FAKULTÄT
FÜR !NFORMATIK

Faculty of Informatics

Ein visuelles Erkundungstool zur zeitlichen Analyse von Kundenbewertungen

DIPLOMARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieur

im Rahmen des Studiums

Wirtschaftsinformatik

eingereicht von

Blagoy Panayotov
Matrikelnummer 01643722

an der Fakultät für Informatik

der Technischen Universität Wien

Betreuung: Ao.Univ.Prof. Dipl.-Ing. Dr.techn. Eduard Gröller
Mitwirkung: Univ.Ass. Dr.techn. Manuela Waldner

Wien, 4. Mai 2020

Blagoy Panayotov

Eduard Gröller

FAKULTÄT
FÜR !NFORMATIK
Faculty of Informatics

A Visual Exploration Tool for Temporal Analysis of Customer Reviews

DIPLOMA THESIS

submitted in partial fulfillment of the requirements for the degree of

Diplom-Ingenieur

in

Business Informatics

by

Blagoy Panayotov
Registration Number 01643722

to the Faculty of Informatics

at the TU Wien

Advisor: Ao.Univ.Prof. Dipl.-Ing. Dr.techn. Eduard Gröller
Assistance: Univ.Ass. Dr.techn. Manuela Waldner

Vienna, 4th May, 2020

Blagoy Panayotov   Eduard Gröller

Erklärung zur Verfassung der Arbeit

Blagoy Panayotov

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

Wien, 4. Mai 2020

Blagoy Panayotov

Kurzfassung

Diese Arbeit untersucht textliche Bewertungen und ihre Änderung mit der Zeit. Die Diplomarbeit ist durch die ständig erstellten textuelle Bewertungen motiviert. Bewertungswebseiten wie Yelp und TripAdvisor generieren monatlich Hunderttausende von Bewertungen. Es ist nicht möglich, diese Menge an Daten zu analysieren indem man jede einzelne Bewertung einfach liest. Wir suchen nach Mitteln, um die Fragen zu beantworten, die von Business-Analysten, Unternehmern und Investoren zu den Daten der Kundenbewertungen gestellt werden. Diese Diplomarbeit beschäftigt sich mit den Fragen wie: Warum verändern sich die Bewertungsnoten und –inhalte im Laufe der Zeit? Was sind die wichtigsten Themen, die diskutiert werden? Was sind die typischen Gründe, weswegen die Bewertungsnoten plötzlich steigen oder fallen? Was sind die Themen, die permanente oder vorübergehende Veränderungen in einer großen Sammlung von Bewertungsnoten hervorrufen?

Wir haben ein Werkzeug namens "Review Watcher" entwickelt, das neuartige Herangehensweisen zur Untersuchung und Analyse von Bewertungsänderungen im Laufe der Zeit bietet. Das Werkzeug hat als Ziel, einfache, leicht zugängliche Informationen über zeitliche Veränderungen in einer Sammlung von Restaurantbewertungen zu liefern. Das Werkzeug verwendet reale Daten, die von Yelp zur Verfügung gestellt werden. Es verwendet grafische Methoden, um Veränderungen in den Bewertungsergebnissen über verschiedene Zeiträume anzuzeigen. Das Werkzeug analysiert die Bewertungsnoten im Laufe der Zeit, und versucht, Änderungen in diesen Bewertungen auf der Basis des Textinhaltes der Bewertungen zu erklären. Das Werkzeug verwendet automatische Textverarbeitungsalgorithmen, um wichtige und häufig verwendete Wörter in Textkorpora hervorzuheben.

Wir benutzten eine qualitative Auswertung, um festzustellen, wie gut das Werkzeug die Forschungsfragen beantwortet. Wir führten eine Benutzerstudie mit Experten im Bereich der Wirtschaft durch. Sie teilten ihre Erkenntnisse mit, die sie mit Review Watcher gesammelt hatten, und verglichen sie mit ihren Erfahrungen bei der Arbeit mit anderen Werkzeuge zur Kundenzufriedenheits- und Bewertungsanalyse.

Als Ergebnis unserer Forschung zeigen wir, dass Review Watcher es schafft, eine bessere Einsicht darin zu geben, was die Hauptthemen in einer Sammlung von textuelle Bewertungen sind. In der Diplomarbeit zeigen wir, dass Review Watcher im Vergleich zu bestehenden Werkzeuge für die Erkundung von Bewertungen besser geeignet ist, um die im Laufe der Zeit auftretenden Bewertungsänderungen hervorzuheben und Einblicke darin zu geben, warum die Änderungen aufgetreten sind. Das Werkzeug erweist sich auch als fähig, Millionen von textuelle Bewertungen von Zehntausenden von Restaurants mit für den Benutzer akzeptablen Ladezeiten zu verarbeiten. Die Benutzerstudie zeigt auch einige der Einschränkungen des Werkzeugs und sein Verbesserungspotential auf, z.B. bei der Einführung verbesserter Kategorisierungsfunktionen und geografischer Informationen über Restaurants.

Abstract

This thesis explores textual review data and how it changes over time. The thesis is motivated by the constantly generated textual reviews. Review sites like Yelp and TripAdvisor are generating hundreds of thousands of reviews monthly. Analysing this amount of data is impossible by simply reading every individual review. We look for ways to answer questions that business analysts, business owners, and investors ask about customer review data. This thesis asks questions such as: Why do review scores and topics change over time? What are the major topics people discuss? What are the typical reasons why review scores suddenly increase or decrease? What are topics that invoke permanent or transient changes in a large collection of review scores?

We created a tool called Review Watcher, which provides novel approaches to examine and analyse review changes over time. The tool aims to provide simple, easily accessible information regarding temporal changes in a collection of restaurant reviews. The tool uses real data provided by Yelp. It employs graphical ways to indicate changes in review scores over different periods of time. The tool analyses the review scores over time, and it tries to explain changes in these scores based on the textual content of the reviews. The tool utilises automated text processing algorithms to highlight important and often used words in text corpora.

We used a qualitative evaluation to determine how well the tool manages to answer the research questions. We completed a user study with experts in the field of economics. They shared the insights they gathered using Review Watcher and compared them to their experiences working with other tools for customer satisfaction and review analysis.

As a result of our research, we show that Review Watcher manages to provide better insight into what are major topics in a collection of textual reviews. In the thesis, we show that Review Watcher is better suited to highlighting review changes occurring over time and giving insights to why the changes occurred, compared to existing tools for review exploration. The tool is also proving capable of handling millions of textual reviews of tens of thousands of restaurants with acceptable loading times for the user. The user study also reveals some of the tool's limitations and potential for future work, for example in introducing improved categorisation functions and geographical information about restaurants.

Contents

CHAPTER 1

Introduction

In the current information age, content is constantly being created, expanded, evolved and overflows. News and opinions are now, more than ever, easily accessible and shareable by people. The technological advances enabled regular people to express themselves through social media applications in a way that can be heard from all over the world. Thus people were encouraged to create content themselves and share opinions among each other on a daily basis. However, this brings new information challenges. Exploring, analysing, and noticing patterns in the way information changes is a problem tackled by researchers from different fields [CC16]. And having an abundance of content and sources makes that even more challenging.

Understanding customers and their levels of satisfaction is another problem that business owners are trying to solve. Modern tools are providing users with the ability to rate products using numerical scores and/or comments in plain text form. A particularly interesting aspect is that scores and text reviews of products are changing and evolving over time. A product starts being generally perceived in a different manner by the majority of users. The reasons why a product suddenly starts generating a more positive or more negative response are not always clear to the owners, investors, and analysts. Therefore, business analysts are trying to find out why and when changes in the reviews occurred.

Putting everything into context, we will be looking at gastronomy related businesses and how they are perceived in platforms with user-generated opinions, such as TripAdvisor [Kau01] and Yelp [SS01]. Our focus is on restaurants because they are constantly changing businesses for which influencing customer's satisfaction is of great importance to their competitiveness. Restaurant owners, investors, general business analysts, and marketing researchers, are concerned with how customer opinions change and why. Therefore, we will be looking at how opinions evolve and change over time. We are attempting to get insight into *why* opinions changed. What prompts restaurants to get decreased or increased ratings? We are trying to gather more concise information from the millions of existing reviews, on which to base conclusions about the ever-changing market. Such information, once it is presented in an accessible way, can later be used to build predictions to aid future business strategies [Asg16].

Visualisation is an often considered method for dealing with information analysis issues. Since we are dealing with not only numeric scores but also long textual reviews, text processing functionalities are to be examined. Another issue is presenting the processed data. Noticing reoccurring terms and distinguishing prevailing words in a text collection or corpora has been done using 2D embedding [CTL18]. Another common way of showing and examining meaning in a text corpora is via word clouds. Such visualisation techniques have the potential to highlight words, which bear some importance at a specific point in time [CC16].

As already mentioned, finding collections of user reviews about different businesses is information made available to interested parties. However, if those reviews are presented in a single, conventional list view, reading them all and grasping the overarching themes could be challenging for a person when dealing with hundreds or thousands of entries. Following changes over time could also be troublesome for a person if the data is presented in such a manner. Connecting the textual reviews with the often given numerical score ratings is another challenge that needs to be tackled. While some tools for temporal data visualisation do already exist and show occurrences of changes over time, they only show the temporal development of the review scores, but they do not provide any means to explore how the text content changed over time [WHS+18]. Ratings becoming more dissimilar over time can be interactively explored and scaled [BSH+16]. However, what caused the increasing variance in the first place is not apparent, since the textual reviews are not explored. Some other tools tend to show content changes, but have issues with scaling large numbers of topics and time frames [GJG+15].

The thesis will focus on providing marketing researchers with a tool that gives answers to questions about the data. Such questions and concrete examples are the following:

• Why did review scores change at a particular time for a particular item? - For example, if there was a sudden decrease in a restaurant's overall score. Was it caused by a change of the restaurant's menu or was it some other specific reason?

• Are there items with a contrary trend over time? - What could be the reason that the acceptance of one item decreases, while the other does not? The inclusion of non-smoking areas might make a restaurant more favourable in a region with a lot of non-smokers and less so in a region with a large number of smokers.

• What are topics that are affecting all items over all time? - Which topics only affect some items or limited time periods? A food poisoning incident could be a transient topic. Does it have a strong influence on the numerical scores?

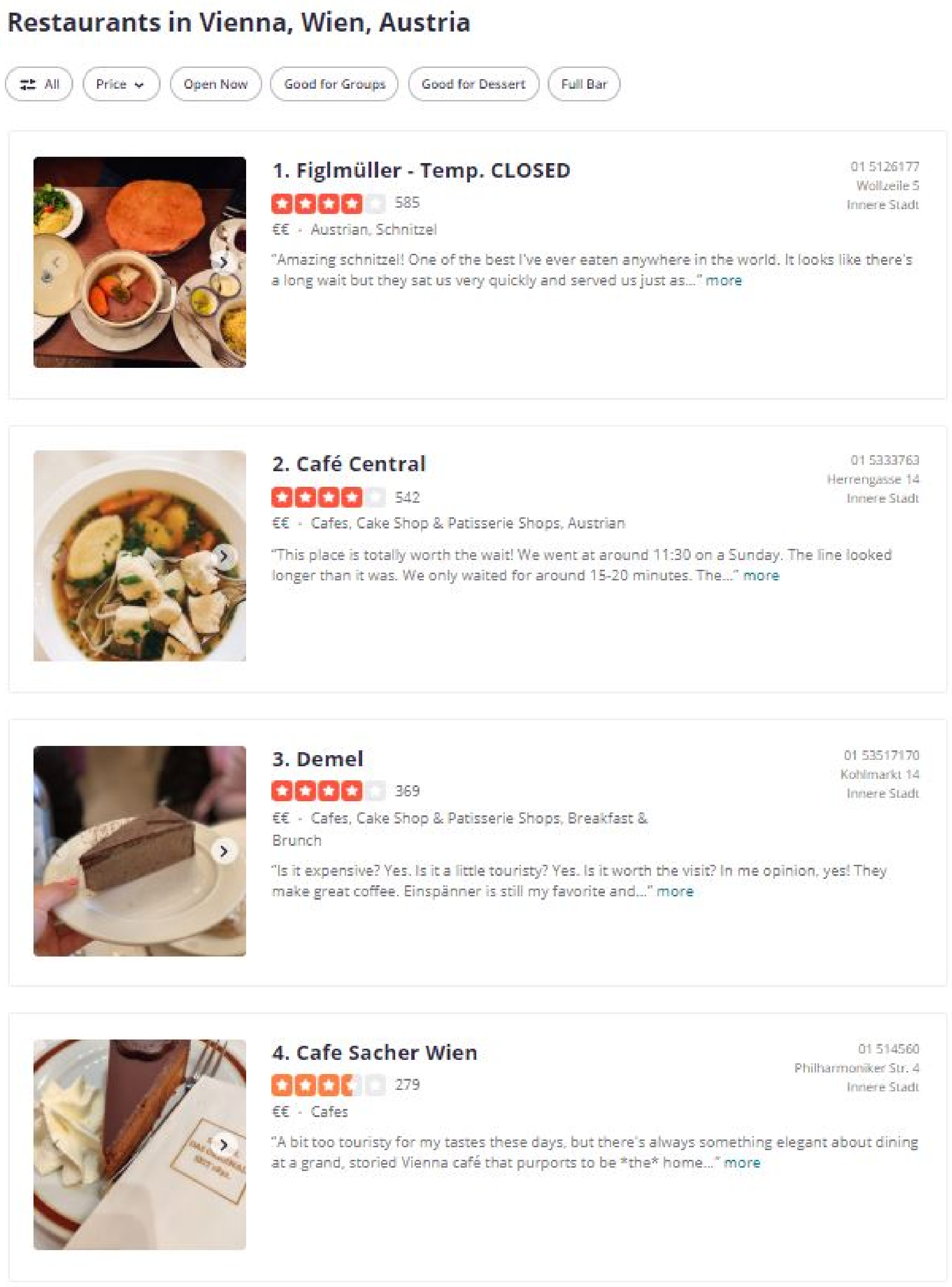
• What distinguishes an excellently scored item from a poorly scored item? - Is the quality of the food the main difference between a good and a bad restaurant?

In order to provide marketing researchers and business investors with answers to their questions above, the thesis will initially derive and answer the following technical questions:

- How to identify a change in the review scores?
- How to identify a change in the textual reviews?
- How to identify major topics?
- How to distinguish between permanent and transient topics?
- How to identify a sudden increase or decrease in review scores?
- How to identify topics of great importance to reviews?

Some assumptions that help us answer the technical questions, we aim to prove or disprove in this thesis, are:

- Visualisation of review scores for each restaurant is superior to a conventional ranked list for selecting restaurants to be analysed. – This assumption will be important for finding restaurants with diverse scores and also comparing restaurants to each other in order to filter. It will also provide a quick way to find an excellently scored restaurant, a poorly scored restaurant, a popular restaurant, or an unknown restaurant. We compare the conventional ranked list Yelp interface in Figure 1.1 to the newly developed ranking visualisation shown in Figure 1.2.



Restaurants in Vienna, Wien, Austria

1. Figlmüller - Temp. CLOSED
585
€€ · Austrian, Schnitzel
"Amazing schnitzel! One of the best I've ever eaten anywhere in the world. It looks like there's a long wait but they sat us very quickly and served us just as..." more
01 5126177
Wollzeile 5
Innere Stadt

2. Café Central
542
€€ · Cafes, Cake Shop & Patisserie Shops, Austrian
"This place is totally worth the wait! We went at around 11:30 on a Sunday. The line looked longer than it was. We only waited for around 15-20 minutes. The..." more
01 5333763
Herrengasse 14
Innere Stadt

3. Demel
369
€€ · Cafes, Cake Shop & Patisserie Shops, Breakfast & Brunch
"Is it expensive? Yes. Is it a little touristy? Yes. Is it worth the visit? In me opinion, yes! They make great coffee. Einspänner is still my favorite and..." more
01 53517170
Kohlmarkt 14
Innere Stadt

4. Cafe Sacher Wien
279
€€ · Cafes
"A bit too touristy for my tastes these days, but there's always something elegant about dining at a grand, storied Vienna café that purports to be *the* home..." more
01 514560
Philharmoniker Str. 4
Innere Stadt

Figure 1.1: The standard conventional ranked list Yelp interface. It is showing the four most reviewed restaurants in Vienna [SS01].

