release year (in Figure 4.1 both “Action” and “Comedy” genres are selected); and local configuration, which offers configuration options that differ according to the selected widget (in Figure 4.1 the selected widget is “Topics”). Some local configurations include: special words/topics, to highlight character names and locations; co-occurrence, to explore the strength of relations between words/topics; among others.

The widgets are divided into four categories corresponding to four different views of the data: the performance view (in Figure 4.2), which includes all metrics related with the performance of the pipeline used to produce the corpora; the words view (in Figure 5.23), which includes all metrics related with word frequency; the topics view (in Figure 4.4), which includes all metrics related with topic frequency and co-occurrence; and the sentiments view (in Figure 4.5), which includes all metrics related with sentiment evolution throughout a movie. Next we will describe each of the 9 widgets available in our dashboard.

(a) Execution time. (b) Dialogue turns.
(c) Input files. (d) Filtered files. (e) Output files.

Figure 4.2: Performance widgets.
The Execution Time and Dialogue Turns widgets (in Figure 4.2a and Figure 4.2b, respectively) do not consist in a visualization *per se*, but rather a textual summarization of the corresponding metrics. In more detail, the former presents the total execution time, as well as the average and maximum execution time per input file, while the latter presents the average and maximum number of interaction pairs per file, as well as the average time difference between trigger and answer.

To encode the Input Files, Filtered Files and Output Files metrics we have considered the use of a donut chart, in which the file type is encoded by the *color hue* or *color saturation* and the total number of files is encoded in the size of the donut slice (in Figure 4.2c, Figure 4.2d and Figure 4.2e, respectively). In more detail: the first widget encodes the ratio of processed and invalid files; the second widget encodes the ratio of files accepted by each filter; and the third widget encodes the ratio of files produced for each output type.

To encode the Word Cloud metric we have considered the use of a *tag cloud*, in which the frequency of the words occurring in the corpora is encoded in the *font size* (in Figure 4.3a). However, if the user wants to aggregate the data by genre or year the chart is re-encoded into a *small multiples tag cloud*, in which each selected genre or year have its own tag cloud. In that case, if the user mouses over a given word, a path is drawn connecting all occurrences of that word in the different tag clouds.

To encode the Word Count metric we have considered the use of a *timeline*, in which the *x-axis*
encodes the movie release year and the y-axis encodes the number of words (in Figure 4.3b). Depending upon the user selected type of aggregation, genres are displayed in additional lines with different color hues and the x-axis will be updated for different time spans. If the user mouses over a given line the corresponding area is drawn and the total number of words throughout the selected time span is displayed.

To encode the **Topics** metric we have considered the use of two different charts: a *tag cloud* displayed by default, in which the frequency of the topics occurring in the corpora is encoded in the *font size*; and a *force-directed network* displayed upon user selection of a word, in which *nodes* are topics and *links* represent their co-occurrence (in Figure 4.4). However, once again if the user wants to aggregate the data by genre or year the chart is *re-encoded* into a *small multiples cloud/network*, in which each selected genre or year have its own tag cloud or force-directed network.

Finally, to encode the **Sentiments** metric we have considered the use of an *heatmap*, in which the x-axis encodes movie time, the y-axis encodes each individual movie in alphabetical order and the square color hue encodes each of the five different sentiments, as well as the end of the movie (in Figure 4.5). As mentioned before, each widget has its own set of local configurations, in the case of the Sentiment widget, one of such configurations is the possibility to *derive* an average sentiment for each instant in a movie by *re-configuring* the heatmap (in Figure 4.5b).

To summarize in terms of interactivity, the dashboard supports: *highlighting* of special words such as character names and locations; *filtering* movie genres and release years; as well as *re-encoding* and *re-configuring* some charts. Furthermore, with the exception of the performance view, all other views are *coordinated*, for example if the user selects a square from the heatmap in the Sentiments widget, both the Word Cloud and Topic widgets are updated. The former displays only the words that occur in the selected movie and selected time instant, while the latter displays only the topics from that movie.


## Implementation

### Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.1 System Architecture</td>
<td>35</td>
</tr>
<tr>
<td>5.2 Pipeline Components</td>
<td>38</td>
</tr>
<tr>
<td>5.3 Visual Analytic Tool</td>
<td>51</td>
</tr>
</tbody>
</table>
In the previous chapter, we briefly described B-Subtle Framework pipeline extensions and B-Subtle App first non-functional prototype we proposed to implement in the course of this work. In this chapter, we will provide further details regarding the overall architecture that integrates both the framework and application in a sole product available to the user in Section 5.1, the motivation, problems and solutions used to extend and implement new pipeline components in Section 5.1, as well as the critique to the application first prototype which lead to our final functional prototype in Section 5.3.

5.1 System Architecture

The implementation of the proposed solution resulted in the extension of B-Subtle framework and the development from scratch of B-Subtle App. However, instead of forcing the user to use two complementary products, we have decided to integrate both in such way that the user would only need to use a single product to achieve the same result as both, which lead to the architecture we will describe next.

5.1.1 Overview

The architecture adopted in this work revolves around the fact that the B-Subtle App reuses the B-Subtle Framework to process, filter and gather data from movies and TV show subtitles for the subsequent presentation of statistical information in the form of a visual dashboard. In more technical detail, the B-Subtle App is a cross-platform application available for the most common operating systems – Windows, Mac OS and Linux –, built with top of the stack web technologies – Electron\(^1\), Vue\(^2\), Bootstrap\(^3\) and Highcharts\(^4\) – and encapsulating the most recent version of the B-Subtle Framework v2.0, also developed in the context of this work. A diagram explaining the overall architecture of the application and information flow is presented below, in Figure 5.1.

![B-Subtle system architecture](image)

**Figure 5.1**: B-Subtle system architecture.

---

\(^1\)https://www.electronjs.org/
\(^2\)https://vuejs.org/
\(^3\)https://bootstrap-vue.org/
\(^4\)https://www.highcharts.com/
Next, we will describe in greater detail the reasoning behind the architectural choices as well as provide a deeper understanding of the information flow in the application.

5.1.2 Specifics

The main limitation of the first version of B-Subtle was the nonexistence of a visual interface and strict recourse to the system console. The system console is a text-based interface which requires the user to have prior knowledge of at least basic commands in order to interact with the system. For that reason, and accounting our goal of extending B-Subtle use to a broader audience of cinematographic enthusiasts apart from researchers, it was clear the need to improve it with a visual-based interface by providing a similar look and feel of a system application and, therefore, following a more familiar approach for its targeted audience.

In achieving such goal it was demanded the preservation of B-Subtle as an independent and reusable module that was executed separately from the encapsulating visual interface logic, meaning the application would be in fact emulating the user console interaction, required in its former version, to execute the B-Subtle framework. This posed some constraints on the choice of the development technologies to use in the application implementation due to the needed support on an easy interface for executing external compiled programs, as well as a solution that would be compatible with the technology stack used in the original implementation of the B-Subtle framework.

The B-Subtle framework was initially developed with the Java programming language. A main problem resulting from the initial decision of such programming language was the lack of a greater support needed for the processing and manipulation of natural language data with recourse to lexical and semantic resources available online, such as WordNet. However, the increasing demand on extending the framework to accommodate additional and improved versions of its pipeline components, as means to provide answer to a larger set of researching tasks, was prioritized over its conversion from Java into a NLP-friendly programming language. For that reason, the second version of B-Subtle framework developed in the context of this work, remained implemented in Java, despite the future intention to change its programming language for Python, which offers much support with integrations such as the Natural Language Toolkit (NLTK)\(^5\).

With the decision of maintaining the Java implementation of the framework throughout this work, we began a hunt on finding development technologies compatible with Java in what regards portability concerns. It is known that the Java programming language is portable throughout all platforms with Java Virtual Machine (JVM), so there was a need to look solutions for the application implementation that would not diminish the portability of the core framework and would continue to allow for a seamless user experience in a diversity of platforms.

\(^5\)https://www.nltk.org/
A straightforward solution to the presented problem would be the implementation of a web application which would rely the concerns of portability over all platforms to the system browser and, additionally, would allow the use of the popular and familiar web technologies including HTML, CSS and JavaScript. However, the unavailability of the required in-house web hosting and the strict requirement of an application capable of being installed on the user local machine, resulted on the disposal of such solution.

An alternative solution was the implementation of a desktop application available for the most common operating systems – Windows, Mac OS and Linux –, which could be downloaded as an executable and installed in the user local machine. However, the next challenge would be to find the technology that would allow to build a cross-platform desktop application with the previously mentioned advantage of web development, which consisted in using the familiar HTML, CSS and JavaScript technologies.

With that goal in mind we were able to find the perfect fit for our needs with Electron. Electron is an open-source framework that allows the development of cross-platform desktop applications using web technologies through the emulation of browser behavior with a Chromium engine and a Node.js runtime environment, resulting in a simple, well documented and fully supported interface allowing the control and manipulation of the underlying system shell and file system. In what regards the shell control, it is required to achieve our former goal of being able to execute the B-Subtle framework as an external compiled program. As for the file system manipulation, its usefulness is justified with the fact that the original B-Subtle framework implementation revolves around file-based resources, given its execution starts with a configuration file and ends with the produced corpora and analytical data persisted in a collection of files. Another reason for choosing the Electron framework was its use in real-life popular applications such as Visual Studio Code\(^6\) (code editor for programming and debugging), WhatsApp\(^7\) (messaging platform) and Twitch\(^8\) (live streaming platform) just to name a few.

In addition to the use of Electron, we have decided on the use of several other JavaScript frameworks with the common goal of attempting to use top of the stack web technologies for the creation of a more robust and reliable application. These frameworks include the whole Vue\(^2\) environment with the integration of Vuex\(^9\) and Vue-Router\(^{10}\) (to ensure reactivity between DOM and application state, component-based code organization and support for Single Page Application (SPA)), Bootstrap\(^3\) (to use predefined consistent and stylized components to build the application interface) and Highcharts\(^4\) (to use configuration-based chart creation with support for an enormous diversity of encodings).

Furthermore, the B-Subtle application reuses the most recent version of the B-Subtle framework, version 2.0, also developed in the context of this work and differing from the former in the addition and extension of some pipeline components, as well as analytics (further described in Section 5.2).

\(^6\)https://code.visualstudio.com/
\(^7\)https://www.whatsapp.com/
\(^8\)https://www.twitch.tv/
\(^9\)https://vuex.vuejs.org/
\(^{10}\)https://router.vuejs.org/
The approach followed to integrate the B-Subtle framework in the system architecture was the determining factor of its overall information flow. Since it was required to maintain the framework as an independent building block of the system, we followed the approach of compiling the framework into a single JAR file executed with shell administrative rights by the application whenever a user inputs a configuration file, consequentially defining the starter point of the system flow of information. Once the configuration file is parsed and the subtitles from the OpenSubtitles Corpus are processed, since the behaviour of the core framework when it comes to the format of output results was kept the exact same as of its original version – with a folder per each type of produced corpora and a folder for the analytics module results –, the application then takes advantage of this predefined folder structure to locate, process and render the analytical data into a visual dashboard. In addition, it can be important to note that all the file-based resources created during this process, remain persisted in the user file system for further usage at any given time, under the location predefined in the starting configuration file.