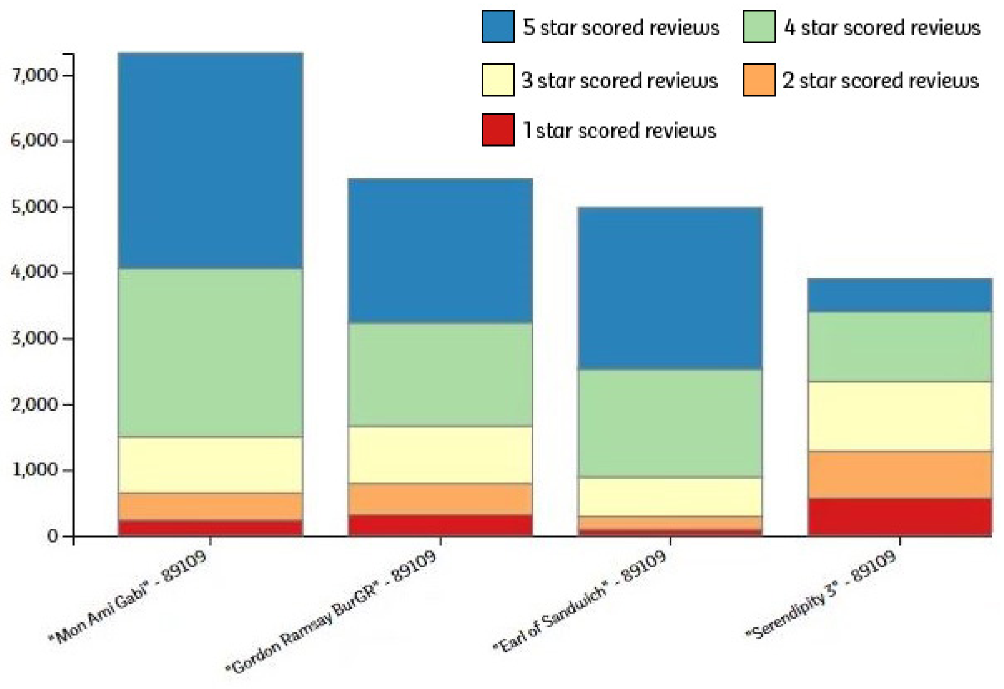


Figure 1.2: Visualisation of review scores for the four most reviewed restaurants taken from a random sample dataset. Along the x-axis are restaurant names. On the y-axis are the numbers of reviews.

• Visualising the temporal development of review scores helps to find time periods of interest. – This assumption helps the users to identify changes in review scores. It can also help them differentiate sudden score changes and notice a permanent or a transient trend in scores. An example for a temporal development view is shown in Figure 1.3.

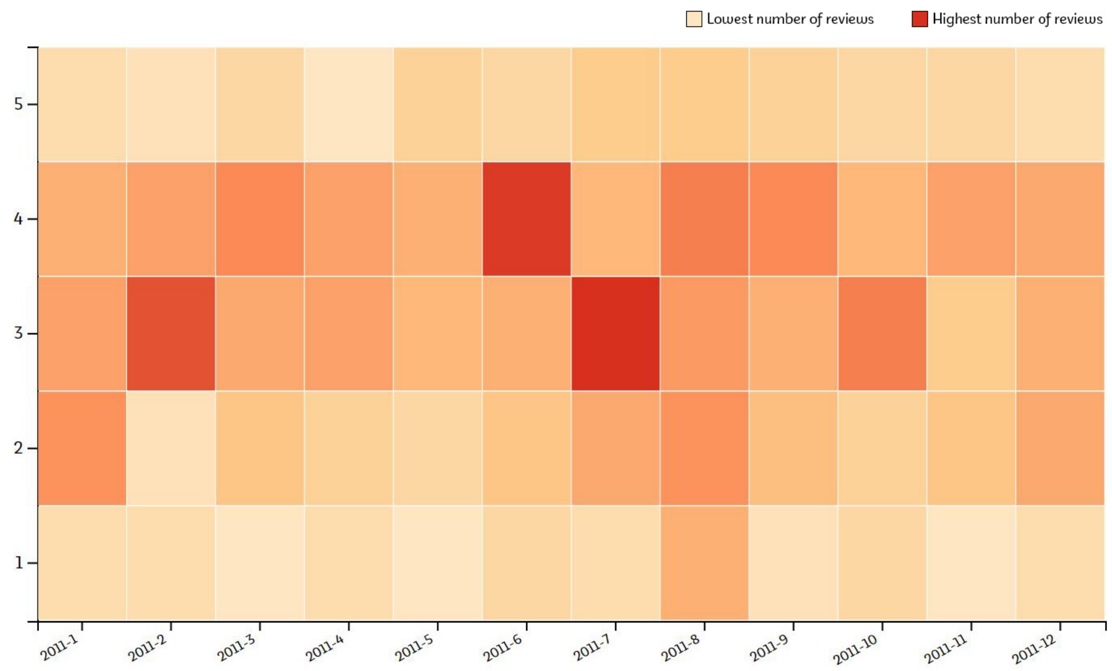


Figure 1.3: Visualisation of the temporal development of review scores for the year 2011 for restaurant 'Serendipity 3'. Along the x-axis are the different months. On the y-axis are the five different star scores.

• Temporal scaling for noticing and showcasing period-related patterns and seasonality is superior to a ranked list. – This helps the users pinpoint times of interest with greater accuracy. It will lead them to discover reoccurring topics and score changes.

• Presenting common words from a collection of textual reviews is a quick way to get a grasp on major topics and important issues. – This assumption helps the users identify major topics among reviews and restaurants.

• Combining the review scores with a short summary of the review content will reveal reasons for changing review scores. – This helps the users reveal the changes in textual reviews that have occurred. It is used to identify topics that make a difference in the overall scores.

• Distinguishing between all-around common words and time-specific common words leads to a better understanding of the content flow. – This helps the users spot and distinguish permanent topics from transient ones. It highlights the reasons why a score has changed.

• Filtering text reviews based on search criteria helps us to get a better grasp on specific events. – It helps the users identify topics of great importance. It gives them specific reasons for score changes and its correlation to a particular topic.

Additionally to focusing on proving the set hypotheses, the tool has to fulfil some performance criteria. It has to be lightweight and possibly used from a browser. It has to be able to load and execute operations, involving a large amount of data in a limited time. The tool should not seem cumbersome to users and the loading times should be unnoticeable.

In order to answer the research questions, before attempting to build a tool, this thesis will delve into the related work to the research. Chapter 2 will be about the already built technologies that will be influential and helpful for constructing our tool. Previous research and scientific works in regard to the topic are examined. Already existing techniques for the visual analysis of texts are compared, and their usefulness and limitations are considered.

In order to give answers to the questions asked by this thesis, we present Review Watcher. It is a novel, visual exploration interface that explores restaurant reviews and gives means of temporal analysis. Review Watcher combines visual interfaces for the exploration of time-dependent review ratings with real-time text processing algorithms, which sets it apart from already existing tools. Chapter 3 will explain more about how the tool was built. Design choices are discussed. General explanations about the concept of natural language processing and how it is used are provided.

Next, Chapter 4 will be about how the tool was implemented. Python scripts and libraries were combined with JavaScript capabilities to create an internet site for quick tool access.

Afterwards, in Chapter 5, we are showing how the tool was tested with real people. A qualitative evaluation approach was used to measure the insights people gathered from the tool. These insights will be the basis for determining the answers to the thesis questions. Confirming or rejecting the established hypotheses will be discussed and concluded in the last chapters of this thesis.

CHAPTER 2

Related Work

In this chapter, we will be looking at works related to our research topic. We are splitting the related work and state of the art chapter in two parts. The first part will be about existing tools and databases of customer reviews. We will be briefly mentioning tools that deal with collecting and presenting review data. The second part will be about the scientific works related to visual analyses of reviews and textual documents in general.

2.1 Customer Review Sites

Companies and businesses of the 21st century are more and more trying to be incorporated into the digital world. Businesses want to be visible by customers and easily accessible. Customers, on the other hand, want to find the best company for their own needs and sometimes share their opinion on the quality of the service they received as customers. As a result, big corporations like Google have their own databases with all kinds of businesses, which a person can easily view and leave a review about. Google Maps [RR01], for example, lets users check on a restaurant using their phone and leave a short text review and a numerical score reflecting their experience. Reviews can later be examined by other people, who might be at home trying to pick a restaurant to go to. The reviews that are left and the numerical score can influence the user's decision to visit the place or not.

Google, however, lets its users examine and vote not only for restaurants, but also for all kinds of businesses and even public places. Other tools and sites, such as Amazon [Bez01], also provide customer reviews for products and media. Researchers are examining both quantitative and qualitative aspects of online reviews and argue on how they affect product sales. They also examine how product sales influence the reviews over time [HLZ08]. As a result, business data analysts and decision-makers are looking for non-traditional methods and techniques for processing reviews. Such methods will support them when it comes to making important business decisions [CAB+20].

Two popular companies such as TripAdvisor [Kau01] and Yelp [SS01] have specialised in collecting and presenting customer satisfaction data. TripAdvisor has proven to be an influential site for tourists and an often-used service to make decisions about travel arrangements and entertainment possibilities. Reasons for its popularity all around the world stem from its plethora of hotel, restaurant, and entertainment services information and a dedicated user-base of reviews. Yelp is a tool that provides users with similar possibilities as TripAdvisor and became initially popular in the United States due to its extensive restaurant reviews collection. This prompted food critics and regular enthusiasts to often resort to the tool before and after visiting a restaurant. Initially, they use it to get recommendations and later to leave their own feedback. Some restaurant owners realising this emerging trend have put special care into courting their customers to leave positive feedback so that they can use this as a commercial for their business. TripAdvisor's extensive data collection has been a point of interest for researchers as well. For example, tools such as TRAS [NPB18] are created to analyse online travel reviews. TRAS has been employed on 60 000 reviews in order to provide travellers with a more personalised and better ranking-based recommendation system.

Yelp itself is constantly trying to improve its services and ways of presenting data. The site gives the ability to sort restaurants by the number of reviews, by the five-star rating score system, and listing reviews in chronological order for users to see. Their program Yelp Dataset Challenge [Inc01] provides people with the ability to freely explore and analyse their raw collected data and find insights hidden there. Since this thesis focus is on how restaurant reviews are changing over time, the Yelp Dataset Challenge is a good way to start exploring.

Research on text evaluation [LTL19] and sentiment analysis [TBH19] [MYW13] provide a good starting point for our work. Incorporating text analysis, using libraries such as the Stanford Topic Modeling Toolbox [Gro], into the mix, is another way to address the issues of text processing and automated sentiment analysis. Our focus is, however, on temporal evaluation. Some work has already been done relating to that area, with research on how sentiment categorisations of text collections should be handled over time [Rub18].

2.2 Visual Analysis of Text Over Time

Researchers in different fields have already investigated text analyses. Machine text processing and corpora content exploration techniques have been proposed and experimented on in a variety of contexts. Tools for content-analysis of literary texts [MKF$^+$02], news articles, and customer reviews [YLK16] have been sought and developed [Cao15]. Noticing content evolution over time is proven critical, both for understanding the past and for potentially predicting the future.

Having the proper tools to visualise these changes in content and the introduction of a dynamic term landscape, could be the key to properly understanding the data. Ways to spot topic evolution of social media text streams is a problem often addressed by researchers [LTL19]. One of the research methods is animated transitions for highlighting the revision history of similar textual documents [CDBF10]. Adapting it to work for news articles and reviews that are essentially different is a challenge.

A common goal of these studies is to find a way to provide a human-readable data out of a big data set impossible for humans to fully grasp by reading. Therefore, proper visualisation tools and technologies are being used to provide some solution. Dealing with text cluttering could often be challenging [ED07]. The visualisation strategies tend to get more complex with the introduction of time parameters. Visualising time-oriented data has come up with multiple methods, however, it is still a key issue in many application domains, including business and project management [AMST11].

Timeline folding is one suggested method for visually revealing informative patterns. It is discussed as the so-called Time Curves [BSH$^+$16] approach for evaluation of temporal data. While this approach shows changes occurring over different time periods, it does not provide any information regarding *why* the changes occurred in the first place. This is because it only conveys information in regards to the ordinal aspect of time.

The resulting developed tool of this Master thesis relies on already created research and concepts. It improves on them and also combines them for effectiveness. For example, tools such as Sparkclouds [LRKC10] have a way of visualising word occurrence frequencies. A Sparkclouds example view can be seen in Figure 2.1.

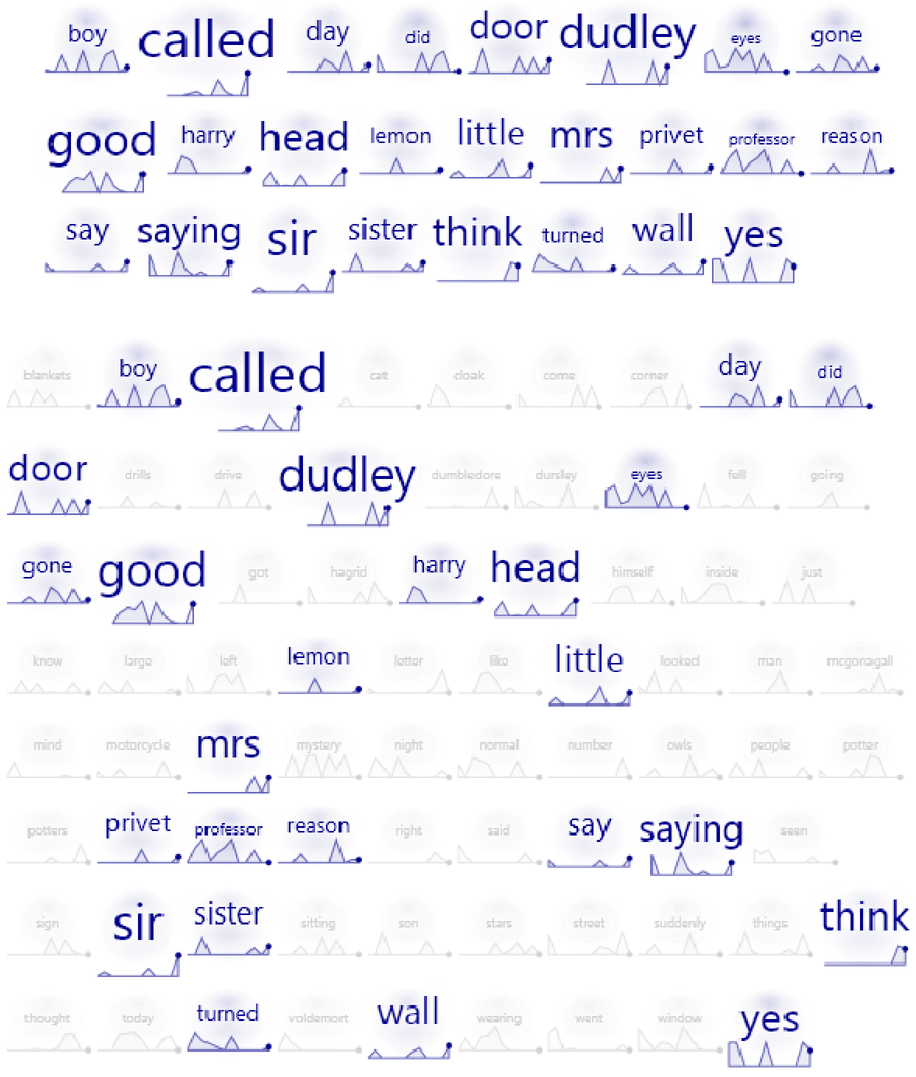


Figure 2.1: Sparkclouds showing the top 25 words for the last time point (12th) in a series. 50 additional words that are in the top 25 for the other time points can be (top) filtered out or (bottom) shown in gray at a smaller fixed-size font. (bottom) is used in the study from Sparkclouds: Visualising trends in tag clouds [LRKC10].

However, Sparkclouds is not very precise and lacks certain details, such as labels for the time-axis.

ThemeRiver is another example of research that has similar goals to our work [HHWN02]. ThemeRiver is a visualization that depicts thematic changes in a collection of documents over a period of time using a river metaphor. The idea, however, can be improved by adapting it with modern tools and technologies to produce a visually more pleasing representation. An example of a ThemeRiver view can be seen in Figure 2.2.

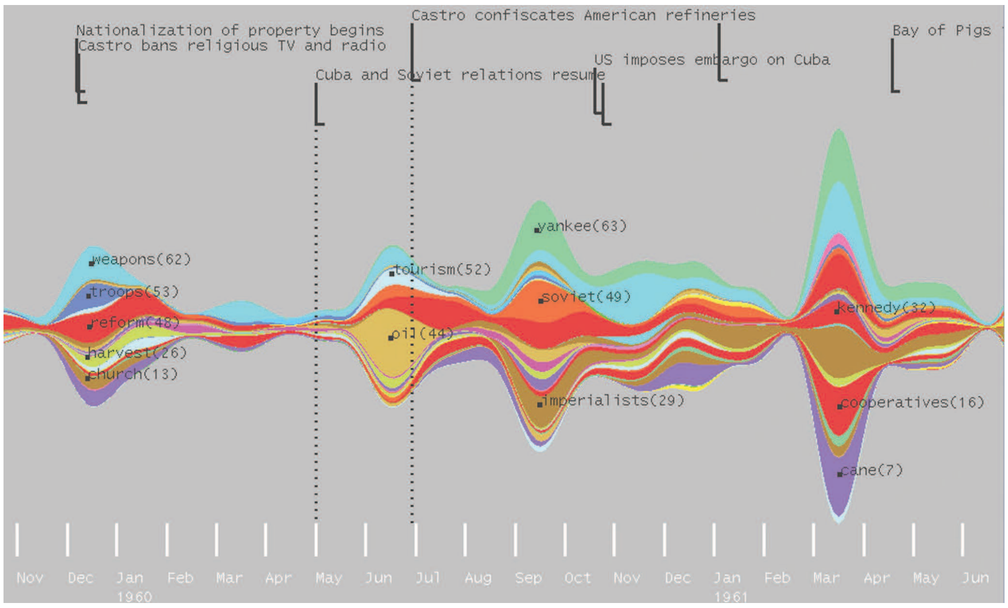


Figure 2.2: ThemeRiver uses a river metaphor to represent themes in a collection of Fidel Castro's speeches, interviews, and articles from the end of 1959 to mid-1961 [HHWN02].

ThemeRiver concentrates on themes from articles, which is a different dataset compared to user reviews. It handles fixed set of topics that become more and less prominent over time. While theoretically encoding the temporal evolution of major topics could be a useful approach, it is yet to be combined with the five-star review score system. A different layout, more suitable for the purposes of review analysts, is required. Another problem that ThemeRiver faces is the absence of event detection to aid in pattern predictions.

ThemeDelta [GJG$^+$15] is another example of research done in an entirely different context from user reviews. ThemeDelta is a visual analytics system for extracting and visualising temporal trends, clustering, and reorganization in time-indexed textual datasets. An example view of the system is visible in Figure 2.3.

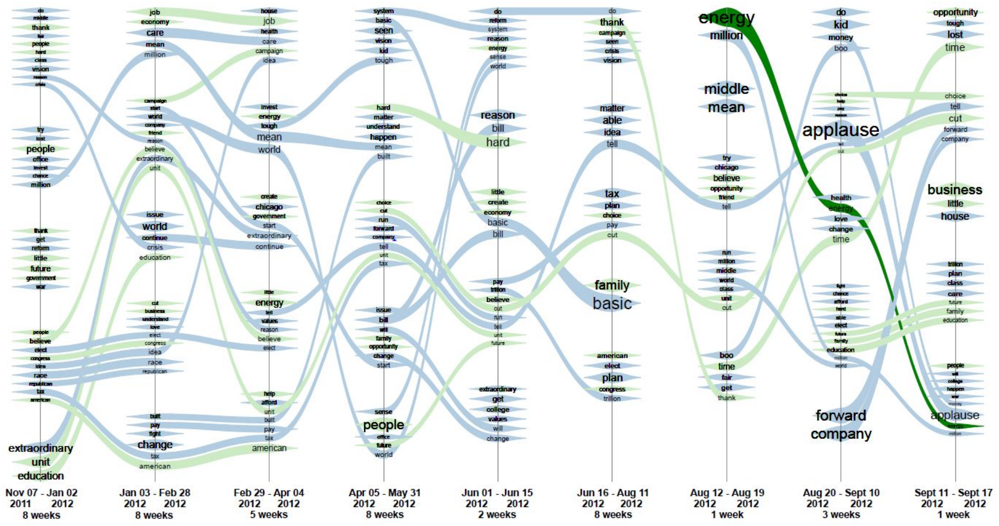


Figure 2.3: ThemeDelta visualisation for Barack Obama‘s campaign speeches during the U.S. 2012 presidential election (until September 10, 2012). Green lines are shared terms between Obama and Romney. Data from the “The AmericanPresidency Project” at UCSB [Pet01] [GJG+15].

This thesis aims to approach the problem of analysis of textual trends by dealing with a different type of dataset. That dataset is a combination of text and numerical scores. The thesis is additionally dealing with extensively larger dimensions when it comes to time and information processing and visualisation. Introducing scalable user interface and granularity options is also missing in ThemeDelta.

TextFlow is another example that includes relatively similar visualisation layouts, compared to the other examples given in this Section [CLT+11]. Some of the layouts present in the tool are shown as examples in Figure 2.4, Figure 2.5, and Figure 2.6.

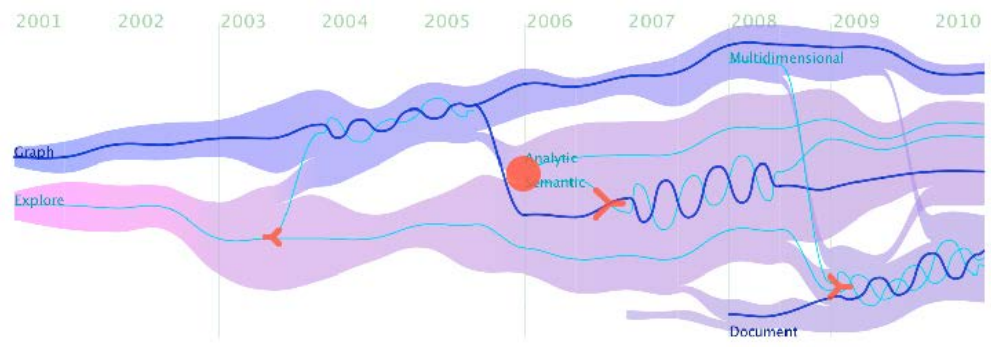


Figure 2.4: TextFlow example view. Selected topic flows of VisWeek publication data with thread weaving patterns related to primary keywords "Graph" and "Document" [CLT+11].

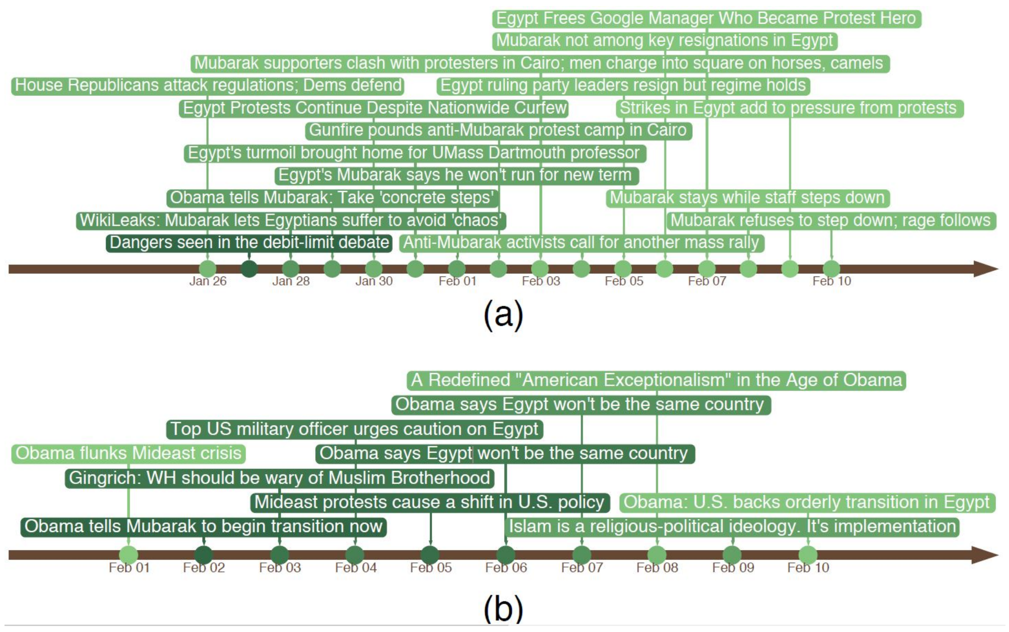


Figure 2.5: TextFlow example view. Comparing timelines extracted from different topics: (a) timeline extracted from topic related to “protest in Egypt”, which mainly describes protest events happening in Egypt; (b) timeline extracted from another topic related to “protest in Egypt”, which focuses on actions of the Obama government on events in Egypt [CLT+11].

TextFlow consists of an integration of visualisation and topic mining techniques, for analysing various evolution patterns that emerge from multiple topics. While it does show topic evaluation trends and keyword correlations, the tool also was not built for the combination of textual reviews with numeric scores.

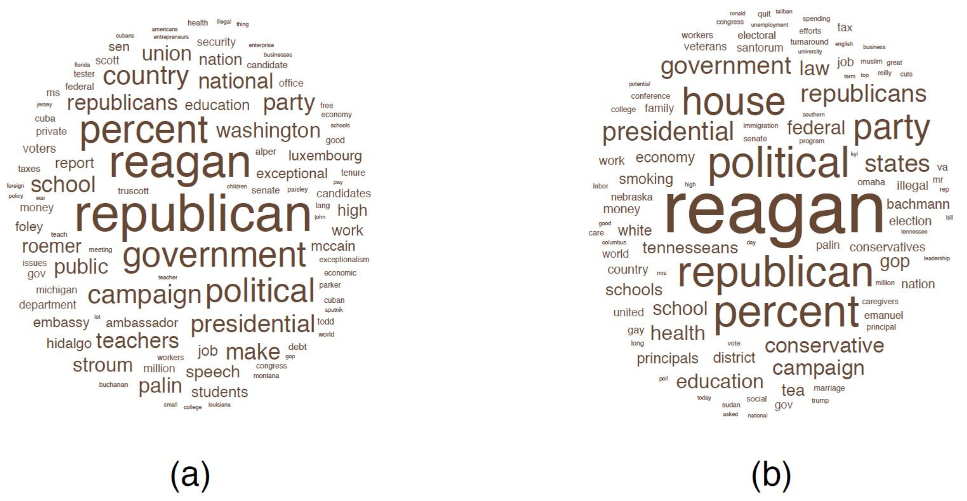


Figure 2.6: TextFlow example view. Comparing word clouds extracted from different topics: (a) word cloud extracted from a topic related to “republican/campaign”; (b) word cloud extracted from another topic related to “republican/campaign” [CLT+11].

Some researches that deal directly with the context of customer-reviews analysis for business establishments already exist. For example, OpinionSeer [WWL+10] directly addresses customer feedback on a temporal scale. The data used there are of hotel reviews taken from TripAdvisor and are very similar to the one used in our research. Some views that OpinionSeer uses are shown in Figure 2.7.

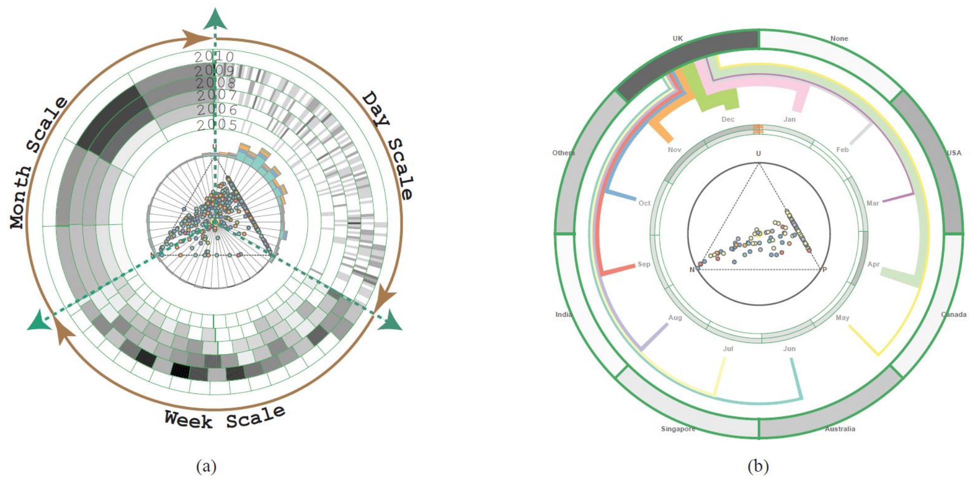
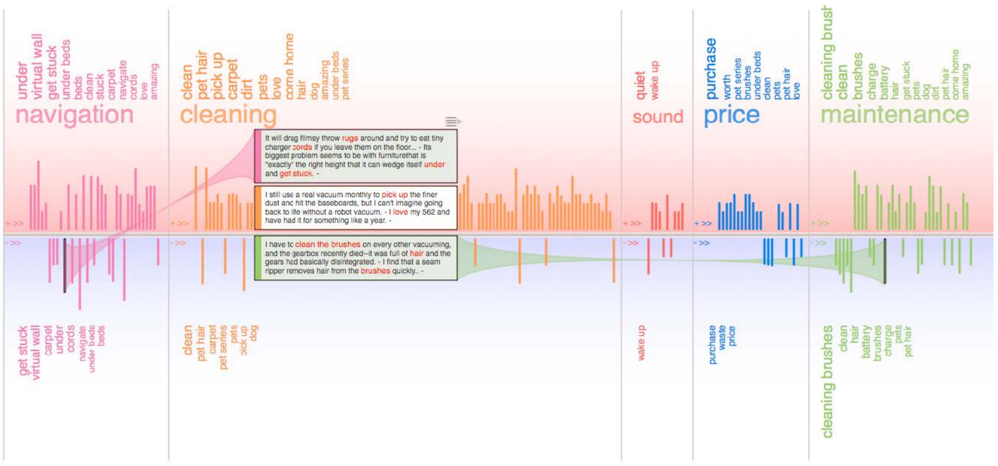


Figure 2.7: OpinionSeer example views. Temporal rings at different scales (month, week, and day); (b) Temporal and geographic rings where their relationships can be shown on demand by the curved belts. [WWL+10].

The questions that OpinionSeer tries to answer about the data, however, are different from the questions in this thesis. OpinionSeer concentrates more on grouping reviews and opinions based on where the reviews come from. It is not as interested in pinpointing temporal trends and the reasons for them. Therefore, most of their methods are not applicable to our goals.

Another business-oriented tool for customer reviews visualisation is OpinionBlocks [AYHK11]. It provides simple methods for context visualisation. The visualisation initially exposes text at the keyword level and then exposes snippets that the keywords are used in, and finally shows the snippets within the context of an entire review. Since the time component is missing, a temporal analysis is not possible. OpinionBlocks can be seen in Figure 2.8 for reference.



It will drag flimsy throw rugs around and try to eat tiny charger cords if you leave them on the floor... - Its biggest problem seems to be with furniturethat is *exactly* the right height that it can wedge itself under and get stuck. -

I still use a real vacuum monthly to pick up the finer dust and hit the baseboards, but I can't imagine going back to life without a robot vacuum. - I love my 562 and have had it for something like a year. -

I have to clean the brushes on every other vacuuming, and the gearbox recently died--it was full of hair and the gears had basically disintegrated. - I find that a seam ripper removes hair from the brushes quickly.. -

Figure 2.8: OpinionBlocks: Overview of snippets from consumer reviews organised by major product features (colour coded) that are discussed most frequently in the reviews and bi-polar sentiments (red region positive, blue region negative). Keywords for each feature-sentiment group are shown in their corresponding region. A single snippet is expanded, so that it shows other snippets from the same review. [AYHK11].

Similar methods that do include temporal analysis usually do not convey textual content changes over time [WHS+18]. Other researches on the topic explore how the reviews' numerical scores change over time [BBN12], which is a point of interest of this thesis as well. However, they do not show *why* the changes occur and are not connected to changes in textual reviews' topics.

CHAPTER 3

Methodology

In this chapter, we are discussing the different methods used in this Master thesis research. In Section 3.1, we describe how we handle the data we are presented. We take a look at what approaches we use to define data concepts and data characteristics that suit our goals. In the following section, we discuss the main interaction design concepts and visualisation concepts, that are employed in the tool creation process.

3.1 Data Characteristics

In this part of the thesis, we take a look at the characteristics of the database we are using. We explain how the data is input into the tool, how it is changed, and what is filtered away.

The original datasets used for the research are taken from Kaggle [uhU01] [Yel01]. They contain information collected by Yelp from October 2004 up to December 2017. One of the datasets is in a file format called Comma Separated Values (.csv) with a size of 31 MB, containing information about 174 567 unique businesses. The table also contains columns containing information about a business's name, multiple columns regarding its location, its average number of star ratings, its review count, whether or not the place is still open for business, and in which business category it is placed. One business can be placed into multiple categories. Since most of this information is not part of our interest, we strip down the dataset. We will be only interested in the unique id, the name of the business, the postal code, and the categories.

The businesses in the database are not exclusively gastronomy related. Therefore, we filter those places, which are not restaurants. We use the values in the category column to help us distinguish and filter the places that serve food from the rest. We only leave the businesses with categories that do include either 'food' or 'restaurant'. As a result, we have 69 079 businesses left in the database.

The second database provided by Yelp is again in the form of a .csv file. The size of the file is 3.7 GB. The table contained columns such as unique id of each review, unique id the author of the review, unique id of the business, star rating, date of issue, review text, and three additional rating scales – useful, funny, and cool.