The former summarized in greater detail all the reasoning behind the architectural choices and derived information flow of the presented system, as well as helped acknowledge its two main components that will be our focus of attention in the following sections: the B-Subtle core framework, responsible for the customized creation of subtitle corpora (described in detail in Section 5.2), and the B-Subtle application, responsible for the display of the data gathered by the framework in a visual dashboard (described thoroughly in Section 5.3).

5.2 Pipeline Components

In this section we will be describing in greater detail the recent updates to the pipeline components collection in the newly implemented version 2.0 of the B-Subtle core framework. Those updates include not only extensions of some already existent preprocessing steps, but also the addition of entire new pipeline components to serve new preprocessing steps for the creation of an even more personalized subtitle corpora. Some of those new preprocessing steps are responsible for reducing the volume of the starter subtitle collection to improve time efficiency of the pipeline, others are responsible for gathering more information on the subtitle corpora, while the remaining are responsible for increasing the number of filtering options to choose from when outlining the process for the creation of custom subtitle corpora. Moreover, to be compliant with the requirements of the B-Subtle application, the newest version of the core framework also includes multiple extensions to the analytics module to collect a larger and more diverse volume of data for further statistical analysis in the visual dashboard. These will be described briefly in Section 5.2.1 and thoroughly from Section 5.2.2 to Section 5.2.5.
5.2.1 Overview

The version 2.0 of B-Subtle core framework, among other extensions, integrates new preprocessing steps in the form of the following pipeline components:

- **Subtitle Duplicates Cleaner** – responsible for removing duplicated subtitles for the same movie/TV show and the same natural language from the starter OpenSubtitles Corpus, using metadata information to filter the single subtitle with either the highest user rating in the OpenSubtitles official website\(^{11}\) or the highest custom rating based on completeness and accuracy measures.

- **Frequency Interaction Filter** – responsible for filtering interaction pairs based on the frequency of occurrence of their trigger, answer or both, in the starter OpenSubtitles Corpus per subtitle.

- **Topics Metadata Filter** – responsible for filtering subtitles based on their underlying conversation topics, using the corresponding movie/TV show plot-keywords available online in the TMDb official website\(^{12}\).

- **Topics Interaction Filter** – responsible for filtering interaction pairs based on their underlying conversation topics inferred through the vocabulary present in their composing utterances.

- **Conversation Filter** – responsible for filtering sequences of subsequent interaction pairs neighbouring the ones previously filtered by any filter under the category of Interaction Filters.

Besides those pipelines components, the new version of the core framework also includes extensions to the original analytics modules such as:

- **Frequency Analytics** – responsible for collecting analytical information on the frequency of occurrence of words, utterances, interactions pairs, as well as Person, Location and Organization named entities filtered by the core framework. In addition, it also collects further information on the number of filtered words per genre, per production year and in total.

- **Sentiment Analytics** – responsible for collecting analytical information on the frequency of occurrence of sentiments in the filtered interaction pairs per genre, per production year and in total.

In the following subsections, we will described in detail all the decision and implementations details of these new extensions integrated in the recent version the B-Subtle core framework.

\(^{11}\)https://www.opensubtitles.org

\(^{12}\)https://www.themoviedb.org/
5.2.2 Subtitle Duplicates Removal

The OpenSubtitles Corpus not only covers an enormous diversity of natural languages – including Portuguese, English, Italian, Russian, Korean, Japanese, just to name a few –, but also for the majority of them, it includes subtitles from movies and TV shows spanning a wide period of time – consider the case of the English language, for which it comprises the years since 1880 up to 2018, the equivalent to almost four decades of movies and TV shows. Furthermore, for each natural language and each movie or TV show, in most cases, the corpus includes more than one subtitle – consider a tiny sample of the English corpus with a total of 20 movies, for which there was a total of 341 subtitles, resulting on an average of 17 subtitles per movie.

The combination of these characteristics results in an insanely extensive corpus, for which the majority of the subtitles are duplicates. The presence of duplicates can be misleading in the creation of subtitle corpora for some research tasks – consider frequency analysis, where researchers intend to study the frequency of dialogue turns in daily conversations, if the produced corpora contains duplicates, the frequency of each turn will comprise not only its frequency among subtitles of different movies, but also its unintended frequency among subtitle replicas of the same movie –, and, consequently and unnecessarily, have a bad impact in the time efficiency of the preprocessing pipeline.

Given our ultimate intention to provide as much support as possible in the customization of subtitle corpora for the most diverse set of research tasks and the mentioned possible disadvantages of the presence of subtitle duplicates for some of those tasks, it was clear the need to integrate in the core framework the ability to exclude subtitle duplicates from the starter OpenSubtitles Corpus. In doing so, the next challenge comprised the decision on the adopted method to narrow a group of subtitles into a single subtitle that for the targeted natural language, would ideally be the most complete and accurate representation of the corresponding movie or TV show.

A straightforward method that was adopted and integrated in the latest version of the core framework for the identification of good quality subtitle, is based on real-life user ratings of the subtitles, submitted through the OpenSubtitles official website and included in the subtitle metadata of the OpenSubtitles Corpus – under the tag `<rating>` in conformance with its original XML format.

However, a possible downside to the sole adoption of the previous method results from the fact that it focuses exclusively on the user subtitle rating. Since the users criteria are relative and one user can overlook incompleteness, while other can overlook misspelled words, the user subtitle rating even though somewhat reliable, cannot be considered an absolute measure of the subtitle quality. Therefore, we have considered the adoption of a complementary method, as an attempt for the identification of the most complete and accurate subtitle of a given movie or TV show, based on a machine subtitle rating.
In order to implement an algorithm for the generation of subtitle ratings, we considered a methodical measurement of the quality of a subtitle in what concerns both its completeness and accuracy.

As for completeness, its use as an heuristic is justified with the frequent incomplete number of utterances of most subtitle duplicates, when compared with the number of utterances in the official script of the corresponding movie or TV show – consider a total of 17 English subtitles for the movie Wonder Woman, of which 5 of them are incomplete due to missing the first seventeen utterances of the official movie script. Therefore, to measure the completeness of a subtitle we have used one of the attributes in the subtitle metadata of the OpenSubtitles Corpus, which represents the total number of utterances in the subtitle, – present under the tag <sentences>. The reasoning behind this heuristic is that, the higher the number of utterances the higher the probability that the corresponding subtitle is complete.

As for accuracy, its use as an heuristic is justified with the need to ensure the orthographic correctness of the subtitle dialogues. Therefore, to measure the accuracy of a subtitle we have used two different attributes in the subtitle metadata of the OpenSubtitles Corpus, which represent the total number of spellchecked words and the total number of unknown vocabulary in the subtitle, – present under the tags <corrected_words> and <unknown_words>, respectively. The reasoning behind this heuristic is that, the higher the number of spellchecked words and the lower the number of unknown words, the higher the probability that the corresponding subtitle is accurate.

Each of these heuristics is primarily used individually to obtain a subtitle rating in what concerns each of its linguistic characteristics. Afterwards, the resultant subtitle completeness rating and the subtitle accuracy rating are linearly combined to compute the final unique rating for the subtitle.

Both the previously described user and machine-based subtitle rating methods, used as an attempt to identify the single highest quality subtitle in a group of subtitles for the subsequent exclusion of subtitle duplicates, are available on the core framework in the form of a new pipeline component called Subtitle Duplicates Cleaner, and differentiated through the use of the Basic or Advanced version of that component, respectively.

Furthermore, because this new component can significantly help reduce the volume size of the subtitle corpora, the ideal way to integrate it into the preprocessing pipeline was to strictly add it to its beginning before the application of any other pipeline component, since that would pose a significant increase on the time efficiency of the preprocessing pipeline. Such components that are executed in the beginning of the preprocessing pipeline fall under the new category of pipeline components called Cleaners. In Figure 5.2, we present a diagram explaining a general use case of this newly integrated pipeline component.
5.2.3 Frequent Pairs Extraction

Movie and TV show scripts are written using consciously and subconsciously as reference collected samples of human conversations throughout the lifespan of screenwriters, such that despite constituting a set of predetermined and fictional dialogues, meaning a set of non natural dialogues, they still convey the idea to the audience of spontaneous and legitimate human conversations reflecting their richness of situational contexts. Subtitles are just partial representations of scripts that exclusively focus on their composing set of dialogues preserving their enormous diversity of situational contexts, with the additional advantage of being largely and easily available online. A consequence of the enormous diversity of situational contexts in subtitles is the underlying diversity of dialogues, which differ in various forms being one of them their frequency of occurrence – from our most common daily life greetings such as “Hello!” and “Goodbye!” (present in an infinite amount of movies and TV shows) to rare utterances such as “Ogres are like onions.” (from the movie “Shrek”, 2001).

Given our ultimate goal of supporting an higher number of research tasks and due to the fact that for some research purposes the ability to differentiate between most and least frequent interaction pairs can be useful – consider a CA developed and trained to interact with children, it would be ideal to avoid it being trained with rare interactions such as “Ogres are like onions.” and rather trained with frequent interactions such as “You want me to read you a bedtime story?” (also from the movie “Shrek”, 2001) -, we have implemented in the latest version of our framework a solution that allows to filter interaction pairs based on the frequency of occurrence of their trigger or answer or both simultaneously per subtitle. Because our framework was implemented with the Java programming language and, for that reason, lacks support on some ready to use NLP procedures, for instance available with NLTK for Python, the baseline algorithm implemented is composed of a simple initial iteration, that for each interaction pair increments a global counter of its frequency through a simple string equality check, followed by
the filtering process based on the minimum, maximum or exact frequency conditions specified in the starter configuration file. However, in the case a user intends to filter the interaction pairs based on just their triggers or just their answers, meaning on just one of its composing utterances rather than both, the previous algorithm needed to be extended in order to account for the **Interaction Pair Collapse problem**. As presented in Section 2.1 under Figure 2.1, interaction pairs consist in overlapping pairs of subsequent utterances extracted from the corresponding subtitle dialogue sequence. In more detail, the overlap means that each interaction pair either shares its trigger with the former pair (which occurs with the last pair in the subtitle), or its answer with the later pair (which occurs with the first pair in the subtitle) or both simultaneously (which occurs with all the remaining pairs in the subtitle). The Interaction Pair Collapse problem results from the interaction pairs overlapping nature and the consequent double count of the frequency of occurrence of their composing utterances. With that said, it is important to note that this problem is solely applicable to all the mid utterances in the subtitle dialogue sequence, meaning all the utterances with the exception of the very first and last for which the previous baseline algorithm already functions properly.

In order to solve the above problem, the previous baseline algorithm needed to be extended with a different approach for the specific case of filtering the interaction pairs based on one of their utterances frequency and, simultaneously, solely when filtering the mid utterances. Such extension consisted in the definition of a different frequency count rule for mid utterances that is reduced in half at the end of the counting process.

The extended version of the described algorithm was integrated in the latest version of the core framework in the form of a new pipeline component called **Frequency Interaction Filter** with possible variations not only including filtering based on frequency of pair, triggers or answers, but also filtering based on a minimum, maximum or exact frequency, all provided in the starter configuration file. This new pipeline component was included under the already existent category of Interaction Filters. In Figure 5.3, we present a diagram explaining a general use case of this newly integrated pipeline component.

![Figure 5.3: Frequent Pairs Filter diagram](image)

*E.g. the starter subtitle corpora includes a subtitle with 3 dialogue turns each with distinct frequencies, through the application of the minPairFrequency version of this Filter we are able to reduce the collection to the turns with frequency equal or greater than 10.*
5.2.4 Movie Topic Filtering

As described in the beginning of the previous subsection, subtitles are composed of dialogues that resemble human spontaneous and legitimate conversations extracted from movie and TV show scripts and, for that reason, preserve their variety of situational contexts. A consequence of the extensive variety of situational contexts present in subtitles is the insane multitude of underlying conversation topics spread throughout their numerous dialogues – including interactions during poker games, discussions on military strategies during war crisis, debates during criminal trials, conversations on gastronomy, just to name a few.

While the originally inherited diversity of conversation topics in the subtitle dialogues can be an advantage for some open-domain research tasks without requiring further preprocessing, for target domain research tasks it can constitute a problem – consider a CA whose target domain is a card game such as poker, only subtitle dialogues that are related with poker and, ideally, occur during poker games would be useful in its training. Because our framework attempts to support a more diverse set of research tasks and to offer a more complete catalogue of preprocessing steps from which researchers can create their customized and optimized combinations according to their specific task requirements, we attempted to provide a solution for the target domain research tasks in the latest version of our framework. This solution allows to filter subtitles or subtitle dialogues based on the underlying conversation topics, as means to refine the produced subtitle corpora to a specific topic or set of topics.

In the development of such solution, we had to account for the possible requirement of different degrees of precision in what regards the filtering of subtitles for the occurrence of specific conversation topics, which, consequently, determined the need to adopt two distinct topic filtering methods. A coarse topic filtering of subtitles, which results in a set of dialogues that despite including the required conversation topics still have a high probability of additionally including many others, and a refined topic filtering of subtitles, which results in a set of dialogues that ideally include exclusively the required conversation topics. The former was achieved in this work with movie-based topic identification and constitutes the subject matter of the present subsection, while the latter was achieved with dialogue-based topic identification and will constitute the subject matter of the proceeding Section 5.2.5.

The movie-based topic identification consists in the identification of topics based on the metadata of a movie or TV show and, for that reason, it is subject to a much more generalized topic classification made by some individuals in their audience and resulting from their digestion and categorization of various specific conversation topics into a reduced group of generic ones. Despite the fact that those human made topic classifications have the disadvantage of being relative to an individual knowledge base and interest of topics potentially leading to incomplete classifications, they simultaneously have the advantage when compared with machine made classifications, of being enriched with the individuals
perception of the entire scenery, plot, morals, emotions, as well as cultural backgrounds and beliefs. Moreover, those human made topic classifications are abundantly available online, scattered throughout an enormous variety of movie and TV show related websites by the distance of a network request.

Having decided on the use of human made topic classifications for the coarse filtering of subtitles based on topics, we first had to find the means to access such classifications online through our framework and later define the entire logic used in the filtering algorithm while accounting for some underlying research problems. Because the OpenSubtitles Corpus does not include metadata information on a generic list of conversation topics discussed throughout each subtitle and to make use of the fact that the previous version of the framework already included the component themoviedb – a metadata collector based on the information available online on the TMDb -, we decided to extend the latter to further request for the above mentioned human made topic classifications, which from this point on will translate into movie or TV show plot-keywords. However, the original implementation of the metadata collector themoviedb used a Java wrapper to access the TMDb REST API – the themoviedbapi by Holger Brandl -, which at the time of implementation presented some issues in the ability to retrieve the list of plot-keywords for a given movie or TV show based on its IMDb ID. As a consequence, after further investigation, we were able to find a complementary TMDb API wrapper for Java – the api-themoviedb by Omertron – that met our requirements and was also presented in the official online documentation of the TMDb API.

The next challenge was to implement a topic filtering algorithm that defined the logic on how to use the topics or list of topics provided by the user, in the starter configuration file, to filter subtitles using as reference their corresponding list of plot-keywords, previously retrieved with the themoviedb metadata collector. A straightforward approach that defined the baseline for the algorithm integrated in the latest version of the core framework, consisted in a simple string equality check. However, given the fact that TMDb is a community-built database that offers the contributors the freedom to assign any relevant plot-keywords to a given movie or TV show, which means contributors can use different plot-keywords to refer to the same conversation topics. Such simple approach posed the following research problems:

- **Compound Noun problem** – this problem occurs when for instance a user intends to filter subtitles from war-related movies providing as input the topic war and the user targeted set of movies, composing part of the starter subtitle corpora, only included plot-keywords such as cold war, colonial war, world war I and war hero. In this case the simple string equality check algorithm would not accept those subtitles, returning an empty match for the user search. We found a solution that worked correctly for both cases when the user searched for a single noun and for compound nouns and that kept the simplicity of the baseline algorithm with the sole replacement of the equality check

---

13 https://github.com/holgerbrandl/themoviedbapi/
14 https://github.com/Omertron/api-themoviedb/
15 https://www.themoviedb.org/documentation/api/wrappers-libraries
with inclusion check. In more detail, the algorithm checks if the string corresponding to a subtitle plot-keyword contains the string corresponding to the topic provided by the user as input. This way, the algorithm when searching for the topic war would filter the subtitles with plot-keywords war or world war I, but when searching for world war I would no longer accept the subtitles with plot-keyword war, only with world war I.

- **Inflected and Derived Word Forms problem** – this problem occurs when for example a user intends to filter subtitles from finance-related movies providing as input the topic finance and the user targeted set of movies only included as plot-keywords inflected forms of that word, such as finances, financing and financed. In this case, the simple string equality check algorithm would not accept those subtitles, returning an empty match for the user search. To solve this problem we resorted to a NLP normalization technique and enforced prior stemming for both the user input and the subtitle plot-keywords, reusing the OpenNLPStemmer producer already implemented in the original version of the core framework. This way, the algorithm would not check for string equality, but rather for words stems equality.

As a result of the solutions found for each of the above research problems, the initial version of the topic filtering algorithm consisting in a simple string equality check, was improved into a word stem inclusion check. Such algorithm allows a user, for instance, to provide as input just the topic work and be able to filter all subtitles that not only include explicitly the plot-keyword work, but in addition the ones that include as plot-keywords inflected forms of that word, as well as compound nouns, such as works, team work, working, work ethic, just to name a few. However, a user might be interested in filtering subtitles not only based on a single conversation topic but multiple ones. Because the user could have two distinct intentions of including at least one of the topics provided as input or all of the topics provided as input simultaneously, we have additionally extended our algorithm to support multiple topic filtering offering the user the possibility to specify, in the starter configuration file, the intention to combine topics using either logical disjunction or logical conjunction (which in programming language typically are referred as the logical operators OR and AND, respectively).

Furthermore, at the time of the dissertation proposal there was an important limitation on the above described solution which consisted in its dependency on the TMDb API and its usage restriction of a maximum of 40 HTTP requests every 10 seconds. However, as of December 16, 2019, the TMDb has disabled the request rate limit, meaning from that point on there was no usage restriction on the number of HTTP requests made to their API and so, there is no such further limitation on our solution. This alteration was reported in an official announcement on the TMDb developers website.\footnote{https://developers.themoviedb.org/3/getting-started/request-rate-limiting}

The previously described algorithm for the coarse filtering of subtitles based on movie and TV show metadata conversation topics, also referred throughout this subsection as plot-keywords, is available on
the B-Subtle core framework in the form of a new pipeline component called Topics Metadata Filter. This pipeline component requires as input a topic or list of topics to filter the subtitles – listed under the parameter named values –, and, additional and optional parameters for the specification of the user intention to consider words stems – to solve the inflected word form problem and available as the Boolean parameter named stemming –, as well as to consider disjunctive of conjunctive topic combinations – to solve the multiple topic problem available as the String parameter named valuesUnifier. It is important to note that this new pipeline component requires the prior use of the themoviedb metadata collector and, in the specific case of the option that utilizes the stemming normalization technique it also requires the prior use of the already available OpenNLPStemmer producer. This pipeline component was introduced as a new component in the already existent category of Metadata Filters. In Figure 5.4, we present a diagram explaining a general use case of this newly integrated pipeline component.

![Figure 5.4: Movie Topic Filter diagram. E.g. the starter subtitle corpora includes 2 subtitles with distinct set of plot-keywords, through the application of this filter we are able to reduce the collection to 1 subtitle that contains all compound, inflected and derived forms of the words “crime and” “computer” simultaneously.](image)

5.2.5 Pair Topic Filtering

As described in the previous subsection, subtitle dialogues include a multitude of conversation topics which for target domain research tasks can constitute a problem. For that reason and given our attempt to support an higher number of research tasks, in the latest version of the B-Subtle core framework we have integrated a solution to filter subtitles and subtitle dialogues based on a specific topic or set of topics provided by the user. This solution offers a coarse topic filtering of subtitles, obtained through movie-based topic identification described in the previous Section 5.2.4, and a refined topic filtering of subtitles, obtained through dialogue-based topic identification that will constitute the subject matter of the present subsection. The dialogue-based topic identification consists in the machine identification of topics based on the set of terms occurring in each dialogue, also know as interaction pair and meaning
occurring either on its corresponding trigger or answer. As a result of the implementation of this dialogue-based topic identification in the latest version of the core framework, by providing as input a list of topics such as trial and judge, a user can for instance filter the interaction pairs that include explicitly the term trial and judge, but also the ones that include the terms trial judge, trialling, trial court, judge advocate, trialled, judges, just to name a few.

Because this topic identification shares the same compound noun, inflected words form and conjunctive or disjunctive topic combination problems described in the previous subsection for the newly added pipeline component Topics Metadata filter, its underlying algorithm logic is precisely the same of a word stem inclusion check with possibility to search for interaction pairs including all topics or at least one of them in case a list of topics is provided as input. However, the relevant difference between them is that the Topics Metadata filter relies on movie-based topic identification, meaning the filtering of subtitles based on human made topic classifications available on their metadata that are relative to their entire set of interaction pairs, while the topic identification presented in this subsection allows filtering subtitles based on the most basic form of machine made topic classifications relative to each specific interaction pair in the subtitles.

The previously described dialogue-based topic identification algorithm is available on the core framework in the form of a new pipeline component integrated in the already existent category of Interaction Filters under the name Topics Interaction Filter. Given their similarity, the further customization of this pipeline component relies in precisely the same set of parameters as the Topics Metadata Filter – the values, for the user input topic or list of topics, the valuesUnifer, for the topic combination to use when a list of topics is provided, the stemming, for the decision on considering the term stem (when set to true) or the original term (when set to false) when filtering, as well as the tokenModelFile, for the specification of the language-dependent OpenNLP tokenization model\(^\text{17}\). In Figure 5.5, we present a diagram explaining a general use case of this newly integrated pipeline component.

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{pair_top_filter.png}
\caption{Pair Topics Filter diagram. E.g. the starter subtitle corpora includes a subtitle with 4 dialogue turns with distinct vocabulary sets, through the application of this filter we are able to reduce the collection to 2 turns that contain a derived form of the word “cook”.
}\end{figure}

\footnotesize\(^{17}\)http://opennlp.sourceforge.net/models-1.5/
5.2.6 Conversation Filtering

With the addition to the B-Subtle core framework described in the previous Section 5.2.5, we were able to offer the possibility to produce a subtitle corpora that includes dialogues with terms commonly used in a specific conversation topic. However, the exclusive use of the presented solution can be highly restrictive if the user input list of topics is limited and does not provide full coverage of the diverse and extensive vocabulary of the targeted conversation topic, resulting in a subtitle corpora that is missing dialogues that could potentially be addressing the same conversation topic, but using different terms from its vocabulary – consider for instance the need to produce a subtitle corpora rich in poker interactions and using for topic filtering the term *poker*, the resulting subtitle corpora would be missing all the interactions that included terms such as *game*, *cards*, *player*, *all-in*, *call*, *chip*, just to name a few, but not *poker*. In other words, the exclusive use of the Topics Interaction Filter allows to filter individual dialogues related to a specific topic, but not an entire conversation regarding that topic.