Since we are not interested in the users, the column with user ids is discarded. The review id column is also discarded, since it will not be used in the future. The three additional rating scales are employed by Yelp users to rate and react to specific reviews. The three columns for the three different categories hold numeric values of the reactions each review has gathered. The three rating scales are also removed since the focus of the tool is not on them. The majority of the reviews containing missing values among these additional ratings also helped with the decision to leave them out. Fortunately, there were no missing values in the remaining cells of the table.

Having stripped down the unnecessary columns from the two datasets, we join the two datasets into one table. By doing this, we drop the reviews, which were not concerned with restaurant businesses. Following this, we try to filter the dataset even further. We filter all reviews that are written in a language that is not English. Afterwards, we count the reviews each separate restaurant has, and we remove all restaurants, which have less than eleven reviews. This is done because we do not consider these restaurants important and worth a graphical text analysis. If a restaurant has only ten reviews, the best way to understand what people have to say about it is probably by just reading the ten reviews. We are interested in restaurants for which just reading many, or even hundreds of reviews, is a tedious and troublesome task.

Since we are interested in how reviews are changing over time, we introduce one new column into the database. The column is called 'trend'. It represents the linear regression of the review scores for a single restaurant over time. We also combine together the columns with the restaurant name and the postal code. Thus we can just by name differentiate between restaurants from the same chain.

Once we have filtered all the unused data, we split the dataset into two separate sets. One is, once again, only containing the restaurants' information and the other one is only for the reviews. This separation is done in order to easily load the restaurants' data with a tool, without requiring to read the whole reviews' data. More on how the data is used by a tool is examined in Chapter 4.

The set with the restaurants has eight columns: unique id, restaurant name and postal code, trend, and five columns showing the number of reviews for each possible star score. Our database at this point has 38 894 rows representing 38 894 different restaurants.

The other dataset has five columns: unique id, review text, star rating, date of issue, name and postal code of the restaurant. It has 3 346 024 rows, one for each review. To this dataset, we add an additional column named 'lemmas'. Since the purpose of this thesis is to evaluate text corpora, we are incorporating natural language processing tools into our work. We are treating every review separately as a text and we are using tools to extract every single word in the text as a token. As a result, we get a bag of words that later can be compared with each other. Later, we remove the words, which are not considered important and do not have inherent semantic meaning. They are also referred to as stopwords. Typically, lookup tables are used to distinguish the stopwords from the rest [JM14].

After performing the tokenization of the text, we convert all the extracted tokens into each word's original lemma. The lemma is the base form under which the word is entered in a dictionary and assigned its place in that dictionary [HY07]. To demonstrate how lemmatization works we take a random text review from the database, which states:

'First, you go here for the view of the Bellagio fountains. Which is really good if you get a table outside and can stand the heat.

As for the food, been here 3 times and the food is way over priced for the quality. Had eggs benedict last time and the muffins were flat and there was very little sauce. Bloody mary was just ok.

Service was good, but if you want a meal with your view of the fountains, try somewhere else.'

The lematization process returns the following list of words from the review above:

'go', 'view', 'bellagio', 'fountain', 'really', 'good', 'get', 'table', 'outside', 'stand', 'heat', 'food', '3','time', 'food', 'way', 'price', 'quality', 'egg','benedict', 'last', 'time', 'muffin', 'flat','little', 'sauce', 'bloody', 'mary', 'okay','service', 'good', 'want', 'meal', 'view','fountain', 'try', 'somewhere', 'else'.

Later on, we use these extracted tokens to quickly reveal what are common words in a text collection. By converting the word tokens to lemmas, we eliminate the possibility to confuse two forms of the same word for two completely different words. We perform all these transformations in the pre-processing stage because these operations are computationally expensive. Additional information on computation times and benchmarks is provided in Section 5.1 of the thesis.

Another considered method for text processing was to extract word tokens from the text and to reduce them to their stems. Stemming is an operation that reduces the feature dimensionality by stripping words from their suffixes using simple rules. These rules primarily deal with conjugation rules of languages, reducing words to the word stem. However, these word stems are not complete words and the procedure does not work with irregularities [Por06]. Taking the same review from before as an example, we show the results from a stemming process:

'go', 'here', 'view', 'bellagio', 'fountain', 'realli', 'good', 'tabl', 'outsid', 'stand', 'heat', 'food', 'here', '3', 'time', 'food', 'way', 'over', 'price', 'qualiti', 'egg', 'benedict', 'last', 'time', 'muffin', 'flat', 'littl', 'sauc', 'bloodi', 'mari', 'ok', 'servic', 'good', 'want', 'meal', 'view', 'fountain', 'tri', 'somewher', 'els'.

A different approach to tokenizing every single word is to use n-grams. This would mean to preserve the textual order of adjacent words and encode all occurring sequences of multiple words. For example, separating an adjective and the noun it is referred to, can be collected as a bi-gram. One method of forming bi-grams is to store every two adjacent words from a sentence. To show an example, we once more use the sample review from before:

’first you’, ’you go’, ’go here’, ’here for’, ’for the’, ’the view’, ’view of’, ’of the’, ’the bellagio’, ’bellagio fountains’, ’fountains which’, ’which is’, ’is really’, ’really good’, ’good if’, ’if you’, ’you get’, ’get a’, ’a table’, ’table outside’, ’outside and’, ’and can’, ’can stand’, ’stand the’, ’the heat’, ’as for’, ’for the’, ’the food’, ’food been’, ’been here’, ’here 3’, ’3 times’, ’times and’, ’and the’, ’the food’, ’food is’, ’is way’, ’way over’, ’over priced’, ’priced for’, ’for the’, ’the quality’, ’quality had’, ’had eggs’, ’eggs benedict’, ’benedict last’, ’last time’, ’time and’, ’and the’, ’the muffins’, ’muffins were’, ’were flat’, ’flat and’, ’and there’, ’there was’, ’was very’, ’very little’, ’little sauce’, ’sauce bloody’, ’bloody mary’, ’mary was’, ’was just’, ’just ok’, ’ok service’, ’service was’, ’was good’, ’good but’, ’but if’, ’if you’, ’you want’, ’want a’, ’a meal’, ’meal with’, ’with your’, ’your view’, ’view of’, ’of the’, ’the fountains’, ’fountains try’, ’try somewhere’, ’somewhere else’.

The extraction of bi-grams and tri-grams, however, is computationally and temporally expensive. That can be seen later on in Chapter 5. Therefore, this method of extraction is also abandoned.

Once we have created and properly formatted our two new datasets, we upload them into a server and make them accessible to the tool. If we would like to update the database, we will be able to just add new entries to the database and the tool will start working with the new data. The newly added data rows, however, do need to conform to the established data rules. This would mean that when adding new reviews, a new list of lemmas needs to be extracted from the text and added as a column before starting the tool. Adding new restaurant reviews would also mean that the trend value of a restaurant would also change. For this, we have to calculate the new values and update them in the database accordingly. Adding new reviews would also mean we have to increase the overall number of reviews a restaurant has for a specific star rating. This process is expected to be computationally inexpensive. On the other hand, the trend computation could be expensive. More on that is elaborated in the Section 4.1 and Section 5.1.

3.2 Visualisation and Interaction Design Concepts

A few visual concepts were created during development as of how the tool should look like. Some of them were discarded as not intuitive or not displaying useful information. The used visualisation and interaction concepts are described in this section, along with the justifications on why they were chosen. The section is split into to two sub-sections based on the two view approaches we employ in the tool.

3.2.1 General View

We begin by describing the General View, which we use as a basis for the tool's landing page. The original concept for the landing page, as shown in Figure 3.1, consists of a page showing some of the restaurants in a stacked bar chart.

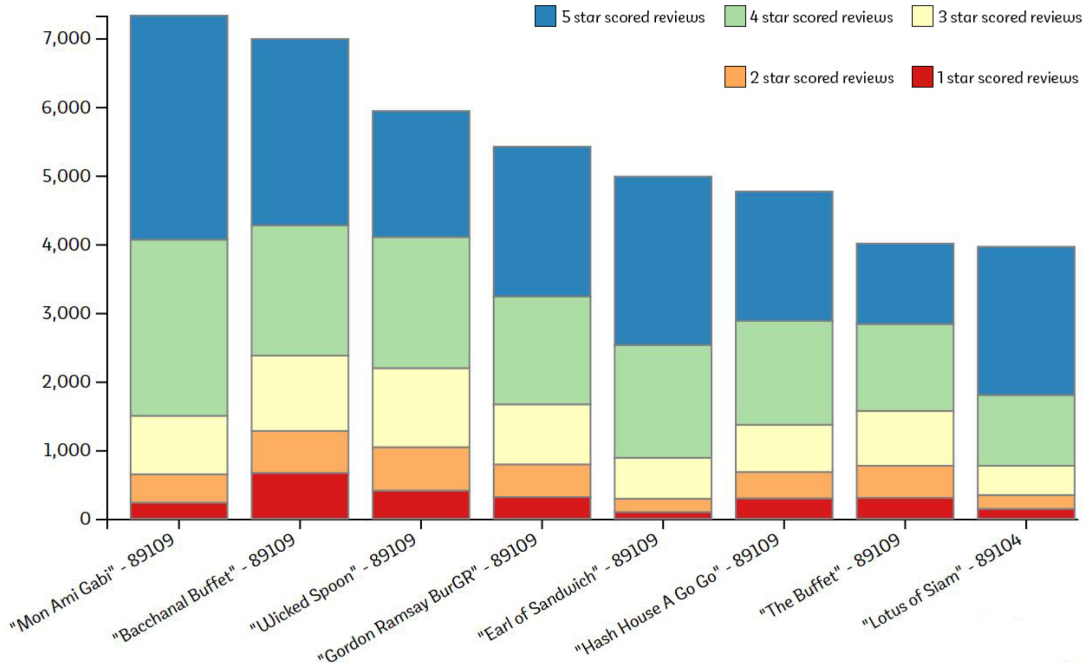


Figure 3.1: Stacked bar chart with the top eight most reviewed restaurants from a dataset, consisting of 38 894 restaurants. Along the x-axis are restaurant names. On the y-axis the number of reviews. Warm bar colours represent low review scores and cold colours represent high review scores.

The bars represent the number of reviews a single restaurant has for each star rating. This view is providing a quick overview of the popularity of a restaurant and the most common review scores it is getting from customers. Returning back to the established hypothesis, we juxtapose this method of visualisation to a conventional ranked list of restaurants. We argue that this view makes it easier for users to find restaurants with diverse scores and compare them at a glance. This argument is evaluated in Chapter 5.

Data, for only ten restaurants, is shown on the screen at a time in order not to clutter the screen with too much data. Since the dataset with restaurants can be huge in size, we introduce different pages showing the different sets of restaurants in the database. While changing the number of items per page is an existing functionality, having a default number of ten items is considered the standard for the design of result pages in search engines [KA15].

In Review Watcher we also have a set of buttons we tied with different functionality for more control over the General view. The buttons help us adjust how many restaurants are shown per page. They help to navigate through different sets of restaurants and enable us to sort them based on some chosen criteria.

The tool is able to arrange the restaurant in ascending or descending order based on a few criteria: the number of reviews, the average star rating, the weighted star rating based on the number of reviews, the variance of the star ratings, the trend, and the name of the restaurant in alphabetical order.

The sorting function we consider critical for answering the research questions. The number of reviews sorting, as seen in Figure 3.1, is the same as the Most Reviews sorting function that is available in the original Yelp website. The average star rating sorting, as seen in Figure 3.2, is identical to the Yelp Highest Rated sorting.

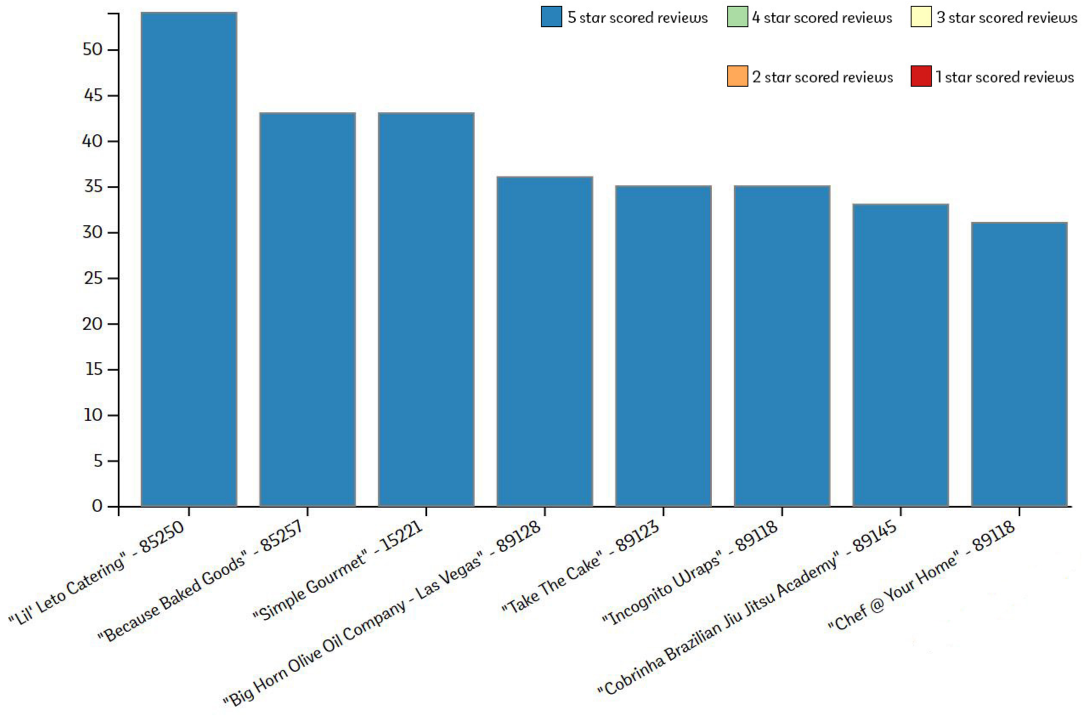


Figure 3.2: Stacked bar chart with the top eight highest average star rating restaurants from a dataset, consisting of 38 894 restaurants. Along the x-axis are restaurant names. On the y-axis are the numbers of reviews. The blue bars indicate these restaurants have only five star reviews.

The weighted star rating based on the number of reviews is developed with the Yelp Recommended sorting in mind. However, since Yelp does not provide the formula used to calculate the Recommended sorting, we resort to using our own sorting algorithm, which is more similar to the IMDb.com movie rating algorithm. An example sorting is show on Figure 3.3 and more information on the chosen algorithm is available in Section

4.2 of this thesis.

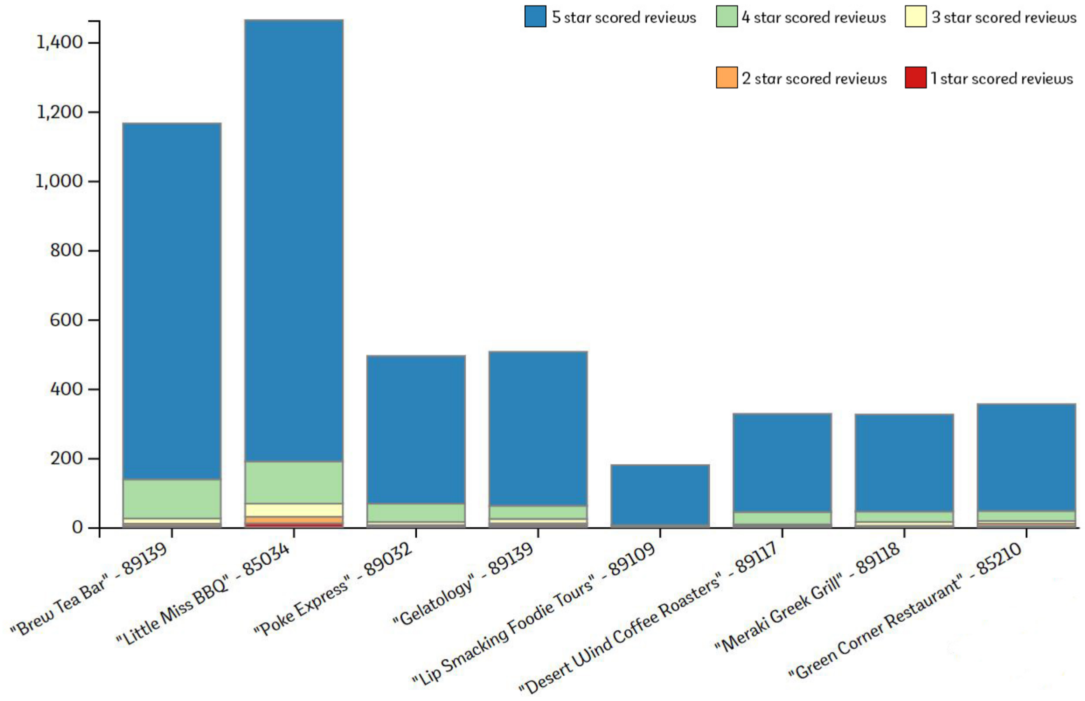


Figure 3.3: Stacked bar chart with the top eight best weighted restaurants in a dataset, consisting of 38 894 restaurants. Weight score is calculated in regard to the number of reviews and the star score. Along the x-axis are restaurant names. On the y-axis are the numbers of reviews. The large blue bars indicate these restaurants have mostly, but not exclusively, five star reviews. This is indicated by the blue bars being larger in height, compared to the different coloured bars.