A conversation can be defined as a sequence of dialogues exchanged between two or more interlocutors about a particular subject, which in the context of our work translates into the technical definition of a sequence of interaction pairs composing part of the entire sequence of dialogues in a subtitle that regard a specific topic. From the various possible solutions for the presented conversation filtering problem, we have sketched two distinct approaches: a conversation vocabulary-based filtering solution and a length-based filtering solution. The former attempts to minimise the limitation of the initial list of topic terms provided by the user considering in addition their semantic and lexical variations, while the latter attempts to determine the conversation length in order to extract the sequence of interaction pairs adjacent to the pair previously filtered by the Topics Interaction Filter. The former presented some disadvantages that lead to the sole integration of the latter in the latest version of the core framework, however next we will describe both in detail.

A straightforward implementation for the vocabulary-based filtering approach, that could potentially lead to the initial list of topic terms provided by the user covering a wider range of terms composing the targeted conversation topic vocabulary, would be based on the usage of WordNet synsets to additionally consider the semantic and lexical forms of the initial topic terms, including their hypernyms, hyponyms and domain terms. A possible downside to the adoption of that approach would result from the fact that the B-Subtle core framework is implemented in Java, which unlike the Python programming language does not offer the same support optimized for NLP tasks (as described in Section 5.1.2), lacking the existence of an integration such as the NLTK Corpus that comprises the WordNet lexical database by default. As a result of this lack of support, the alternative would be to use the WordNet Rest API and perform an HTTP request for every conversation topic term provided as input. However, not only this would constitute an additional external dependency that increases the preprocessing execution time with response time of every HTTP request made to the WordNet API, but also, since we would be considering
all the term hypernyms, hyponyms and domain terms, would not be scalable for big samples of the OpenSubtitles Corpus – just its English version comprises a total of 370K subtitles corresponding in a staggering 60GB sized sample.

The alternative length-based filtering approach that also aims to solve the possible disadvantage of the exclusive use of Topics Interaction Filter, does not attempt to complete the initial list of topics provided by the user as means to cover the targeted topic vocabulary prior to the application of the Topic Interaction Filter, but rather uses the accepted interaction pairs resulting from the application of that filter to define hooks in the original subtitle in order to find their neighbouring pairs and extract sequences of interaction pairs that ideally belong to the same conversation and regard the same topic. We have considered the hooks neighbouring set of interaction pairs to be parted half before and other half after the corresponding hook and the total quantity of pairs composing each half to determine what in the context of our work we have defined to be the conversation length for simplicity purposes – consider the Topics Interaction Filter filtered a single interaction pair in an entire subtitle, translating into a single hook, the further use of this length-based filtering approach with a conversation length set to 2 would be able to additionally filter the sequence of two pairs occurring before and the sequence of two pairs occurring after the previously filtered hook.

As mentioned before, the previously described length-based filtering approach is available in the latest version of the core framework in the form of a new pipeline component called **Conversation Filter**. This pipeline component requires the user to input the intended conversation length – defined before and available as a Numeric parameter named *length* – and additionally offers the user the ability to decide on how to integrate its filtered neighbouring interaction pairs in the final produced corpora, with the possibility to either include them explicitly in the final corpora along with the previously filtered interaction pairs or include them implicitly as attributes of the pairs corresponding to the hooks – available as a Boolean parameter named *include*. In Figure 5.6, we present a diagram explaining a general use case of this newly integrated pipeline component.

**Figure 5.6: Conversation Filter diagram.** E.g. this filter is applied to the corpora resultant from the previous application of the Pair Topics Filter and with it we are able to extract 2 additional neighbouring interactions occurring immediately before and after the ones previously filtered.
It is important to note that despite the fact that the new pipeline component presented in this subsection, the Conversation Filter, was created as means to solve a possible disadvantage for some research tasks of the newly integrated Pair Topics Filter, described in the previous Section 5.2.5, it can in fact be used as a complementary preprocessing step for any other Interaction Filter or even combinations of Interaction Filters.

5.3 Visual Analytic Tool

As described in the previous section, in this work we have extended the original version of the B-Subtle core framework to integrate additional preprocessing steps to support a wider range of research purposes. However, besides this extension of the core framework, in this work there was the additional requirements to help researchers in using a visualization as part of an incremental process of discovering the optimal preprocessing pipeline according to their research requirements, as well as to expand the usage of the framework to a broader audience not only researchers and not strictly familiar with operating a system console. Both these conditions for audience expansion were determinant in the process of deciding and implementing a complementary interface for the sole console-based interface available in the original version of the framework, leading to the development of a visual dashboard interface rendered by a framework-wrapper application, which will be the subject matter of the present section and which we refer to as B-Subtle App.

As mentioned in Section 5.1, the B-Subtle App was developed as a desktop application available for Windows, Mac and Linux operating systems, built with Electron, Vue, Bootstrap and Highcharts web technologies and wrapping the most recent version of the B-Subtle framework. Furthermore, the purposes of this application consist in being an optional alternative to the console-based interface, offering the ability to receive a configuration file and use it to execute in the background the B-Subtle framework process, as well as present statistical information regarding the subtitle corpora produced with the help of the framework in the form of a visual dashboard, using state-of-the-art Information Visualization techniques. It is important to note that in order for the B-Subtle App to have data to populate the different idioms presented in its dashboard, the framework besides being responsible for the implementation of the preprocessing pipeline is also, and simultaneously, responsible for collecting and storing statistical data gathered during the preprocessing over file resources, further parsed and processed by the B-Subtle App. As described in Section 2.1, this additional responsibility of the framework is ensured by its analytics modules, which gather statistical data consisting in a variety of metrics related not only with the execution of the preprocessing pipeline, but also with the produced subtitle corpora since the integration of the new Frequency and Sentiment analytics modules in the recent version of the framework (which are described briefly in Section 5.2.1) – such metrics include the total execution time of the pipeline,
the average execution time per subtitle, the average number of interactions pairs per subtitle, the number of subtitles and interaction pairs accepted per each filter in the collection of pipeline components, the frequency of occurrence of words, utterances, interaction pairs, as well as Person, Location and Organization named entities, the number of interaction pairs per sentiment, just to name a few.

In order to ensure the B-Subtle App fits the requirements of its targeted audience, which, since the implementation of the proposed solution in this work, includes not only researchers, but also general public falling under the category of cinematographic enthusiasts, during its development we followed a top-down approach. Following this approach, we first focused on gathering a clear definition of the tasks and questions the application visual dashboard needed to answer, followed by the decision and implementation of the idioms that attempt to provide answer to those exact questions.

To achieve the former gathering of tasks and questions, we have conducted an online survey to samples of the two distinct user groups our application is aiming to target and which included members of the HLT community\textsuperscript{18} and students with strong interest in movies and TV shows from IST university\textsuperscript{19}. The fourteen survey participants were asked to suggest new preprocessing steps useful to produce custom subtitle corpora, as well as to suggest statistical information on the produced corpora from which they would benefit the most if presented in the visual dashboard. These suggestion requests were formulated into the open-ended non-technical questions “What filters would you like the tool to support?” and “What statistics would you like the visual interface to support?”. This survey allowed us to gather a list with a total of 46 suggestions, from which some were mapped into new pipeline components integrated in the recent version of the B-Subtle framework (described in Section 5.2) and others were considered during the development of the B-Subtle App, however from those, only some, depending on their conformance with existent limitations of the application implementation, were mapped into idioms and idiom filter options available in its visual dashboard. As depicted in Figure 5.7, the top three most suggested tasks were character, sentiment and topic analysis. While the latter two were integrated in the B-Subtle App in the form of two distinct idioms, the former constituted an example of a suggestion not compliant with our application limitations. Since our application works with the OpenSubtitles Corpus, a corpus of enriched subtitles, however still lacking some relevant information for instance dialogue-based character identification, such suggestion would translate into the complex and extremely time-consuming task of adding an entire new type of pipeline to B-Subtle framework specific and optimized for script corpus. It is important to note that the survey participants constituted a very small sample of the application targeted audience, reason why besides considering the survey participant suggestions we have additionally considered all the tasks and questions researchers were attempting to answer in the course of their works presented in Chapter 3, in the course of both the extension of the B-Subtle Framework and the development the B-Subtle App.

\textsuperscript{18}https://www.hlt.inesc-id.pt/
\textsuperscript{19}https://tecnico.ulisboa.pt
Once we have gathered a sketch list of possible requirements to integrate in the B-Subtle App, the next logical step in our top-down approach was to decide which of those requirements were compliant with the application limitations and constituted the most value to both user groups the application is aiming to target. This resulted in the definition of the final list of statistical data needed to be collected by the B-Subtle Framework (presented in Table 4.2 and Table 4.3) and the final list of tasks and questions the B-Subtle App will attempt to provide answer using those framework collected statistics (presented in Table 4.4).

In the next subsections, we will be describing in detail the subsequent steps followed in our top-down approach, which consisted in an incremental prototyping process that resulted in the final version of the B-Subtle App.

### 5.3.1 First Prototype Critique

The first non-functional prototype of the B-Subtle App consisted in the customizable dashboard presented in Figure 4.1. This first prototype was composed not only by (i) a set of 9 draggable, detachable and removable widget-based idioms, grouped into four major views – performance, words, topics and sentiment –, and encoding the statistical data previously gathered and stored by the B-Subtle Framework, but also by (ii) a control panel including a search box, a widget toggling section, a widget configuration section and a help section with reset, tutorial and about features. A more thorough description on the various composing elements of this prototype is presented in Section 4.2.3. This prototype defines the baseline for the subject matter of the present subsection in what regards the critique to some of its underlying conceptual decisions and further exploratory work leading to the implementation of the final functional prototype described in the following subsection.

One of the main critiques to the B-Subtle App first prototype was the nonexistence of an initial plan to integrate it with the B-Subtle Framework, resulting in the offering of two decoupled programs the users had to execute separately, but sequentially, to achieve their research purposes and, for that reason, lacking the integrity and robustness that, on the opposite side, could be achieved with a single application responsible for mirroring all the functionalities of the underlying framework and presenting the additional...
statistical visualizations simultaneously.

In more detail, the first prototype assumed the users would start by downloading and executing the B-Subtle framework program, to produce customized subtitle corpora and generate file resources in JSON format with statistical data gathered during preprocessing, followed by the users clicking on an entry file named index.html, previously downloaded alongside the framework, to open the visual dashboard on the system default browser and, consequently, trigger the event of parsing and reading the statistical data from the JSON files, previously created by the framework. However, as mentioned before, such assumption would lead to the existence of two decoupled programs that needed to be executed separately. Taking into account this disadvantage of the first prototype, besides deciding on a new system architecture that allowed the B-Subtle App to wrap the B-Subtle Framework executing it automatically in a background process (described in detail in Section 5.1), we had to additionally decide on how to incorporate in the first prototype a new preceding page flow that would start with a landing page, offering the user the ability to choose the starter configuration file through the file system explorer, followed by a loading page, presenting the progress in the execution of the framework, and finishing in the already sketched dashboard page (presented in Figure 4.1).

Regarding the landing page, there was the intention to provide an alternative way for the creation of the starter configuration file embedded into the application, since by default such creation should be done prior by the user in any preferred text editor. An example of a sketch for such landing page version is presented in Figure 5.8. However, due to the fact that the previous version of B-Subtle had been used by researchers in the HLT community\textsuperscript{18} which had already familiarized with the idea of creating the starter configuration file in a system text editor separately from the framework, to preserve recognition and minimize preattentive processing [Munzner, 2014], we have decided to not include this alternative form of configuration file creation in the new version of the B-Subtle App. The previous decision resulted in the landing page presented in Figure 5.9a, however after some thought we realised that with the sole offering of the means to choose the starter configuration file from the file system explorer, if the user intended to simply analyze the statistical visualization over an already produced corpora, it would be obliged to execute the framework and produce the exact same corpora once again. For small samples of the OpenSubtitles Corpus, the former would not constitute a critical problem, however big sized samples depending on the complexity of the user-defined pipeline could result in a couple preprocessing hours, not being time efficient if forced to duplicate. With that in mind, we decided to renew the sketched landing page to include an additional option for the user to provide as input the folder named analytics, containing all the statistical data gathered and generated in the first framework execution of the corresponding produced corpora, and, once more, chosen through the file system explorer. This new landing page sketch is presented in Figure 5.9b.
The next critique to the first prototype is due to the fact that at the time we had not completely determined the exact subject of the visualization, if either data regarding the starter subtitle corpora or data regarding the final produced corpora. We have opted for the visualization to target the final produced corpora since that would be more beneficial to researchers, provided they could use that visualization as part of an incremental process of discovering the optimal preprocessing pipeline according to their research requirements, through iterative evaluation of the produced corpora. Besides, setting the subject of the visualization to be the final produced corpora would not exclude the possibility of still being able to analyze the starter subtitle corpora, since the latter is the former when a user simply defines an empty preprocessing pipeline in the starter configuration file. However, this decision came with the price of statistical data heterogeneity.

In the first prototype we have sketched a solution that proposed multiple coordinated views, meaning if the user selected a data instance in an idiom, the same would be highlighted in all the remaining idioms. Such coordinated behaviour would only be possible if all idioms shared the same data instances. However, as mentioned before due to our previous decision, we now had to account for the heterogeneity of the statistical data gathered by the framework during preprocessing. This heterogeneity results not only from the fact that this data can be grouped into four major views either performance, words, topics or sentiments, including data instances ranging from execution time, files, filters, dialogues, words, topics...
up to sentiments, but also and most importantly, from the fact that the final produced corpora constitutes a collection of fragments from the starter subtitle corpora, which in most cases would not be able to reflect the evolution of data using its continuous attributes, such as period of time and movie duration, as expected initially – for instance with the proposal of the sentiments heatmap in Figure 4.5 and the words and topics co-occurrence in Figure 4.3a and Figure 4.4 respectively –, resulting in various multiple missing data gaps in the idioms. Therefore, with such high heterogeneity of the statistical data we have opted to not implement the initially desired multiple coordinated views pattern into the visual dashboard.

Another critique to the first prototype that can be considered as another cause-effect of the heterogeneity of the statistical data, regards the placement of the control panel specific configuration section.

The control panel configuration feature offered alternative scopes for the aggregation of the data in the idioms. Such scopes varied between global, aggregating the data based on the same attribute in all the dashboard idioms, and specific, aggregating the data based on a given attribute solely in the currently selected idiom. Furthermore, the supported aggregation methods varied according with the available data attributes. Once an aggregation method was selected or deselected, it would trigger re-encodings and/or re-configurations of the corresponding scoped idioms, in response to the adjustment of their underlying set of attributes.

Since most of the idioms share the attributes movie genre and production year, these were both considered for aggregation methods with global scope. Such global aggregation methods would not span for the remaining idioms that did not share the attributes genre and year, and which mainly consisted in the idioms falling under the performance view category, related with the execution of the preprocessing pipeline in what regards counting time, files, filters and dialogues. Despite the fact that the previous meant these global aggregation methods were not so global after all, since they were still the only aggregation methods shared among a group of idioms and there is the possibility that such group will scale in future implementations, their placement in the control panel outside of each corresponding idiom was not questioned.

However, opposing to the reasoning behind the decision of placing the global aggregation methods in the control panel, was the first prototype placement of the alternative specific aggregation methods to each idiom, also in the control panel. For that reason we considered an alternative placement of the specific aggregation methods next to the corresponding idioms as depicted in the below image.
Figure 5.10: Parallel Tag Clouds. Compare word frequencies between movie genres with selected words police and question.

Figure 5.11: Wordle. Words positioned horizontally and vertically for efficient usage of idiom area and to support big data sets.

(a) Word frequencies.

(b) Named entities frequencies.

Figure 5.12: Tag Cloud.

(a) Word frequencies.

(b) Named entities frequencies.

Figure 5.13: Packed Bubble Chart.
In the first prototype we have proposed to encode the Word Cloud and Topics metrics, corresponding to the frequency of occurrence of both words and topics in the dialogues of the produced corpora, using tag clouds. If the user intended to compare such frequencies of occurrence between movie genres and production years, it could aggregate the data by those attributes using the global configurations, which would trigger the re-encoding of the idiom into the small multiples tag cloud presented in Figure 4.3a and Figure 4.4. This brings us to the next critique, which concerns the scalability of this small multiple tag cloud. Provided that each movie can have multiple genres, that there is a total of 28 possible genres and that, just for the English version of the OpenSubtitles Corpus, there is an equivalent to almost four decades of movie production years, according with Weber's Law [Fechner, 1966] in such extreme cases it would be hard to compare the frequency of occurrence between tag clouds farther away from each other. Therefore, the small multiple tag cloud solution to compare the frequency of occurrence between genres or years was not scalable and we had to consider an alternative solution consisting in the use of parallel tag clouds presented in Figure 5.10, in which each column would encode in alphabetical order the words/topics composing the fragment of the produced corpora for a specific genre or year and the frequency of occurrence of those words/topics would be encoded through the font-size. These parallel tag clouds would solve the problem of comparing words/topics occurring in tag clouds farther apart provided once selected they would be connected with edges colored distinctively. However, this encoding of words occurrence across different columns through color hue would conflict with the further encoding of the words/topics categorical attribute corresponding to their named entities classification – persons, locations and organizations –, also through color hue, which was planned to be included in the format of a specific configuration since the first prototype. Given that this high dimensionality of the underlying data, caused by the high number of attributes including frequency of occurrence, genre, year and named entity type, hardened the task of finding an idiom able to encode all those dimensions simultaneously, we have decided for now to delay the encoding of the genre and year attributes for a future version of the application. With that said, these idioms no longer supported the aggregation of data by genre and year attributes through the global configurations, remaining solely the encoding of word frequency through the tag cloud and the encoding of different types of named entities through the re-configuration of such idiom using the color hue channel, which lead to the sketches show in Figure 5.12 and Figure 5.13. Another characteristic of the underlying data that had to be accounted for during those sketches, besides its dimensionality was the volume of the set of words and topics present in the subtitle dialogues, which just for a small sample of 12 movie subtitles can exceed 46K instances. To solve such problem, we considered the replacement of the tag cloud with a wordle to position the instances both horizontally and vertically for a more efficient use of the idiom area, as shown in Figure 5.11. However, such solution could lead to the perception of the orientation as a way of encoding an extra attribute and, for that reason, violating the Expressiveness Principle [Mackinlay, 1986]. Instead
we have opted to follow a similar approach to Mitchell WordCloud [Schwarz et al., 2017] by restricting the volume of data instances set in the tag cloud to a maximum of 100 instances, with the slightest difference that in addition we intend to offer the possibility to choose between the top 100 most frequent and the top 100 least frequent, in the form of specific configurations for the corresponding idioms.

Another critique to the first prototype is due to the fact that we have proposed to encode the Word Count metric, corresponding to the total number of words present in the dialogues of the final produced corpora, using the timeline presented in Figure 4.3a. The use of such idiom not only assumed that the global configuration corresponding to the production year would always be toggled on without a possible solution for when just the movie genre global configuration is toggled on, but also assumed a continuous progression on the number of words throughout the user selected time span. However, the OpenSubtitles Corpus time granularity is limited to production years resulting in a discrete progression of time. Therefore, to avoid violating the Expressiveness Principle [Mackinlay, 1986] by misleading the...
user into considering a value between two given years as a valid value for the total number of words in the middle of the first year, we have considered the replacement of the timeline with a grouped bar chart presented in Figure 5.14, in which each group of bars represented the number of words in a given year and each bar represented the group encoded the number of words in a given genre for a given year, with the years being encoded in the x-axis position and the genres both in color hue and x-axis position within a group. When the user toggled off one of the global configurations remaining either just the year or just the genre, the idiom would suffer a re-encoding into a simple bar chart presented in Figure 5.16. However, soon it became evident the non-scalability of such solution when all global configurations where toggled on, which once more was caused by the fact that a movie can have multiple genres and there is a total of 28 possible genres, resulting in difficulty to encode a high number of genres with the color hue channel. With that said, we considered a different re-configuration of such idiom which consisted in swapping the channels between the year and genre attributes resulting in the idiom presented in Figure 5.15, where the genres where no longer encoded with color hue, but rather the years with color saturation.

![Figure 5.17: Divergent Stacked Bar Chart.](image1)

![Figure 5.18: Stacked Bar Chart.](image2)

(a) With only global configuration genre on.  
(b) With only global configuration year on. 

![Figure 5.19: Grouped Bar Chart.](image3)
The last critique to the first prototype concerns the idiom proposed to encode the Sentiments metric presented in Figure 4.5 and results from the previously mentioned decision on having the visualization target the final produced corpora rather than the starter subtitle corpora, and, consequently, from the heterogeneity of the statistical data. The proposed idiom consisted in a heatmap and, once more, assumed the continuous progression of dialogue sentiments throughout the movie duration. However, since the final produced corpora can include just a small fraction of the dialogues present in the original subtitle of a given movie, it could potentially result in the idiom having multiple grey-colored gaps due to missing multiple dialogues and, corresponding, sentiment classifications. Given this possibility of missing a big fraction of the underlying data if we considered the encoding of its movie duration attribute, we have decided to exclude such attribute from the encoding. Furthermore, the proposed idiom also considered a movie as a data instance, however provided the framework limitations this was also not possible to achieve. So, given both the removal of the movie duration attribute as well as the limitation on having a movie as data instance, the idiom no longer encoded a movie sequence of sentiments throughout its duration and, consequently, no longer provided answer to the initially planned task of summarizing sentiments evolution during a screenplay. Therefore, we had to reconsider the task and respective questions we could provide answer to with this sentiments metric, which resulted in the representation of the frequency of dialogue turns sentiments in the final produced corpora, providing answer to Task 7 in Section 4.2.2 and encoded through the divergent stacked bar chart presented in Figure 5.17. This divergent stacked bar chart seemed ideal given the divergent nature of sentiments with the Neutral sentiment defining the vertical baseline and the Very Negative and Very Positive sentiments defining the extremes to the left and right, respectively. However, in this idiom apart from the bars corresponding to the Neutral sentiment, all the remaining ones would prove to be hard to compare given they are not aligned and, for that reason, violating Weber’s Law [Fechner, 1966]. With that said and since a stacked bar chart would have the exact same non-alignment disadvantage as the divergent stacked bar chart as shown in Figure 5.18, we have opted instead for the grouped bar charts presented in Figure 5.19, where all the bars encoding each sentiment are aligned relative to the same baseline.