Another method for sorting the review score variance is demonstrated in Figure 3.4. It is used to help users spot restaurants with polarising reviews and restaurants with drastic changes in review scores. The sorting can be a sign of interesting trends among reviews and fluctuating scores over time.

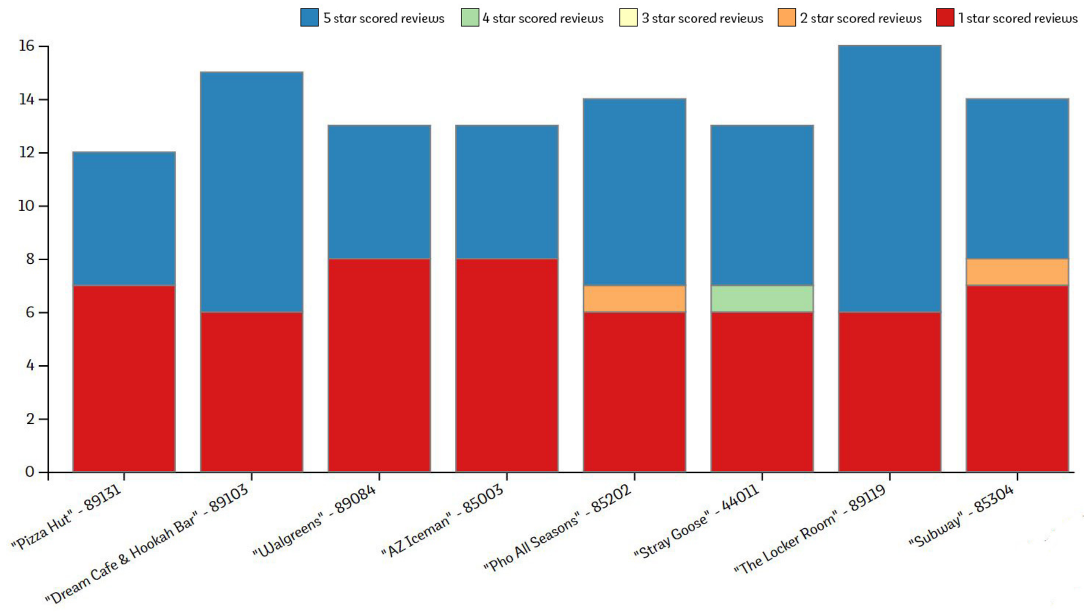


Figure 3.4: Stacked bar chart with the top eight highest score variance restaurants from a dataset, consisting of 38 894 restaurants. Along the x-axis are restaurant names. On the y-axis are the numbers of reviews. The large blue and red bars indicate these restaurants have both a lot of five and one star reviews, respectively.

Trend sorting is introduced, also in order to help users spot restaurants with polarising reviews, but it is also used for spotting restaurants with major review changes over time. Using the pre-calculated trend values, mentioned in Section 3.1, we sort the restaurants from the ones with the highest increase of score rating over time to the highest decreasing score over time, or vice-versa. This helps us easily identify restaurants with changing review scores over time. It also points us towards restaurants with strong, permanent changes of scores. Noticing score trends is an essential requirement to answering our research questions.

Another function of the tool is to represent what are the most common words used in the reviews for a particular restaurant. This aims to give the tool's user a basic grasp of what the reviews and the restaurant are about. For this purpose, we extract the most common words from the collection of lemmas in the database. One way to present them to the user is as a list, showing what are the ten most common words and the number of times they appeared. Review Watcher does that, by displaying a tooltip, when a user clicks on a restaurant name, as shown in Figure 3.5. If a person wants to view more information about a particular restaurant, they can select that restaurant and be transported to the Detail View.

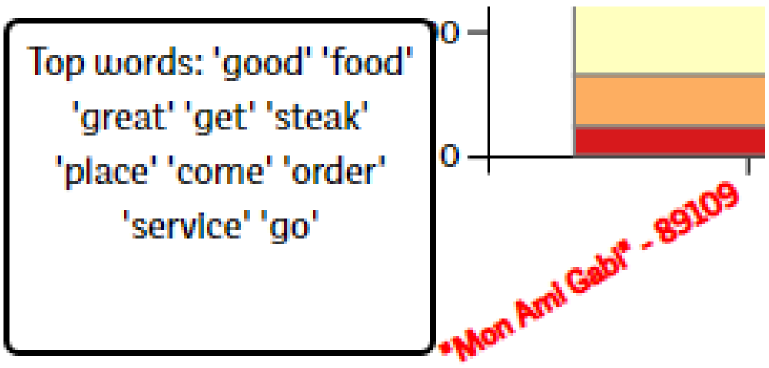

Top words: 'good' 'food' 'great' 'get' 'steak' 'place' 'come' 'order' 'service' 'go'

Figure 3.5: A tooltip showing the top ten most common words in reviews for the restaurant 'Mon Ami Gabi', sorted by number of occurrences.

3.2.2 Detail View

The Detail View uses a few visualisations to highlight different information about one particular restaurant. It is a more concentrated approach in the sense of scope than the General View.

One visualisation technique that is used, is in the form of a word cloud, as shown in Figure 3.6.

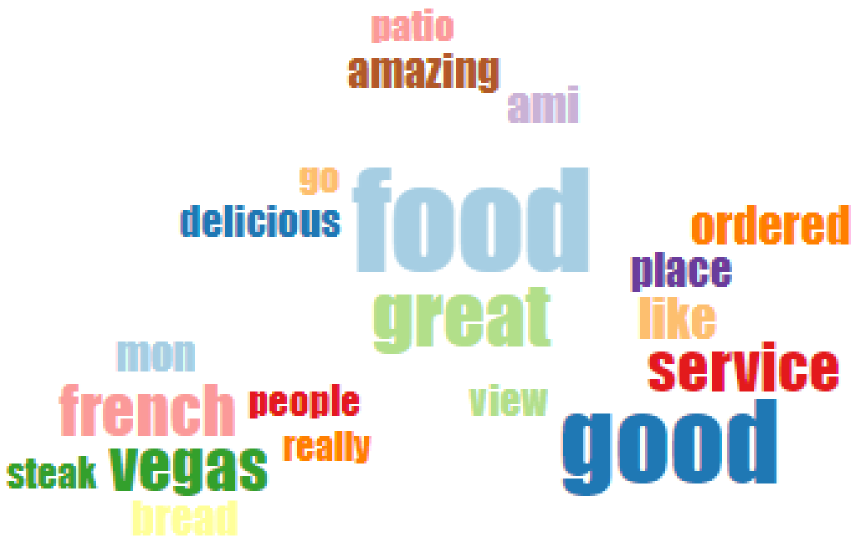


Figure 3.6: Word cloud of the 20 most common words used in the reviews for restaurant 'Gangnam Asian BBQ Dining'. The larger the font, the more common the words are. The positioning and the colour of the words is random and of no consequence.

The word cloud is considered a more visually appealing way of presenting words, compared to an ordered list. It is something that attracts people's attention [HLLE14]. This is also aided aided by the use of different colours, word positions, and font sizes. With different font sizes however, there is the risk that longer words appear bigger than shorter ones. This could mislead the reader to think a longer word is more common than a shorter word, when that is not the case. In the tool, we show a word cloud of the most common words from all reviews for a restaurant. We also separate word clouds for different periods of time. For example: A user may select a specific year, let us say the year 2015. The new word cloud is formed and is populated with the 20 most common words from reviews of the restaurant, only for the year 2015.

Cycling through different time periods, the changing word clouds would reveal to the user the changing most common words. We use this method as one way to indicate that there are changes in the reviews over time. It helps us to answer questions, such as what are topics that affect a restaurant over all time. It points us towards changes of topics in textual reviews. It also helps us identify topics of great importance at a glance.

Another function the tool provides is the ability to search for reviews containing specific search words among a text collection. This search function provides users with the opportunity to find out why users are mentioning a specific word or combination of words. As an example, if a user wants to view what reviewers are saying about the burgers in a restaurant during the year 2015, they can do so. Setting the appropriate search filters and searching the word 'burger' will provide them with the requested information. The search queries the database of reviews for ones that fulfil the requested criteria and filter out everything else.

This functionality is crucial for helping users to get a better grasp on specific topics. The function directly addresses out previously defined hypothesis. It helps to identify topics of importance and which in turn could be useful to find specific reason to why reviews changed. Combining this functionality with time filters and review score filters, lets users drill down the data and correlate content changes to score changes.

We show the word cloud and the search options, and results, on the same page, because we want the users to make the connection between the functionalities. In addition to that, on the same page, we also present a visualisation of the review scores over time.

When coming up with a view that shows how star ratings of reviews are changing over time, a few different visualisation techniques were considered. Their goal was to provide visualisation at what star scores, users were rating a restaurant at a particular time. Considering we are dealing with data that have five categories of review scores and time of issue of each review, we look for the best way to show that data. Considering the data characteristics, most of the methods discussed in Chapter 2, such as the ones used for ThemeRiver, are not properly suited for this task.

One example view that was not used in Review Watcher was showing a box-plot similar to the one in Figure 3.7.

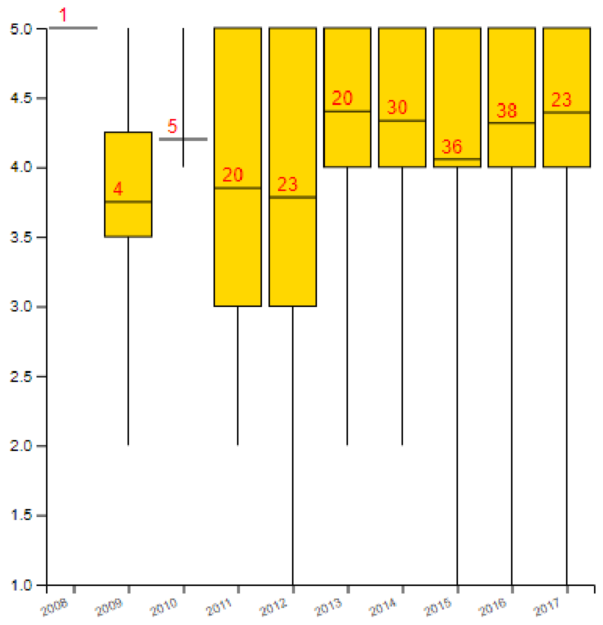


Figure 3.7: Box-plot showing review score distribution over time. On the x-axis are the different years. On the y-axis are the five different star ratings. The numbers in the separate boxes show how many reviews are there for the particular time. The lines below the numbers show the mean values of the review scores.

This view was considered however too restrictive in regards to scale and not revealing enough information. An example issue with that view is that even though a restaurant must have more than ten reviews, it might have very few at certain periods of time. Therefore, the box-plot becomes impossible to interpret, because the descriptive statistics that are necessary to construct a box-plot, cannot be computed.

Another considered approach was the inclusion of a timeline with confidence bands, as the one shown in Figure 3.8. While this view is a fine example of a continuous representation of discrete data over time, it is not without its limitations. Mainly, the view does not link the changes of review scores to the changing textual reviews. This view does not give us answers to *why* review score changes occurred. Therefore a method to link it to the text reviews or word clouds is sought.

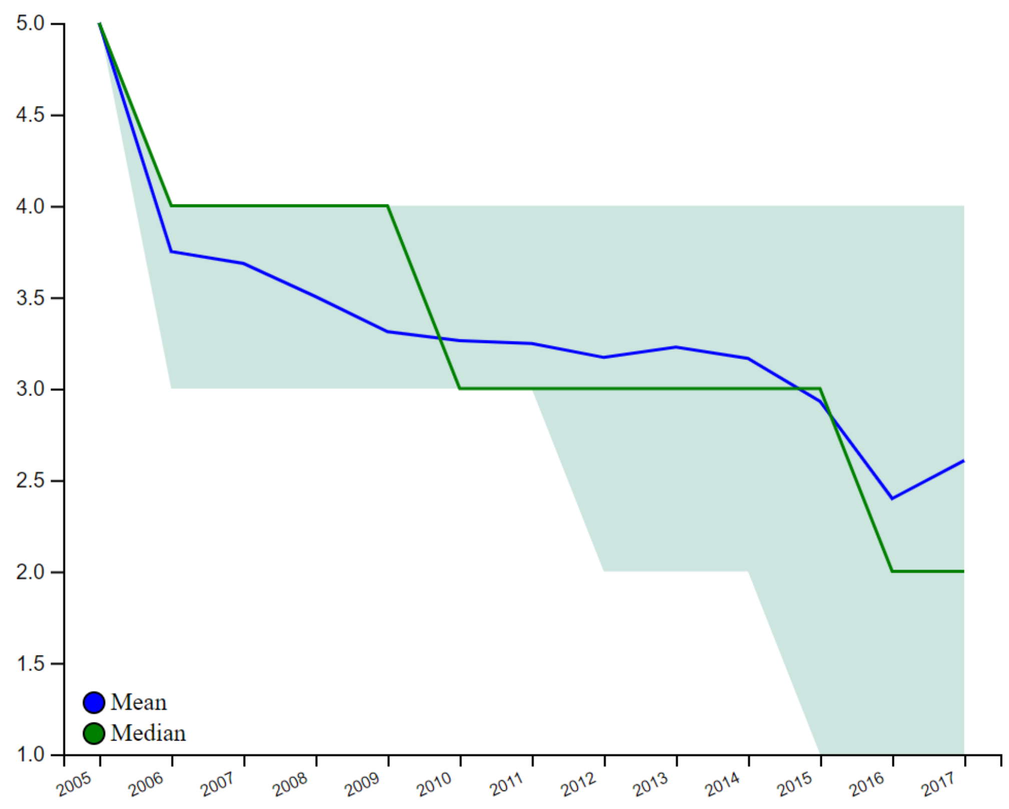


Figure 3.8: Timeline with confidence bands for the score ratings of restaurant 'MGM Grand Hotel'. The x-axis shows the years. The y-axis shows the different star scores. The mean and the median of the star scores for a particular year are labelled. The light green confidence band shows the reach of the score ratings.

Another potential view, as seen in Figure 3.9 and in Figure 3.10, was discarded due to being too cluttered with information. This view carries the idea to show multiple restaurants in an effort to compare them to one another.

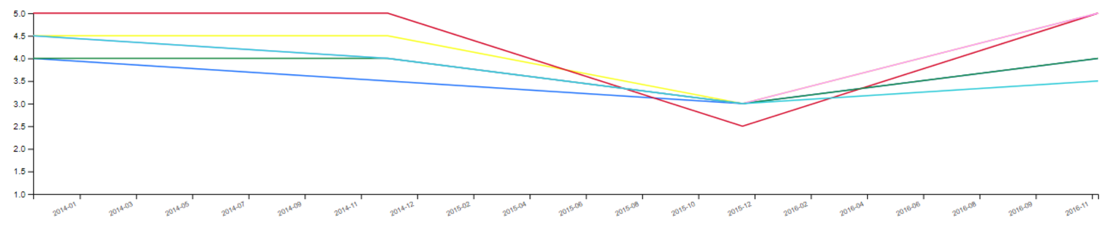


Figure 3.9: Average review star score over time for six restaurants. The line chart has different months as the x-axis and average star ratings on the y-axis. The presented set of restaurants is shown in Figure 3.10. Every bar colour relates to the colours on the line chart.

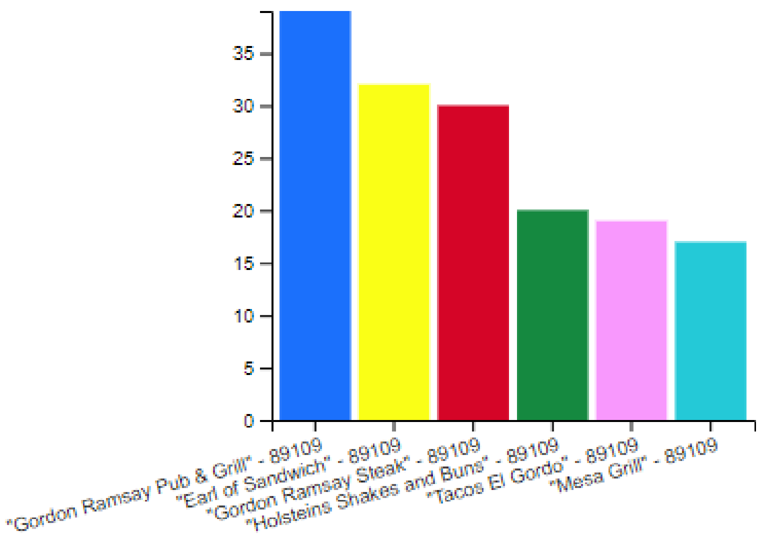


Figure 3.10: A bar chart with a set of restaurants is shown, along with their total number of reviews. Every bar colour relates to the colours on the line chart in Figure 3.9.