5.3.2 Functional Prototype

As mentioned earlier in this section, the B-Subtle App was developed as an optional alternative to the original console-based interface of the encapsulated framework, instead offering a visual-based interface providing the same experience as of any other desktop application installed on the user local machine. The purpose of this development is justified with its vital contribution not only in attempting to help researchers during the incremental process of discovering the optimal preprocessing pipeline through the iterative evaluation of the final produced corpora, but also in allowing to extend the usage of B-Subtle to a broader audience of cinematographic enthusiasts apart from researchers.

61
In the previous subsection we have used the B-Subtle App first non-functional prototype as a baseline to build a stream of constructive critiques of its underlying conceptual decisions and to explore alternative solutions, with the sole intention of improvement and implementation of the final functional prototype of the application, which will constitute the subject matter of the present subsection, where we will be providing a thorough description of its pages, components, widgets, as well as corresponding idioms and configurations.

The B-Subtle App final functional prototype is initiated with the landing page presented in Figure 5.20. In this page the user is prompted to choose between preprocessing a user-defined sample of the OpenSubtitles Corpus to create new personalized corpora with further analysis, corresponding to the Configuration File option, or, alternatively, simply analyse personalized corpora created beforehand, corresponding to the Analytics Folder option. In the former, the user is required to upload the starter configuration file, afterwards the application will navigate to the loading page presented in Figure 5.21, which displays the progress in the execution of the preprocessing pipeline by the framework and, once the execution is over, displays a success or failure message with the possibility to open the file system.
explorer on the folder containing the recently generated corpora, however depending if the user-defined pipeline included the analytics component or not, the application either allows the user to proceed to the analytics dashboard or displays a slideshow presenting the further possibility to analyse the produced corpora, respectively. As for the latter, the user is required to upload the entire analytics folder generated beforehand by the framework during the preprocessing and creation of a given personalized corpora and, afterwards, the application will immediately navigate to the analytics dashboard, which is presented in the following image.

![Figure 5.22: B-Subtle App analytics dashboard.](image)

In comparison with the first prototype and further sketches described thoroughly throughout the previous subsection, in general the final functional prototype has preserved the concept of a customizable dashboard, composed by a left side control panel and a right side collection of available widgets, as shown in Figure 5.22.

In what concerns the control panel, both structurally and conceptually remained the same as in the first prototype. The control panel is composed of a search section, which includes a search box that at the present moment contrary to the plan only allows to search for available widgets, a widget section, which allows to toggle on and off widgets, a configuration section, which allows to choose different data aggregation scopes and methods, and a support section, which includes a reset button, to return the application to its initial state, and a help button, to present a popup with a help guide for the application. Therefore, the sole difference between the prototypes consists in the fact that the support section includes, as part of its help guide introduction, the functionality of the previously sketched individual about button, meaning the help guide presents information regarding the development context of the application in its introduction.

Furthermore, regarding the support section, since it could not be found in the majority of the works
described in Section 3.4, it constituted an important feature to include in our application as an attempt to improve the overall application usability and, consequently, provide a better user experience according to the Nielsen usability heuristics “User control and freedom” and “Provide help and documentation” [Nielsen, 1994]. Regarding its composing reset button, as mentioned before when clicked it allows the application to return to its initial state, however in more detail, this returning procedure includes the creation of all the available widgets that were closed during the user interaction, as well as, the adjustment of the global and specific configurations of each widget to its default values. Regarding its composing help button, as mentioned before when clicked it allows the user to read an application help guide documentation, however in more detail, this help guide documentation consists in a brief, precise and detailed guide on the purpose and usage of the application along four chapters including an intro chapter, with an introduction on the development context of the application, a configurations chapter, with information on how to download a sample of the OpenSubtitles Corpus and create a starter configuration file, a control panel chapter, with information on all the different functionalities available in each of the control panel sections, and, finally, a widgets chapter, with information on extra functionalities available in each widget of the dashboard. This help guide documentation is presented in following image.

![Image](image-url)

**Figure 5.23:** B-subtle App help guide documentation.

In what concerns the widget collection, widgets have remained grouped according to the subject matter of their metrics, meaning into the same four major views: performance providing answer to Task 1 and 2; words providing answer to Tasks 3, 4 and 5; topics providing answer to Tasks 3 and 4; and sentiments providing answer to Task 6 and 7. Despite the similarity in the grouping of the widgets, the main difference between the first and the present prototype lies in the widgets actual content.
which includes not only the idiom, used to encode its corresponding data metrics, but also the specific configurations for that idiom, used to aggregate the data in that specific idiom making use of various of its underlying data attributes. Next we will describe in detail the idioms and specific configurations composing each of the available widgets in our visual dashboard.

![Figure 5.24: Execution Time widget.](image1)

![Figure 5.25: Dialogue Turns widget.](image2)

The Execution Time and Dialogue Turns widgets, presented in Figure 5.24 and Figure 5.25, respectively, both fall under the performance view and are the sole widgets that do not include an idiom per se, meaning do not include a visual encoding of the corresponding metric with a chart, but rather a table. In addition, this widgets also do not have any specific configurations given the nature of their metric. They differ from the first prototype in their increased simplicity with a more effective and clear presentation of the information as means to increase the signal-to-noise ratio associated with good data visualizations. This was achieved through the removal of the additional confusing textual organization and color brightness variations in the first sketches presented in Figure 4.2a and Figure 4.2b.

Regarding the Execution Time widget, as the name implies, it concerns the execution time of the framework preprocessing pipeline for a given starter subtitle corpora. It includes the total execution time, as well as the average and maximum execution time per subtitle file in the starter subtitle corpora. The metrics are all presented in the same seconds unit.

Regarding the Dialogue Turns widget, as the name implies, it concerns the quantity of dialogue turns that the framework filtered and included in the final produced corpora. It includes the total number of filtered dialogue turns, as well as the average and maximum number of filtered dialogue turns per each file of the starter subtitle corpora. The metrics are all presented in the same dialogue turns unit.
The Input Files, Filtered Files and Output Files widgets presented in Figure 5.26, Figure 5.28 and Figure 5.27 respectively, all fall also under the performance view and are all composed of both an idiom and specific configurations.

Regarding the Input Files widget, it is composed of a donut chart idiom which encodes the volume of the files composing the starter subtitle corpora emphasizing the validity and invalidity of such files according to their conformance with the OpenSubtitles Corpus XML standard schema. In more detail, in this idiom the mark is each donut slice and as for the used channels: since validity and invalidity concern the same type of file however following more or less a file standard schema, we have considered them as extreme values of a file validation scale, constituting an attribute with sequential nature and, therefore, are encoded through the color saturation channel, at the same time the file volume also constitutes a sequential attribute and, therefore, is encoded through the length channel. Furthermore, this widget specific configuration allows to change the unit of the corresponding volume metric between number of files and corresponding size in Megabytes (MB).

Regarding the Filtered Files widget, it is composed of a donut chart idiom which encodes the number of files accepted per each type and instance of filter composing the preprocessing pipeline defined by the user. In more detail, in this idiom the mark is each donut slice and as for the used channels: the different types and instances of filters constitute a categorical attribute and, therefore, are encoded through the color hue channel, while the file number constitutes a ratio attribute and, therefore, is encoded through the length channel. Furthermore, this widget specific configuration allows to choose between analyzing the number of files per each type of filter, including Metadata and Interaction filter types, or alternatively per each filter instance, including all the specific filters available under each of the previous filter types – for instance, Genre and Duration metadata filters as well as Interaction Interval and Frequency interaction filters.

Regarding the Output Files widget, it is composed of a donut chart idiom which encodes the volume of the files composing the final produced corpora differentiating between the four different produced file formats supported by the framework including JSON, Legacy, Parallel and XML. In more detail, in this
idiom the mark is each donut slice and as for the used channels: the different supported formatting of the produced files constitute a categorical attribute and, therefore, are encoded through the color hue channel, while the volume of each file format constitutes a ratio attribute and, therefore, is encoded through the length channel. Furthermore, this widget specific configuration allows to change the unit of the corresponding volume metric between number of files and corresponding size in MB.

The Word Cloud and Word Count widgets presented in Figure 5.29 and Figure 5.30 respectively, all fall under the words view and are all composed by an idiom, however only the Word Cloud additionally includes specific configurations.

Regarding the Word Cloud widget, it is composed of a bubble chart idiom which encodes the frequency of occurrence of 100 words present in the subtitle dialogue turns composing the final produced
corpora and which is presented in Figure 5.29b. In more detail, in this idiom there are two marks being used simultaneously to encode the same data instance consisting in each term and bubble and, additionally, the channel used to encode the frequency of occurrence of each word, since this constitutes a ratio attribute, is the radius channel. Furthermore, this widget specific configuration allows to re-encode the idiom by choosing between all types of words, which resulted in the encoding already described, or alternatively the words corresponding to named entities including Persons, Locations and Organizations, which results in a different encoding including a packed bubble chart idiom. This different idiom results from the addition of the new categorical attribute corresponding to the type of named entity, which is additionally encoded through a color hue channel, resulting in the new encoding presented in Figure 5.29c.

Regarding the Word Count widget, at the moment this widget does not include any specific configurations, however its corresponding idiom depends on the global configurations genre and year active at the moment. By default both those global configurations are active, which results in a grouped bar chart idiom, encoding the number of words present in the subtitle dialogue turns composing the final produced corpora per movie genre and per production year. In more detail, in this idiom the bar length channel encodes the ratio attribute corresponding to the number of words, the x-axis position channel encodes the categorical attribute corresponding to the movie genre and, finally, as for the continuous attribute corresponding to the production year it is encoded with the x-axis position within each group and the color saturation redundant channels. This choice of channels for the movie genre and the production year are justified in more detail in Section 5.3.1. However, when only one of those global configurations is active, the previous idiom is re-encoded into a simple bar chart, in which the sole difference lies in the channels used to encode the movie genre or production year attributes consisting just on the x-axis position channel. This new encodings are presented in Figure 5.30b and Figure 5.30b.

![Figure 5.31: Topics.](image)

![Figure 5.32: Filtered Files.](image)

The Topics widget presented in Figure 5.31, at the moment is the only widget that falls under the
topics view and is composed of a tag cloud idiom, which encodes the frequency of occurrence of the plot-keywords from the movies present in the final produced corpora, extracted from the TMDb as described in Section 5.2.4. In more detail, in this idiom the mark is each term and the channel used to encode the frequency of occurrence of each plot-keywords, which constitutes a ratio attribute, is the font-size. At the moment this widget does not include specific configurations.