While this view does show the temporal evolution of the average review score, it just shows the mean time line. Therefore the variance indicator is completely lost.

The end result was to create a matrix view as the one in Figure 3.11. The matrix is structured as a temporal heatmap. On the heatmap, the x-axis corresponds to the time periods and the y-axis corresponds to the five different star ratings. The colour of the matrix cells denote the number of reviews. This view was considered the most expressive when showing variances and time-related changes, without being too confusing. Changing the granularity of the time periods shown in the matrix is also an important feature. This way a user explore the data in-depth and concentrate on a specific time period, or even manually search for seasonality patterns.

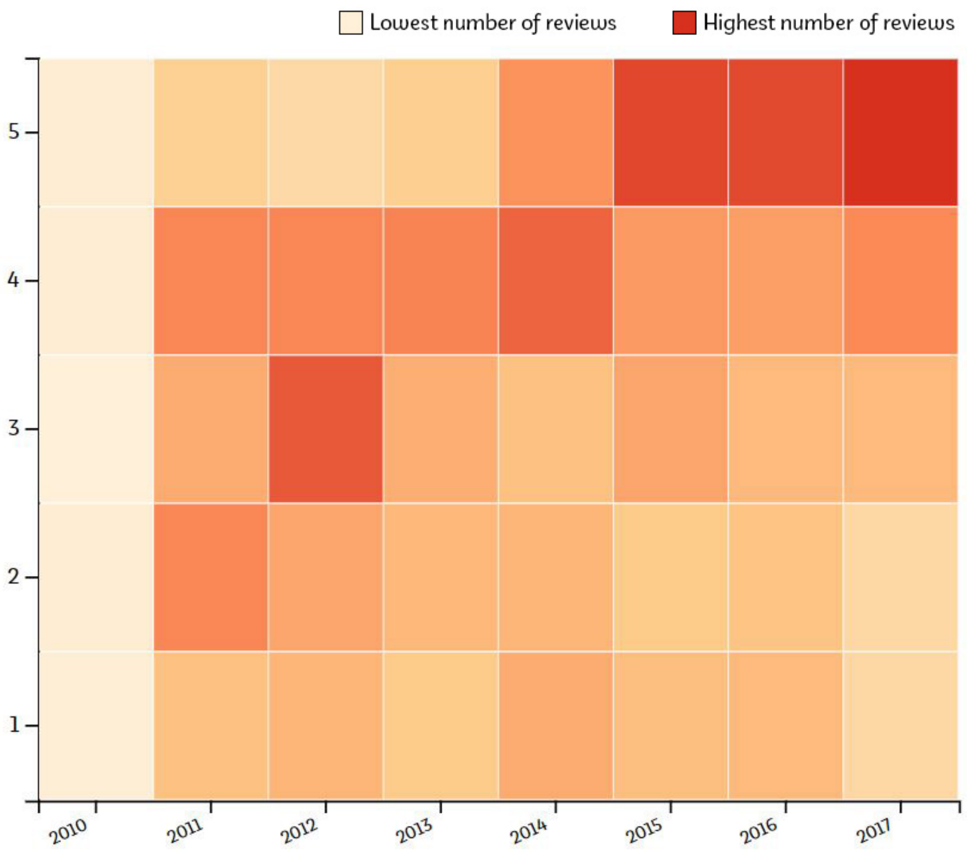


Figure 3.11: Temporal heatmap of review scores for restaurant 'China Poblano'. Along the x-axis are the different years. On the y-axis are the five different star scores. Darker colours in cells denote larger number of reviews for that cell. White cells show there are no reviews.

Additionally, cells can be easily selected by the user to obtain a summary of the reviews for a particular score at a particular time interval. In order to do that, the matrix was also combined with the most common words function of the tool. Upon clicking on a cell, the user is able to see a list of the most common words contained in the reviews from that cell. The function is as the one demonstrated in Figure 3.12. The same function is applied to choosing a specific time period from the matrix's axis or a specific star rating.

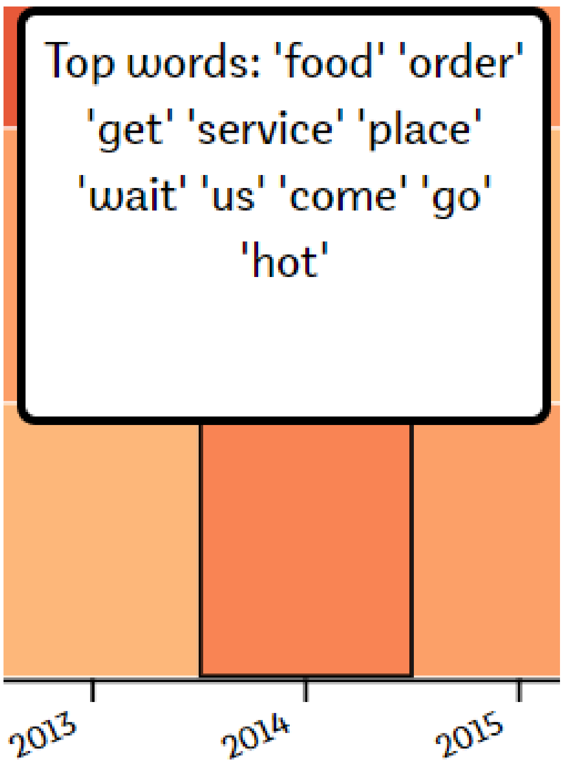

Top words: 'food' 'order' 'get' 'service' 'place' 'wait' 'us' 'come' 'go' 'hot'
2013
2014
2015

Figure 3.12: Tooltip showing the most common words from reviews for the restaurant 'Serendipity 3'. The words are only for 1 star reviews from the year 2014.

This functionality is added in a effort to answer the question of why did review scores change at a particular time for a particular restaurant. For example, it can show us the emergence of new commonly used words in a point of time, where review scores are changing general direction.

The function can also provide users with ideas what is the difference in content of favourably scored and unfavourably scored reviews. Thus,it can reveal topics that make a difference in the overall scores. We use this visualisation method to prove the hypothesis that presenting common words from a collection of textual reviews is a quick way to get a grasp on major topics and important issues. Evaluation of the hypothesis is present in Section 5.3.

One potential problem that occurs when using the most common words (also referred to as top words) feature is that some words are often appearing almost everywhere. Words like 'food', 'good', or just the name of the selected restaurant can often be spotted among the top words results. Sometimes, it happens that a restaurant with seemingly fluctuating reviews is rarely changing its top five most common words. As a result, a method to better filter words, with major significance to specific time-periods, had to be used. Therefore, a term-weighting scheme is introduced. Term-weighting assumes that not all words in a text contribute equally to the semantics.

The method we are using in this thesis is term frequency-inverse document frequency (TF-IDF) [SJ88]. With it, we determine the weights of all the words in a particular text. This is calculated by multiplying the term-frequency of a word in a document with the inverse document frequency. The inverse document frequency is a measure that determines how much information a word provides dependent on the corpus. This weighting considers words as important if they are common in a text and also specific to it. The motivation behind this choice is that words that are frequent in a document but nowhere else hold relevant information for the document that is not available in the remainder of the corpus.

Coming back to previous example with the most common words in Figure 3.12, we apply the TF-IDF algorithm to extract the discriminative words for the same set of reviews. In Figure 3.13 we can see the difference in the results.

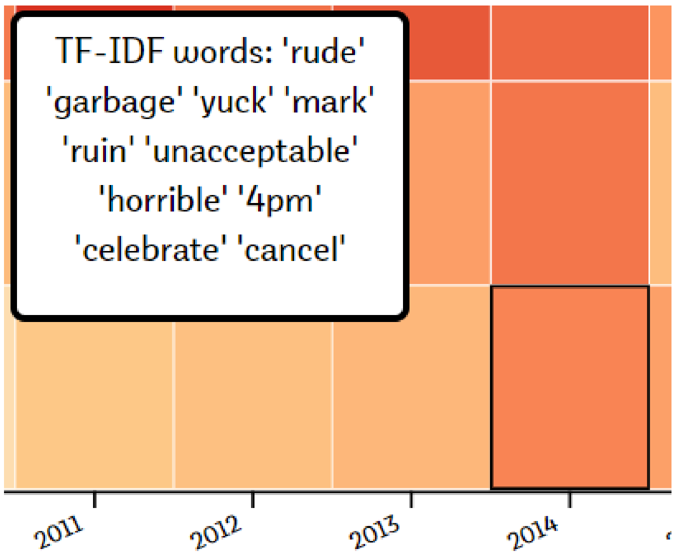

TF-IDF words: 'rude' 'garbage' 'yuck' 'mark' 'ruin' 'unacceptable' 'horrible' '4pm' 'celebrate' 'cancel'

Figure 3.13: Tooltip showing the words with the highest TF-IDF scores from reviews for the restaurant 'Serendipity 3'. The words shown are only for 1 star reviews from the year 2014.

Since we have already converted the text reviews to collections of lemmas, we can easily calculate the TF-IDF score on top of the lemmas. This weighting scheme allows us to list words that are of great importance to a specific time period, but are not common for other time periods. Which time period we are interested in, can be chosen from the temporal heatmap. A different way to divide corpora applying TF-IDF is to compare the words from the reviews for a specific star rating to the other words from reviews from different star ratings of the same restaurant. A combination of the two corpora-dividing scenarios is also possible and present in Review Watcher.

We use the TF-IDF approach in combination with the temporal heatmap, in an effort to help users better understand the content changes over time. This once again aids users to figuring out why review scores changed and what changed in their content. It also tends to help noticing transient topics. The method of finding common words also can be used in the comparison to distinguish between the permanent topics of importance for reviewers.

CHAPTER 4

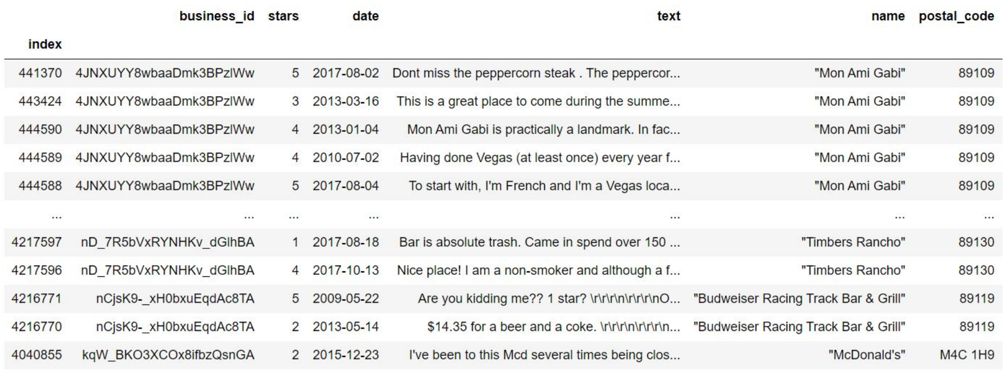
Implementation

This Chapter of the thesis will be concerned with how Review Watcher is implemented. We use the document manipulation and data visualisation JavaScript library D3 [Bos01] and the Python-based Flask [Ron01] to create an online tool that is lightweight and can be used from any browser. We transform the database we have originally gathered from Yelp using Python scripts and natural language processing algorithms. We store the resulting dataset in a MongoDB [MHR01] server and make it accessible to our tool.

4.1 Database pre-processing

As mentioned before in Chapter 3, the data used by Review Watcher has to be processed and changed beforehand. After acquiring the two datasets (restaurant information and review information) from Kaggle as .csv files we load them in a Jupyter Notebook version 6.0.1 [PG01] with a Python 3.6 script. Python was chosen for this tool because of its ability to provide libraries for data processing, natural language processing and its compatibility with JavaScript, which will be later used for the front end. For easier readability of the data, we use default Python libraries, combined with Pandas [McK01]. These tools will be used for all parts of the back-end implementation of the tool and the pre-processing part. We load the dataset into a data frame. Using Pandas' filters, we drop all entries with categories that do not include either 'food' or 'restaurant'. Then we drop all the unnecessary columns until we have just business id, name, and postal code left.

We also drop all unnecessary columns from the reviews' data frame. For the star rating column, we use NumPy in order to set the type to Int8 for further processing and filtering. Setting proper column types and removing unnecessary information from Pandas data frames reduces processing and filtering time. Ultimately, the data frame looks like the one in Figure 4.1.



index	business_id	stars	date	text	name	postal_code
441370	4JNXUYY8wbaaDmk3BPzlWw	5	2017-08-02	Dont miss the peppercorn steak . The peppercor...	"Mon Ami Gabi"	89109
443424	4JNXUYY8wbaaDmk3BPzlWw	3	2013-03-16	This is a great place to come during the summe...	"Mon Ami Gabi"	89109
444590	4JNXUYY8wbaaDmk3BPzlWw	4	2013-01-04	Mon Ami Gabi is practically a landmark. In fac...	"Mon Ami Gabi"	89109
444589	4JNXUYY8wbaaDmk3BPzlWw	4	2010-07-02	Having done Vegas (at least once) every year f...	"Mon Ami Gabi"	89109
444588	4JNXUYY8wbaaDmk3BPzlWw	5	2017-08-04	To start with, I'm French and I'm a Vegas loca...	"Mon Ami Gabi"	89109
...	...	...	...	...	...	...
4217597	nD_7R5bVxRYNHKv_dGlhBA	1	2017-08-18	Bar is absolute trash. Came in spend over 150 ...	"Timbers Rancho"	89130
4217596	nD_7R5bVxRYNHKv_dGlhBA	4	2017-10-13	Nice place! I am a non-smoker and although a f...	"Timbers Rancho"	89130
4216771	nCjsK9-_xH0bxuEqdAc8TA	5	2009-05-22	Are you kidding me?? 1 star? \r\r\r\n\r\r\r\nO...	"Budweiser Racing Track Bar & Grill"	89119
4216770	nCjsK9-_xH0bxuEqdAc8TA	2	2013-05-14	$14.35 for a beer and a coke. \r\r\r\n\r\r\r\n...	"Budweiser Racing Track Bar & Grill"	89119
4040855	kqW_BKO3XCOx8ifbzQsnGA	2	2015-12-23	I've been to this Mcd several times being clos...	"McDonald's"	M4C 1H9

Figure 4.1: An example of a table structure of the reviews' data frame using Pandas. Only the first five and the last five rows of the data are visible.

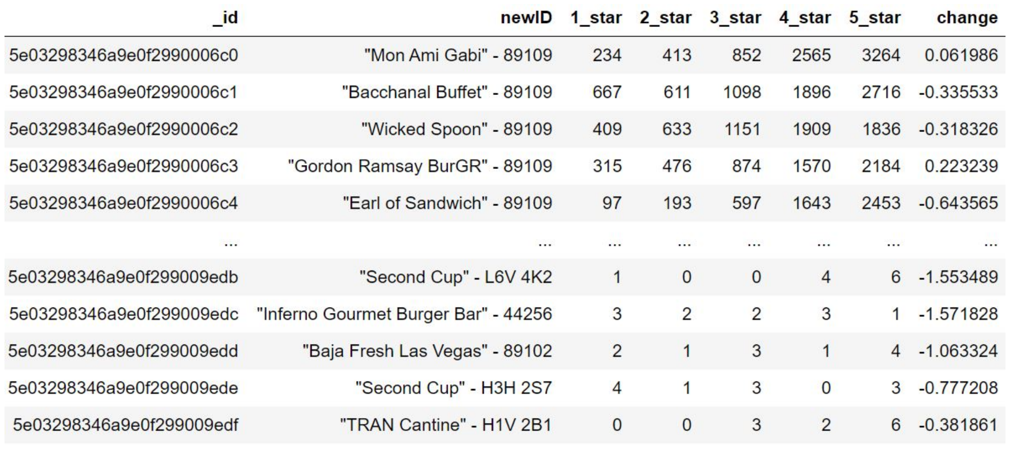
The next step was combining the first dataset with this one. Having one column in common, the 'business_id', the two tables were joined. The merge, however, was executed in a way that any reviews that were not referencing a place in the filtered businesses table were dropped.

The data underwent some more pre-processing before being loaded by the tool. Firstly, the restaurants, which had ten reviews or less, were removed since they were not considered to be of interest to a person using the tool. Each unique restaurant id was assigned a number representing how many times it exists in the database. All entries with values of ten or below were dropped from the data frame.