The Sentiments widget presented in Figure 5.32, just as the Word Count widget, at the moment does not include any specific configurations, however its corresponding idiom depends on the global configurations genre and year active at the moment. By default both those global configurations are active, which results in a bar chart idiom, encoding the total number of dialogue turns present in the final produced corpora classified under each of the five sentiments, including very negative, negative, neutral, positive and very positive. In more detail, in this idiom the bar length channel encodes the ratio attribute corresponding to the number of dialogue turns of each sentiment and, as for the ordered attribute corresponding to the sentiments, it is encoded with the x-axis position and the color hue redundant channels. However, when only one of those global configurations is active, the previous idiom is re-encoded into a grouped bar chart, in which the sole difference lies in the addition of either the movie genre or the production year attribute which is encoded with the x-axis position channel. In this case, the previous sentiments attribute x-axis position channel changes to x-axis position within each group. This new encodings are presented in Figure 5.32b and Figure 5.32c.

Moreover, still concerning the widget collection however changing the perspective from its representation to its interactivity, besides the re-encoding and re-configuration possible through the usage of the
global and specific configurations available in the control panel, in each widget there are extra functionalities that were implemented and allow the user to further interact with the corresponding idioms.

One such interaction is the user *mouse hover* over a given data instance in the idiom, which will render a *tooltip* with detailed information on the attribute-value pairs of the tracked data instance, as shown in Figure 5.33.

Another interaction is the user *mouse click* on a given *legend* in the idiom, which results in filtering out all the data instances falling under such attribute-value pair group corresponding to the tracked legend, as shown in Figure 5.35.

And, finally, some other extra widget functionalities shown in Figure 5.34 include the possibility to resize the idiom to *fullscreen mode*, with the purpose of avoiding any possible data occlusion due to screen size limitations, as well as the possibility to *print* and *download* idioms either in *PNG*, *JPEG*, *PDF* or *SVG* file formats, with the purpose of helping the user persist its explorations and discoveries made with the analytics dashboard for reporting reasons.
6 Evaluation

Contents

6.1 Usability .............................................................. 72
6.2 Utility ................................................................. 74
6.3 Discussion ........................................................... 76
In the previous chapters, we have described in detail the extensions we performed to the original version of B-Subtle framework through both the addition of new pipeline components and the creation of an entire visual analytic tool in the form of a desktop application. In this chapter we focus in the evaluation of that visual analytic tool. Such evaluation will encompass both usability and utility testing. For each of those tests, we defined a different set of questions to be answered by the users with the help of our tool. Those sets of questions attempted to cover all the tasks described in Section 4.2.2 and included simple questions involving a single task and more complex questions involving more than one task. We have conducted those tests with both NLP researchers from the HLT community\(^\text{18}\), students from the IST university\(^\text{19}\) and personal acquaintances, all with high interest in cinematography, as well as familiarized and comfortable with the use of technologies. Furthermore, we mainly deployed observation and survey data collection methods for the tests with our users. The former allowed us to note the difficulties they encountered while using our tool and the latter allowed us to understand their overall satisfaction. Our ultimate goal is to use state-of-the-art-techniques in the field of Information Visualization to develop a more user-friendly interface for B-Subtle framework that might be useful in the creation of personalized dialogue corpora for the accomplishment a wide range of research tasks. This requires to offer both a good usability and utility for our target users Munzner \([2014]\).

### 6.1 Usability

Performing usability tests enabled us to ascertain how effectively and efficiently the users can accomplish tasks with the help of our tool and their overall satisfaction. Even though our objective was to follow an iterative design process with multiple iterations throughout the system development phase, formative evaluations, and a final iteration concluding the system development phase, summative evaluation, due to the time variable we were only able to perform a single iteration consisting in the summative evaluation. We performed this evaluation with 15 users, a number strictly chosen to fall within the range of 10 to 20 users as suggested in Fonseca et al. \([2012]\), however opposing to the recommended in-person evaluation due to the present global pandemic, the evaluation was performed through social communication platforms such as Skype and Google Hangouts over mainly voice calls and, in some cases, also video chats. The evaluation was composed of a first section where the users were asked to provide answer to four questions presented in Table 6.1 using our visual analytic tool, followed by a section where the users were asked to help us quantify the usability of our tool providing answer to the classic 10 System Usability Scale (SUS) questions \([Brooke, 1996]\). It is important to note that for the first section of the evaluation each question had a different complexity, which progressively increased from lowest in question Q1 to highest in question Q4 also presented in Table 6.1.
In the first section of the evaluation, we have adopted a quantitative testing approach by measuring the time required to answer each question, referenced in variable **Time**, as well as the number of actions that did not contribute to answer the question, referenced in variable **Errors**. In Table 6.2 we present the means and standard deviations for both time and errors variables measured during the users usability evaluations. From those statistics we can notice that despite having considered that question Q2 had a low complexity L2, it in fact proved to be the question with highest complexity given its higher mean for both time and errors in comparison with the remaining questions.

<table>
<thead>
<tr>
<th>ID</th>
<th>Question</th>
<th>Solution</th>
<th>Complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Q1</td>
<td>What is the second most common movie topic(s)?</td>
<td>Find widget <strong>Topic</strong> and find it is “based on novel or book”, “magic”, “dystopia” and “future”</td>
<td>Lowest (L1)</td>
</tr>
<tr>
<td>Q2</td>
<td>What is the most popular character name?</td>
<td>Find widget <strong>Word Cloud</strong>, check specific configuration <strong>Named Entities</strong> and find it is “Chip”</td>
<td>Low (L2)</td>
</tr>
<tr>
<td>Q3</td>
<td>Which movie genre is the most talkative?</td>
<td>Find widget <strong>Word Count</strong>, remove global configuration <strong>Year</strong> and find it is “Action”</td>
<td>High (L3)</td>
</tr>
<tr>
<td>Q4</td>
<td>What is the most predominant sentiment(s) in 2017?</td>
<td>Find widget <strong>Sentiments</strong>, add and remove global configuration <strong>Year</strong> and <strong>Genre</strong> respectively and find it is “Positive” and “Very Positive”</td>
<td>Highest (L4)</td>
</tr>
</tbody>
</table>

Table 6.1: Application usability evaluation questions definition and corresponding solutions and complexity.

<table>
<thead>
<tr>
<th>ID</th>
<th>Time</th>
<th>Errors</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Mean</td>
<td>Std. Deviation</td>
</tr>
<tr>
<td>Q1</td>
<td>48.13</td>
<td>53.116</td>
</tr>
<tr>
<td>Q2</td>
<td>133.20</td>
<td>43.080</td>
</tr>
<tr>
<td>Q3</td>
<td>67.13</td>
<td>38.706</td>
</tr>
<tr>
<td>Q4</td>
<td>27.67</td>
<td>15.783</td>
</tr>
</tbody>
</table>

Table 6.2: Application evaluation questions definition and descriptive statistics.

Still regarding the first section of the evaluation and the corresponding quantitative testing, provided that as mentioned before each question had a different level of complexity, it is also interesting to further understand if the different question complexities impacted the variables time and errors. To derive conclusions on that impact we conducted various statistical tests starting with a normality test known as Shapiro-Wilk, to decide if further analysis of each variable should be done with a parametric or non-parametric test depending on its significance against a normal distribution. Starting from a null-hypothesis that both variables were normally distributed with a chosen $\alpha = 0.05$, as shown in Table 6.3 through the application of the Shapiro-Wilk test we concluded that for both variables there was evidence in at least one of the questions, from Q1 to Q4, of the null-hypothesis rejection with significance values less than $\alpha = 0.05$, meaning both variables were not normally distributed and, therefore, both needed to be further analysed with a non-parametric test know as Kruskal-Wallis. In the application of the Kruskal-Wallis test we defined a null-hypothesis that for both variables there was no difference in their mean between question complexities and further defined the degree of freedom to be the number of different complexities minus 1 $df = 3$ and a chosen $\alpha = 0.05$ which lead to a critical value $Q = 7.81473$. As
shown in Table 6.3 through the application of the Kruskal-Wallis test we concluded that for both variables there was evidence of the null-hypothesis rejection with statistic values $H$ greater than $Q = 7.81473$, meaning there was a difference in the means of both variables depending on the question complexity and, therefore, the complexity impacted the users time and errors in each question.

<table>
<thead>
<tr>
<th>Variable</th>
<th>Question</th>
<th>Shapiro-Wilk</th>
<th>Kruskal-Wallis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Q1</td>
<td>0.000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Q2</td>
<td>0.061</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Q3</td>
<td>0.458</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Q4</td>
<td>0.009</td>
<td></td>
<td>28.843</td>
</tr>
<tr>
<td>Errors</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Q1</td>
<td>0.000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Q2</td>
<td>0.015</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Q3</td>
<td>0.000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Q4</td>
<td>0.000</td>
<td></td>
<td>21.411</td>
</tr>
</tbody>
</table>

Table 6.3: Application evaluation on question complexity statistics.

Regarding the next section of the usability evaluation where we asked users to answer the classic 10 questions of the SUS, we used the method defined in Brooke [1996] to measure the usability of our tool for each individual questionnaire on a scale from 0 to 100. Afterwards, to determine the overall usability of our tool we calculated the average of the usability scores per questionnaire and achieved a final usability of 85.83, which given its greater then 80.3 means our tool achieved an excellent usability. The Table 6.4 summarizes the calculations performed to achieve the final usability of 85.83.

<table>
<thead>
<tr>
<th>Users</th>
<th>Questions</th>
<th>Individual Score</th>
<th>Total Score</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1  2  3  4  5  6  7  8  9  10</td>
<td></td>
<td>85.83</td>
</tr>
<tr>
<td>U1</td>
<td>3(2) 1(4) 5(4) 1(4) 4(3) 2(3) 5(4) 1(4) 4(3) 1(4)</td>
<td>87.5</td>
<td></td>
</tr>
<tr>
<td>U2</td>
<td>5(4) 1(4) 5(4) 1(4) 5(4) 1(4) 5(4) 1(4) 4(3) 1(4)</td>
<td>97.5</td>
<td></td>
</tr>
<tr>
<td>U3</td>
<td>5(4) 2(3) 4(3) 2(3) 5(4) 1(4) 4(3) 2(3) 4(3) 1(4)</td>
<td>85.0</td>
<td></td>
</tr>
<tr>
<td>U4</td>
<td>3(2) 1(4) 4(3) 1(4) 4(3) 1(4) 5(4) 1(4) 4(3) 2(3)</td>
<td>85.0</td>
<td></td>
</tr>
<tr>
<td>U5</td>
<td>5(4) 2(3) 4(3) 2(3) 4(3) 1(4) 4(3) 1(4) 4(3) 1(4)</td>
<td>85.0</td>
<td></td>
</tr>
<tr>
<td>U6</td>
<td>3(2) 2(3) 3(2) 2(3) 3(2) 2(3) 3(2) 2(3) 3(2) 1(3)</td>
<td>62.5</td>
<td></td>
</tr>
<tr>
<td>U7</td>
<td>5(4) 1(4) 5(4) 2(3) 5(4) 1(4) 4(3) 1(4) 3(2) 1(4)</td>
<td>90.0</td>
<td></td>
</tr>
<tr>
<td>U8</td>
<td>3(2) 2(3) 4(3) 2(3) 5(4) 1(4) 5(4) 1(4) 4(3) 2(3)</td>
<td>82.5</td>
<td></td>
</tr>
<tr>
<td>U9</td>
<td>3(2) 2(3) 5(4) 1(4) 5(4) 1(4) 4(3) 1(4) 5(4) 1(4)</td>
<td>90.0</td>
<td></td>
</tr>
<tr>
<td>U10</td>
<td>5(4) 1(4) 5(4) 1(4) 4(3) 1(4) 5(4) 1(4) 5(4) 1(4)</td>
<td>97.5</td>
<td></td>
</tr>
<tr>
<td>U11</td>
<td>4(3) 1(4) 5(4) 1(4) 5(4) 1(4) 5(4) 1(4) 5(4) 1(4)</td>
<td>97.5</td>
<td></td>
</tr>
<tr>
<td>U12</td>
<td>3(2) 2(3) 4(3) 2(3) 3(2) 1(4) 4(3) 2(3) 4(3) 1(4)</td>
<td>75.0</td>
<td></td>
</tr>
<tr>
<td>U13</td>
<td>3(2) 1(4) 3(2) 3(2) 4(3) 2(3) 3(2) 1(4) 4(3) 1(4)</td>
<td>72.5</td>
<td></td>
</tr>
<tr>
<td>U14</td>
<td>4(3) 1(4) 4(3) 2(3) 4(3) 1(4) 4(3) 1(4) 3(2) 1(4)</td>
<td>82.5</td>
<td></td>
</tr>
<tr>
<td>U15</td>
<td>5(4) 1(4) 5(4) 1(4) 5(4) 1(4) 5(4) 1(4) 4(3) 1(4)</td>
<td>97.5</td>
<td></td>
</tr>
</tbody>
</table>

Table 6.4: Application evaluation SUS. For each question we present the answer in a scale of 1 (Strongly Disagree) to 5 (Strongly Agree) followed by the score contribution in brackets. The Individual Score column represents the usability score for each questionnaire.