The last filter applied to the data, in order to remove as much noise as possible, was to try to filter textual reviews which are not in the English language. The python library Langdetect [Dan01] was used to determine the language of each of the textual reviews. A limitation is how accurately the tool is determining the language. If a review is not written in proper literal English, it contains slang, typos and does not conform to proper grammar it fails to be classified as language at all. There were 21 such occurrences. These reviews were kept in the database. 47 346 reviews were classified as not written in the English language and 47 325 of them were removed. We can say, that is not a large portion, considering the number of reviews left is 3 346 024.

Once this is done, we again divide our database into two separate datasets. One containing the restaurants and the other one containing the reviews. Both datasets were loaded into a NoSQL server database using MongoDB. MongoDB was chosen because it is built to load and read large amounts of data quickly. The newly created databases are loaded from the MongoDB to the Pandas data frames. This process takes approximately between 2 and 3 minutes but greatly reduces the downtime once the tool is started and requests are made from it in real-time.

The first newly created dataset contains basic information about the restaurants. It is a small table consisting of columns with information about the restaurants. The columns, as seen in Figure 4.2, start with a unique id value, the name of the restaurant, combined with its postal code, the number of reviews for each different star ratings, and the trend.



_id	newID	1_star	2_star	3_star	4_star	5_star	change
5e03298346a9e0f2990006c0	"Mon Ami Gabi" - 89109	234	413	852	2565	3264	0.061986
5e03298346a9e0f2990006c1	"Bacchanal Buffet" - 89109	667	611	1098	1896	2716	-0.335533
5e03298346a9e0f2990006c2	"Wicked Spoon" - 89109	409	633	1151	1909	1836	-0.318326
5e03298346a9e0f2990006c3	"Gordon Ramsay BurGR" - 89109	315	476	874	1570	2184	0.223239
5e03298346a9e0f2990006c4	"Earl of Sandwich" - 89109	97	193	597	1643	2453	-0.643565
...	...	...	...	...	...	...	...
5e03298346a9e0f299009edb	"Second Cup" - L6V 4K2	1	0	0	4	6	-1.553489
5e03298346a9e0f299009edc	"Inferno Gourmet Burger Bar" - 44256	3	2	2	3	1	-1.571828
5e03298346a9e0f299009edd	"Baja Fresh Las Vegas" - 89102	2	1	3	1	4	-1.063324
5e03298346a9e0f299009ede	"Second Cup" - H3H 2S7	4	1	3	0	3	-0.777208
5e03298346a9e0f299009edf	"TRAN Cantine" - H1V 2B1	0	0	3	2	6	-0.381861

Figure 4.2: An example of a table structure of the restaurants' data frame using Pandas. Only the first five and the last five rows of the data are visible. The trend column is named 'change'.

The trend column contains numerical values which might be positive or negative. These values are found by performing a simple linear regression on the different star ratings for a particular restaurant and the time of issue of these ratings [Kum01]. The values from the date of issue column and the star rating column are used as the two vectors for the regression. Since the date of issue column is in date format we convert all the values there into proleptic Gregorian ordinals using Pandas' to_datetime.toordinal(). The reason we do this is, because now, the values can be part of an arranged vector. Therefore, they can be fed to the next code sequence for proper calculations. As a result of our simple linear regression code, we get a regression line with coordinates. We subtract the starting coordinates of the line from the end coordinates. The result is our single number value. The time it takes for the trend to be calculated greatly depends on the number of reviews we are processing. In the end, the computed trend was for overall 38 894 restaurants, for all 3 346 024 reviews.

The second newly created dataset is much larger and contains information not only about the restaurant names and postal code but also about the reviews, ratings, and times. The dataset has 3 346 024 rows. Every row has a unique random id and is a single separate review. It also contains a business name column which is used to connect it to the other dataset. We add an additional column to the dataset. That column consists of the extracted lemmas as a list of string values.

For creating the lemmas column we must first extract all the separate words from the text reviews as tokens. For extracting the tokens, NLTK tokenizer is used [NLT01]. This method was chosen because it is performing faster compared to other python alternatives when it comes to tokenizing text. All the gathered tokens were converted to lowercase letters in order to be easily processed later on. Afterwards, some of the words are removed if they are not considered important. For this action, an NLTK English corpus of stopwords is used. This is a collection of common words in the English language where the words do not have inherent meaning in which people will be interested if encountered separately. Pandas 'apply' method is used to perform these actions on the data frame.

After the tokenization is finished, with the resulting list of words a lemmatization technique is initiated. For this part, SpaCy is used [Hon01]. The reason is, because of the lower computation time, compared to other tools. The list of tokens is fed into a SpaCy word dictionary and every word in that dictionary is stripped down to its lemma. This is done so that common words with the same lemma and meaning are easily detected by the top words counting and the TF-IDF algorithm applied later on.

The resulting list of lemmas is what is in the column of the dataset. The computation times of the tools used are analysed in Chapter 5 of this thesis. When starting the server, before starting the tool, the databases are converted into Pandas dataframes. This is done, in order to reduce the real-time delay users can experience when using the tool and requesting different data from the server. Only parts of the whole dataset are ever sent fro the server side to the tool at one particular time. Which parts depends on what data the tool requested from the server. This is done because of system and size restrictions employed by browsers. The requested data is converted into JSON files.The JSON format is used because it is light and can be easily sent and interpreted between the server-side back-end of the tool and the JavaScript front-end. More on how the data is later transformed and used can be found in Section 4.3.

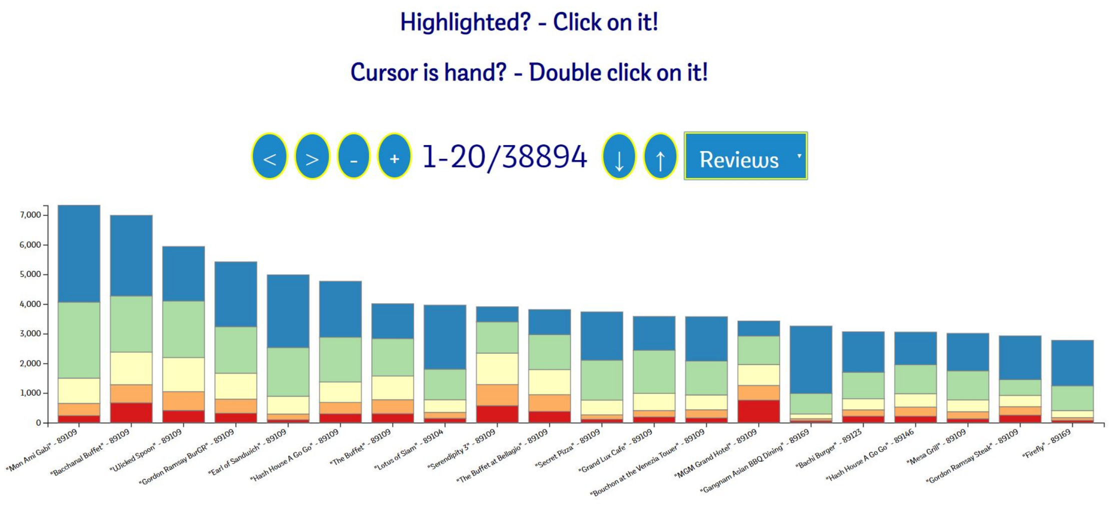
In order to initiate a real-time update of the database, a few things need to be considered. An update might be needed, since new reviews and restaurants data is constantly being added into Yelp and having the latest reviews for the relevant businesses is important for users of the tool. If we are adding new review entries to the review data set, the new entries have to be following the corresponding data conventions and characteristics. Additionally, the lemmas column has to be provided. As already mentioned in Section 3.1 of the thesis, this would require extracting a list of lemmas from the text reviews of the new data entries. Afterwards, just adding the new entries to MongoDB would be enough.

A more problematic database update would be dealing with the restaurants' dataset. Since we have added new reviews, the number of reviews, some restaurants have has changed. That change needs to be reflected in the restaurants' dataset. Simply incrementing the corresponding values from the table with the number of new review entries for each rating would suffice. Updating the values of the trend column would, however, require a more complicated approach. This is because new data entries could potentially change the direction of the trend.

4.2 Front-end

For the interface of the tool and the front-end part, multiple technologies are being used. The tool is leaning heavily on JavaScript. All the designs and graphics created are done with the Data-Driven Documents library, or D3.js for short. JavaScript is combined with HTML and CSS (.css) scripts. These technologies were chosen, because the original idea was to create a tool that is lightweight and can be used in a browser. The tool does not have to be platform specific but is connected to a server. The server-side should be able to execute quick calculations and queries with minimal delay and downtime.

The tool consists of three main pages. Additionally, there is an index page that has a simple button that initiates the tool. Upon clicking the Start button once, the user is transported to the first of the three main pages. Figure 4.3 presents how the first page looks like.


Highlighted? - Click on it!

Cursor is hand? - Double click on it!

<
>
-
+
1-20/38894
↓
↑
Reviews
7,000
6,000
5,000
4,000
3,000
2,000
1,000
0
"Mon Ami Gabi" - 89109
"Bacchanal Buffet" - 89109
"Wicked Spoon" - 89109
"Gordon Ramsay BurGR" - 89109
"Earl of Sandwich" - 89109
"Hash House A Go Go" - 89109
"The Buffet" - 89109
"Lotus of Siam" - 89104
"Serendipity 3" - 89109
"The Buffet at Bellagio" - 89109
"Secret Pizza" - 89109
"Grand Lux Cafe" - 89109
"Bouchon at the Venezia Tower" - 89109
"MGM Grand Hotel" - 89109
"Gangnam Asian BBQ Dining" - 89169
"Bachi Burger" - 89123
"Hash House A Go Go" - 89146
"Mesa Grill" - 89109
"Gordon Ramsay Steak" - 89109
"Firefly" - 89169

Figure 4.3: Review Watcher landing page. Currently showing the 20 most reviewed restaurants from the database of 38 894.

At the top of the screen, there are two messages shown. The first one reads 'Highlighted – Click on it!' Immediately below it, there is the second message that reads 'Cursor is hand? – Double click on it!' The two messages are put centred on the screen and are stylised using .css. The messages aim to be as instructions to the user, as to how to proceed with using the tool. It is not that common that internet sites have double-click options, and explaining to the user that this is one such site, was considered useful. Introducing visual cues to aid the users and show them what they can click, what they can double-click, and what is not click-able at all is considered intuitive and user-friendly. The visual cues are only present when the user hovers with the mouse cursor over an item. That item is highlighted and it surrounded with a black tick frame. Most of the background of the whole tool is white and the text is usually blue or black with a Plovdiv Sans theme font.

Centred below all the buttons there is a chart. This chart is built with the help of D3.js. This library has been chosen for because it provides vast functionality when it comes to building different kinds of interactive, lightweight graphs and charts. The chart is actually a stacked bar chart, such as the one described in Figure 3.1.

The x-axis contains the names and postal codes of 20 restaurants. The names of the restaurants are surrounded by double quotation marks and separated from the postal code with a dash. Originally there are just 20 restaurants visible on the graph. However, more can be added, and further manipulated using the top buttons. When hovering on the restaurant names with the mouse, the font colour changes from black to red and the mouse cursor turns into a pointer. This is to indicate to the user that these elements can be interacted with. They can be both clicked and double-click. The size of the graph is constant. It does not change depending on the dimensions of the monitor it is viewed from. It also does not change based on the number of restaurants shown or the number of reviews they have. This limitation, however, can be fixed in the future with slight adjustments to the JavaScript code of the page, in order to make the tool more suitable to different screens and devices.

Hovering on the bar charts triggers a JavaScript event function. When the mouse cursor is positioned on a particular bar of the chart, a tooltip appears. The tooltip is in the form of a white rectangle, like the one in Figure 4.4. When the user moves the mouse away from the rectangle into an empty space, the tooltip disappears.

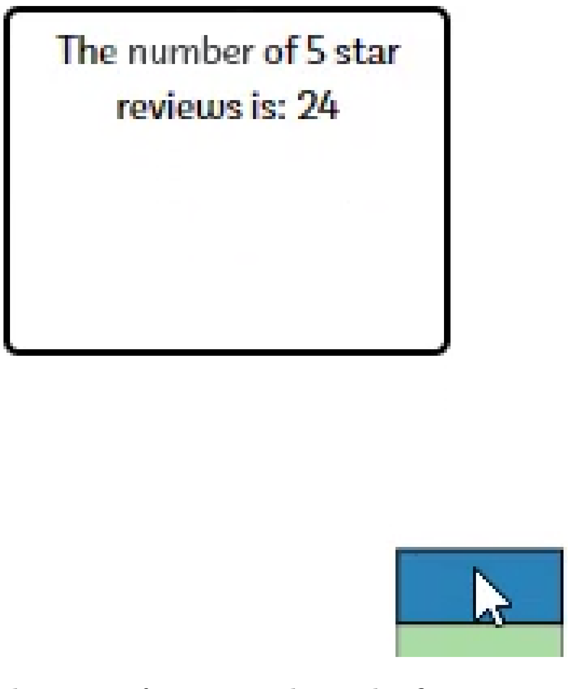

The number of 5 star reviews is: 24

Figure 4.4: Tooltip showing information about the five star reviews of the restaurant 'The Buffet at Excalibur'. The restaurant has 24 five star reviews.

This way of showing tooltips is also used in other parts of the tool. The D3 JavaScript code for the tooltips is similar and reused. The differences are the texts that the tooltip shows. In the case of a bar, a message reads that the current rectangle is for a specific star rating and that the number of reviews for that rating are as follows. Hovering on the bars also highlights them by making their border twice as tick. An example of that is depicted in Figure 4.5.

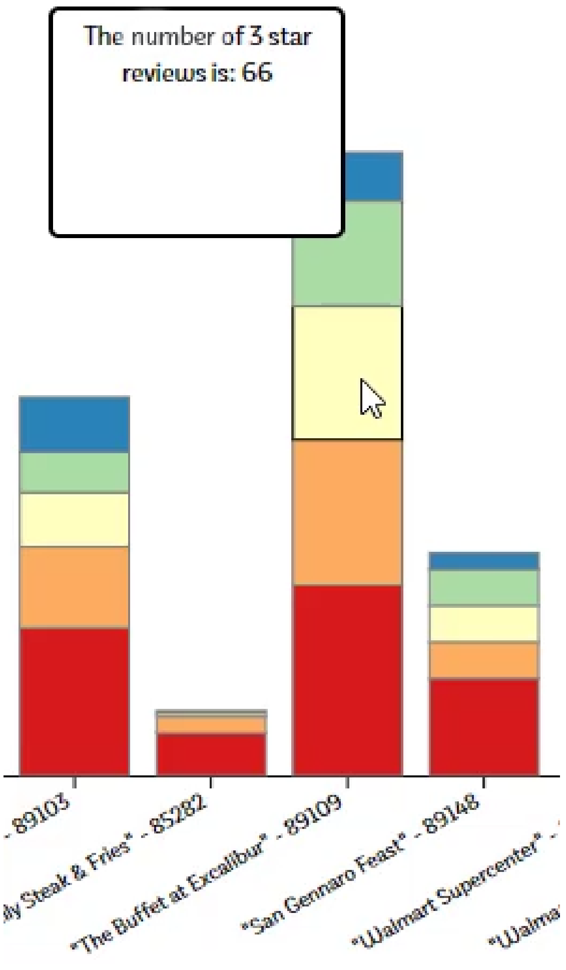


Figure 4.5: The Figure shows the effect of hovering with the mouse cursor on a bar from the chart.

Moving the mouse away from the bars returns them to their original state. This is another indication that the area can be clicked on.

Clicking on a bar will start a function to retrieve common words for the specific restaurant and the specific rating. After a small delay, a tooltip will appear again with a message written, containing the top ten most common words, starting with the most common and continuing in descending order of commonality. Performing this action could take between half a second and two seconds, depending on the number of reviews involved in the search. More information on the loading times is available in Section 5.1. The delay time results, because when the button is clicked an Ajax [Gar01] post request is sent to the server to retrieve the list of words that are to be shown in the tooltip. Clicking on a graph or a chart evokes a JavaScript asynchronous function. Since D3 is asynchronous by default, clicking on the bar will just display the tooltip with no text. That is because the Ajax request that is sent to the server has just been posted. In order to make the graph to display the tooltip after a response from the server has been received, an await clause needs to be put on the post call. When clicking the mouse, cursor will change into a time-watch and after the server returns the call, only then the tooltip will show up with the appropriate information. Moving the mouse away from the clicked element will again hide the tooltip.

This approach for showing information has its limitations, however. One aspect that users experienced problems with during the evaluation (see Chapter 5), for example, is that they often move the mouse, before being able to read through the tooltip. Keeping the mouse steady is what ensures that the tooltip stays visible. Some of the test users also expressed their desire to be able to view information about the common words just by hovering, without needing to click. This could have been problematic from a technical standpoint. Since there is a certain downtime and delay when requesting to see the top words, hovering impatiently over multiple bars would lead to sending multiple requests to the server. Since the tool is built to deal with each request one by one, that would lead to unnecessary pipelining, which might, in turn, result in making the tool unusable That is why all the functions, which require post requests are restricted to mouse clicking or double-clicking actions.