### 6.2 Utility

Complementary to usability, performing utility tests will enable us to ascertain if our tool offers all the features required by our target users and possibly more [Munzner, 2014]. These tests were performed...
at the end of the tool development phase and involved a smaller number of 2 users per each group of researchers and cinematographic enthusiasts. These users had also participated in the previous usability evaluation, provided it was required that they had some context knowledge, meaning were acquainted with our tool and already had some understanding of the concepts involved. During those tests we provided a list of four questions the users had to answer with the help of our tool, however we did not measure the time they spent or the mistakes they did while accomplishing such tasks, but rather we considered subjective metrics by adopting a thinking aloud testing approach with no time restrictions and live qualitative comments. As a result we gathered a rich collection of not only user constructive critiques to implemented features, but also our examiner observations on usage patterns, which will both be described next.

In the context of researchers which constitutes the primary target group of our application, one of the most predominant critique was the need to provide high-level description on the information being visualized in each widget. In detail, users were not clear on the precise definition of the concept “filtered file” if it was referring to files accepted or reject by the framework, if the Execution Time and Dialogue Turn widgets referred to the starter subtitle corpora or the final produced corpora, as well as what was the underlying concept of the Input and Output Files widgets. In addition to those critiques we also gathered an interesting observation on usage patterns which occurred with both users and consisted in them removing all the widgets from the dashboard that did not seem related with the problem being solved prior to any further problem-solving step.

In the context of cinematographic enthusiasts which constitutes the secondary target group of our application, users predominant critiques concerned the design inconsistency between the global and specific configurations which might lead the user to misleadingly perceive them as different forms of control, the difficulty in noticing the border highlighted in selected widgets, the hidden position of the reset button on some screen sizes, as well as the difficulty in comparing word frequencies solely through bubble radius in the Word Cloud widget. Additionally, we also gathered observations on usage patterns and found that most users were not aware of the existence of interactive legends for some widgets that could be toggled on/off to reconfigure the visualization and present solely the data they intended, in most cases the users overlooked the existence of legends.

Some observations commonly found on both user targeted groups were the frequent misunderstanding between the widget title and subtitle, which in most cases lead the users to think a widget referred to a different corpora analysis aspect, as well as the highly frequent case in the beginning of the test when users used the widget section available in the control panel to toggle on/off widgets from the dashboard, but due to screen size limitations it was not perceptible to them that widgets were being added and removed since these were outside the viewport and to be visible required scrolling, which also in such case proved not to be an intuitive action for most users. The latter leads to another highly frequent
critique regarding the additional difficulty in understanding when a given widget was selected and that upon selection there existed a cause-effect relationship between the selected widget and the content of the specific configuration section available in the control panel. The reason for such difficulty supported by users opinions, was due to not being intuitive the decentralized placement of the specific configurations for a given widget outside of the corresponding widget, as well as the screen size limitations and current visible content in the viewport.

6.3 Discussion

As mentioned before, our visual analytic tool targets two user groups including researchers and cinematographic enthusiasts. The former would use our tool to perform research tasks in a work environment, while the later would use our tool to explore personal interests mostly during leisure time. This work evaluation proved such difference in usage, since each user target group showed slightly different concerns during their interaction with our tool. Researchers mainly focused on features that could provide a precise description on the information being displayed on each widget, reflecting their concern not only on the value the application could provide in performing their research tasks, but also the need to have a deeper understanding of the application with easy to access memos in order to speed the learning curve and quickly formulate a mental model on the usage of the tool. In opposition, cinematographic enthusiasts mainly focused their attention on visual aspects of the visualization with frequent comments on design choices.

Moreover, our usability evaluation results determined our tool achieved a good usability score and there was a strong correlation between the task complexity and the time required to perform such task, as well as the possible number of errors committed when performing it. However, in an attempt to further correlate the results of both usability and utility evaluations we can justify such correlation due to few inconsistencies and information gaps found in our tool by users throughout the tests. For instance multiple users claimed to had difficulty in finding the location of the widget specific configuration mainly due to its placement farther away from the corresponding widget, proving to be the cause for the increase in mean time and error rate for question Q2 in Table 6.2. In addition, we can also argue that such correlation might have been affected by the order in which questions were performed. For instance questions Q3 and Q4 both required the use of global configurations as explained in table 6.1, however question Q3 marked the first user interaction with such feature and was immediately followed by question Q4 where users easily recalled that feature placement and behaviour from shot-term memory, contributing to the lowest mean of time when compared with the remaining questions.

Among others, this work evaluation allowed us to understand the aspects we should focus in the future to improve even further the usability and utility of our tool and the need to further evaluate such
improvements with an even larger group of users to improve the accuracy in the analysis on correlation between question complexity and both time and errors.
Conclusion

Contents

7.1 Conclusions ................................................................. 79
7.2 System Limitations and Future Work ................................. 80
7.1 Conclusions

B-Subtle is an open source framework developed in the context of a previous dissertation work, that allows researchers to automatically personalize corpora containing dialogue turns extracted from subtitles belonging to the OpenSubtitles Corpus. However, its original version needed further extension of its preprocessing steps catalogue and implementation of an alternative to its sole console-based interface. Those combined determined the following fundamental goals and solutions of our present work.

The ultimate goal of our work is a framework that attempts to assist researchers working with subtitles corpora in the creation of personalized dialogue corpora specific to fit their research requirements. In more detail, the framework attempts to help researchers to avoid having to repeatedly implement the same preprocessing steps and to reduce the corpora to a more manageable size, while still being able to produce corpora tailored to their requirements. For that reason, we have extended the B-Subtle framework with additional preprocessing steps which offer the possibility to reduce the volume of the starter subtitle corpora through the removal of subtitle duplicates, to filter dialogue turns based on the frequency of occurrence of their composing trigger, answer or both, to filter both subtitles and dialogue turns based on the presence of specific topics, as well as to further extract conversations from the sequence of dialogue turns.

The secondary goal of our work is not only to further attempt to assist researchers in the incremental process of discovering the sequence of preprocessing steps that results in the ideal dialogue corpora according to their research tasks, but also extend the usage of B-Subtle to a broader audience of cinematographic enthusiast apart from researchers. In more detail, we intend to further help researchers in their iterative evaluation of the produced corpora by easing the analysis process with visualizations of the statistical information gathered during the production of such dialogue corpora, while still keeping the simplicity for users not as familiar with technical concepts in the field of NLP. For that reason, we have developed from scratch B-Subtle App, a cross-platform desktop application powered by B-Subtle Framework and which is responsible for rendering a visual dashboard that aims to provide answer to various questions including metrics on the framework performance, as well as on words, topics and sentiments frequency of occurrence, among others.

To determine if B-Subtle App offers in an effective and efficient manner all the features our target users require for the accomplishment of their tasks, we conducted a usability and utility evaluations which allowed us to conclude that our application achieved a good usability score of 85.83, as well as the existence of a correlation between tasks complexity and both time spent and errors committed in performing such task. Moreover, allowed us to gather user constructive critiques of implemented features and usage patterns. These offered us insights on the concerns of each user group, with researchers mainly prioritizing effectiveness and efficiency while cinematographic enthusiasts mainly prioritizing user experience, as well as on possible causes for the unexpected higher/lower complexity.
7.2 System Limitations and Future Work

The main limitation of the present version of the B-Subtle framework derives from its initial development with the Java programming language, which despite having in its favor some advantages such as portability allowing for a seamless user experience in a diversity of platforms, in fact given the framework field of application it lacks on a greater support needed for the processing and manipulation of natural language data with recourse to lexical and semantic resources available online, that can be found in NLP-friendly programming languages such as Python with default integrations like NLTK. For that reason, in future work prior to possible further extensions of the framework catalogue of preprocessing steps with addition of components considering for instance semantic similarity between utterances and existence of unwanted subber annotations, we propose to change the framework underlying programming language preferably to Python.

Another B-Subtle framework limitation consists in its sole support to produce dialogue corpora from subtitle corpora. Despite the fact that subtitles have in its favor the advantages of high online availability and a wide variety of multilingual translations, however when compared with other form of unnatural corpora such as scripts, it lacks additional attributes that could be of interest for researchers to include in their preprocessing pipelines such as the precise identification of scenes, as well as the existence of a mapping between utterances and speakers. The former would allow to greatly improve the newly implemented Conversation Filter avoiding the currently possible extraction of false-positives conversations that span two distinct scenes. The latter would allow to extend the catalogue of preprocessing steps with a new dimension of speakers information. For that reason, in future work we also propose the implementation of a new type of pipeline specific for preprocessing script corpora.

Regarding B-Subtle App, during its evaluation with users we have gathered a rich collection of user constructive critiques and usage patterns, which showcased us the existence of few inconsistencies and information gaps in our application that once solved could greatly increase its value. For that reason, in future work we propose: to provide within each widget an help option that when moused hover would render a tooltip with an high-level description of the information being visualized for each widget; to rethink the control panel design to occupy lesser viewport area and avoid the need to scroll, in an attempt to solve preattentive processing of cause-effect relationship between the control panel and widget dashboard; to reposition the specific configurations inside the affected widget, in an attempt to adapt the application to our users mental model of centralizing interactions specific to a widget within that widget; and, finally, to complement the Word Cloud bubble chart with a redundant bar chart, in an attempt to ease comparison between word frequencies through both radius and length. These constitute some of the proposed improvements we consider have higher priority due to priority of affected group of users and corresponding issue coverage.
Bibliography


M. Angelo. Dissertation: One Million Agents Speaking All the Languages in the World. (May), 2018.


A. Bazo, M. Burghardt, and C. Wolff. TWORPUS - An easy-to-use tool for the creation of tailored twitter corpora. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 8105 LNAI:23–34, 2013. ISSN 03029743. doi: 10.1007/978-3-642-40722-2\{\_\}_3.