The graph is able to quickly view the restaurants' data without noticeable delays because that same restaurants' data is loaded into the JavaScript object variable when the tool is first started. All the information, from the previously described and shown in Figure 4.2 in the table as dataset 1, is passed to the JavaScript side of the tool from the Python Flask server using the JSON file format. This was the only dataset that was able to be loaded into the browser and used in full by D3. This is, as explained before, because of its small size. And while the whole data is loaded into the tool at once, not all of the data is shown at once. As explained in Sub-Section 3.2.1 we only display ten restaurants at a time in the overview

Starting left to right, we have four oval buttons, as the ones in Figure 4.6.

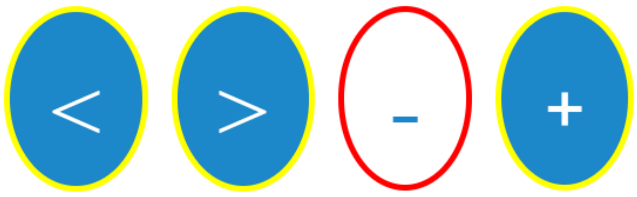


Figure 4.6: The first four buttons seen when starting Review Watcher. When hovering the mouse cursor on a button, it will change its colours. Its colours will become like the colours of the '-' button.

Hovering over the buttons changes their colour. An indication that they can be clicked once. All the buttons here control the graph that is directly below them. Since the graph represents only a set of the restaurants at a time, the first two buttons on the left change the set of restaurants shown. When the site is open for the first time, the data is sliced into sets of restaurants. The first ten restaurants are banded into a set and used for the bar chart. When the '>' button is clicked, another data slice is created which contains the next set of restaurants. Immediately afterwards, the graph is erased and redrawn in the same spot with the new data set in mind. If the < button is pressed a previous data set of restaurants is taken and displayed. These two buttons can be used like page-turners in order to manually go through all the restaurants in the database. The number of restaurant sets is always the upper-bound rounded integer number of all restaurants divided by the number of desired restaurants per page. If, for example, there are 38 894 restaurants in the database and the user desires to view only 20 restaurants per page, then the number of possible pages or slices of the whole dataset equals to 1 945.

If the user wants to change the number of restaurants shown at a time, it can be done with the next two oval buttons. The button with the label '-' will decrease the current number of restaurants per page by one. The '+' button, right next to it, will increase that number by one. The number of restaurants shown per page cannot get lower than two. As expected, increasing the number of restaurants will decrease the number of pages and potential slices of the data object and vice versa. Clicking on these buttons will make the bar chart to be redrawn with the new values set.

On the right to these four buttons, there are three numbers. The first number shows the placing of the first shown restaurant in the chart. The number is position of that restaurant in the sorted list restaurants in the database. After a dash is the number of the last shown restaurant from the dataset. If the '>' button is pressed, the next set of restaurants will be shown. These numbers are the positions of items in the whole, sorted data set. After the second number, there is a slash and then we have the overall number of restaurants in the whole database. This number is constant and can only be changed if the database is changed externally.

After these numbers, we have two arrow buttons and a drop-down button. The up and down arrow buttons are used to sort the whole dataset in an ascending and descending order, respectively, by certain criteria. That criteria can be chosen from the drop-down menu button. When the arrow buttons are pressed, the graph will be redrawn with the list of restaurants being sorted using a certain sorting algorithm and direction of sorting. The number of restaurants shown per page will remain as is. The page number will also remain the same. However, the restaurants shown might be different, because the new order of the large dataset changed the positions of the restaurants on the list. The drop-down button options decide which sorting algorithm will be used. The six sorting algorithms available are named in the tool Reviews, Mean, Weight, Variance, Trend and Name.

The Reviews option on the drop-down will arrange the restaurants based on the number of reviews they have. For this job, the JavaScript code must simply count all the reviews each restaurant has. Adding all the values from the 5-star rating columns for each restaurant together will provide that answer.

The Mean will arrange the list by the average star rating of the restaurant. The mean rating of a restaurant can be computed from the score frequencies as follows:

$$m = \frac{n_1 + n_2 \cdot 2 + n_3 \cdot 3 + n_4 \cdot 4 + n_5 \cdot 5}{N}$$

where n_1 is the number of one-star reviews, n_2 is the number of two-star reviews, n_3 is the number of three-star reviews, n_4 is the number of four-star reviews, n_5 is the number of five-star reviews, N is the number of all reviews and m is the mean.

The next drop-down option is the Weight. It is the mean, weighted by the overall number of reviews for a restaurant. This is calculated using a Bayesian estimate formula. The formula is sensitive to two additional parameters. The first one is the minimum number of observations required for a weighting scheme to be used. This number strongly depends on the data we are working with. However, for this version of the tool, this number is hardcoded to be equal to 40. It was decided that this is a number that brings the best results using the test data, in the sense that it first sorts restaurants that have a balance between having lots of reviews and overall high score. The other parameter is the average rating we can expect for a restaurant. This parameter is, again, hardcoded and is equal to 3.5. This is because, after working with the data it was discovered that users tend to give an average score for all restaurants to around 3.5. These numbers might need to be changed if the database is changed or updated. Changing them would simply require altering the two values in the JavaScript code. Having this all in mind, we compute the Weight of a single restaurant as follows:

$$w = \frac{40 \cdot 3.5 + n_1 + n_2 \cdot 2 + n_3 \cdot 3 + n_4 \cdot 4 + n_5 \cdot 5}{40 + N}$$

where w is the weight.

The next drop-down option is named Variance. We perform in JavaScript a calculation of the variance of one restaurant's review scores from that restaurant's mean score.

Using that sorting will bring restaurants with a high variance to to the beginning of the arranged set and restaurants with zero variance to the end of the set. Restaurants with zero variance will be restaurants that have only reviews in one star-category. Having a lot of 1-star reviews and a lot of 5-star reviews would mean that the restaurant has high variance.

The next sorting option is called Trend. It simply sorts the restaurants by their trend parameter. No additional calculations are performed in real-time for this sorting.

The last sorting option is Name. It arranges the restaurants in alphabetical order. This sorting, again, does not require additional calculations. When the user changes the values of the drop-down button, the currently shown restaurants are immediately arranged accordingly. Note that only the current slice is sorted in that case. In order to sort the whole dataset, the up-arrow and down-arrow buttons must be used.

Another way to interact with the graph, as already mentioned in Section 3.2 and shown in Figure 3.5, is by clicking and double-clicking on the x-axis values. Clicking just once on a restaurant name will send a request for the most common words used in the reviews for that restaurant. After the short expected delay, the tooltip will show the ten top words. This is similar to clicking on the bars on the chart. The difference is, this time we get the top words for the whole restaurant and not just for some specific star ratings for the restaurant.

Double-clicking on a restaurant, however, sends a different request and leads us to the second page of the tool. A delay algorithm is used for the double-clicking functionality. Since the regular double-click function in D3 does perform the action that is set to the function, it also executes the single-click function twice. This is unnecessary in this case and could only delay the execution of our request by pipelining unwanted single-click requests. To fix this issue, a separate function is written for some click functions. This function introduces timed-event handler that watches whether a user clicks once or twice in the span of a few milliseconds. This function makes sure the user only clicked one and performs the single-click function as normal. If there was a second click, however, only the double-click action is executed. Thus only the appropriate double-click request is sent. This post request to Flask redirects us to the second page of our tool.

The second page of our tool is named after the restaurant that the user double-clicked and its postal code. The post request returns two JSON arguments that will be used for the two new graphs that will be revealed to the user. The first argument is a table containing all star ratings for the particular restaurant, accompanied with dates of issue. This argument will be used for the temporal heatmap (see Figure 3.11). The second argument is a table with the 20 most common words from the reviews. The table has columns for all the reviews and for the different years for which there is data. For each word, there is also a separate column with how many times the word appeared. The information from this argument is used for the word cloud. Receiving a post answer from the server with this information takes between one and two seconds, depending on the amount of data being requested. More information on the times of post requests is available in the next chapter.

After the second page of the tool is completely loaded, it will look like the one in Figure 4.7.

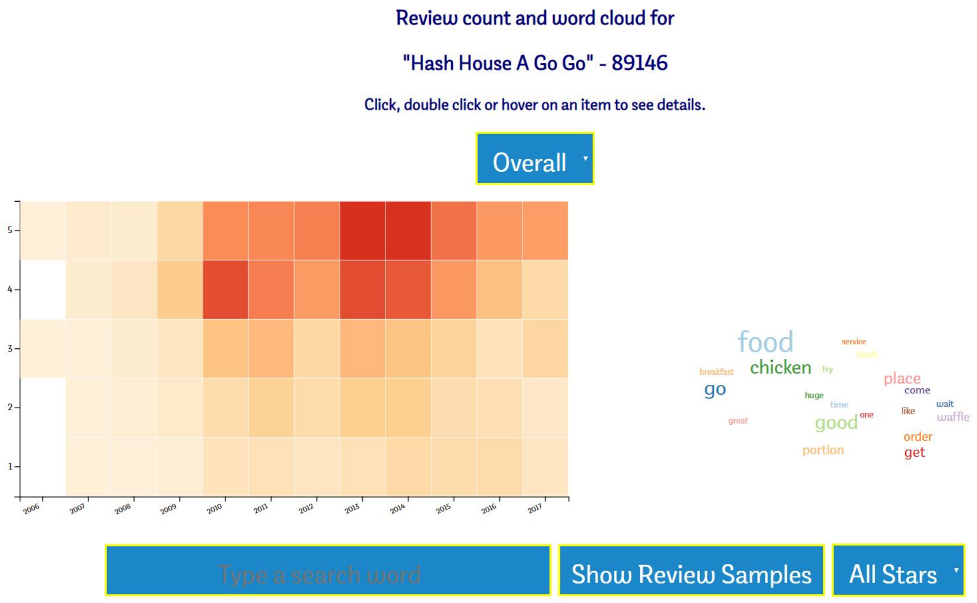


Figure 4.7: Review Watcher second page. It shows data about restaurant 'Hash House A Go Golue'.

The word cloud seen on the right side in Figure 4.7 is built using D3.js and its word-cloud building tools. The size dimensions of the word cloud are always standard and cannot be changed or influenced in any way. The word cloud can contain up to 20 words written in different random colours. The colour scheme is taken from ColorBrewer 2.0 [BH01].

When the page is first loaded the Overall words are chosen. Meaning that the top words from all the reviews for the restaurant are shown. In the case that the drop-down button option is changed the whole word cloud is erased and redrawn with new data depending on what is the new value that was selected. The option values of the drop-down button are different depending on the restaurant that is selected. The option values are loaded from the columns of the table. The first, default option is always called Overall. Afterwards, it strongly depends on the dates of issue of the reviews for the restaurant. The years, in which the restaurant was operational, are listed as options. If, for example, the restaurant was working and receiving reviews for five years between the years 2006 and 2011, the listed options in the drop down would be: Overall, 2006, 2007, 2008, 2009, 2010, and 2011. If a specific year is selected the most common words among the reviews only from that year will be shown in the word cloud.

As already mentioned, the table with common words also contains the number of times a word appeared in the respective year, or in all the years, if that column is selected. This information is used by the word cloud to determine the font size of each word. Each word has a maximum font size of 35 points and a minimum size of 10 points. The size of each separate word (S) is calculated using the formula, where N_w is the number of times the word appears in the reviews, A is the times the least common word appears in the list and B is the times the most common word appears in the list:

$$S = \frac{(N_w - A) \cdot (35 - 10)}{B - A} + 10$$

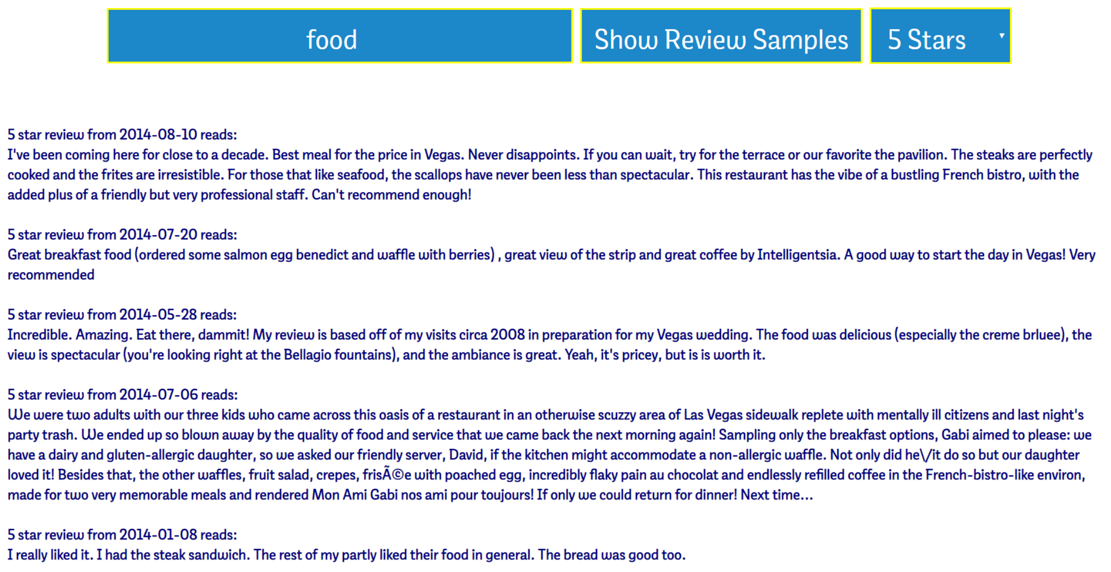
All the words are randomly spread along the word cloud drawing area without overlapping with each other.

Upon clicking the button 'Show Review Samples' another Ajax post request will be sent to the server to retrieve actual text reviews written about this restaurant. The final drop-down button on the page has a default value labelled 'All Stars'. This button is used to filter reviews based on their star score. The options to choose from are respective to all the possible five score ratings. The options are All Stars, 1 Star, 2 Star, 3 Star, 4 Star and 5 Star.

The drop-down button at the top of the page is also connected with the 'Show Review Samples' button. If it is set to a particular year, the search request will only be about that particular year. For example, if the selected value on the drop-down button is 2010 and the Show button is pressed the results will be only reviews of the year 2010.

As seen in Figure 4.7, below the temporal heatmap, on the left side of the Show Review Samples button there is a text field. When the Show Review Samples button is pressed the search words written in the text field are passed as a string into the post request parameters. As a result, only reviews that contain in them that exact same string will be returned as results from the search. The search string function is case sensitive. If the search field is left empty the search results will not be filtered by any string, but only by the attributes from the two drop-down buttons.

When the review sample post request is sent to the server, after some time an answer is received. If the result from the query is five reviews or less all the reviews are sent as a result. However, if the query results are more than five, five randomly chosen reviews are selected and sent to JavaScript as an answer. We are not showing all the available reviews as that might clutter the page with too many reviews. The resulting reviews are being listed on the page in random order. An example query result is shown in Figure 4.8.



5 star review from 2014-08-10 reads:
I've been coming here for close to a decade. Best meal for the price in Vegas. Never disappoints. If you can wait, try for the terrace or our favorite the pavilion. The steaks are perfectly cooked and the frites are irresistible. For those that like seafood, the scallops have never been less than spectacular. This restaurant has the vibe of a bustling French bistro, with the added plus of a friendly but very professional staff. Can't recommend enough!

5 star review from 2014-07-20 reads:
Great breakfast food (ordered some salmon egg benedict and waffle with berries) , great view of the strip and great coffee by Intelligentsia. A good way to start the day in Vegas! Very recommended

5 star review from 2014-05-28 reads:
Incredible. Amazing. Eat there, dammit! My review is based off of my visits circa 2008 in preparation for my Vegas wedding. The food was delicious (especially the creme brluee), the view is spectacular (you're looking right at the Bellagio fountains), and the ambiance is great. Yeah, it's pricey, but is is worth it.

5 star review from 2014-07-06 reads:
We were two adults with our three kids who came across this oasis of a restaurant in an otherwise scuzzy area of Las Vegas sidewalk replete with mentally ill citizens and last night's party trash. We ended up so blown away by the quality of food and service that we came back the next morning again! Sampling only the breakfast options, Gabi aimed to please: we have a dairy and gluten-allergic daughter, so we asked our friendly server, David, if the kitchen might accommodate a non-allergic waffle. Not only did he\/it do so but our daughter loved it! Besides that, the other waffles, fruit salad, crepes, frisÃ©e with poached egg, incredibly flaky pain au chocolat and endlessly refilled coffee in the French-bistro-like environ, made for two very memorable meals and rendered Mon Ami Gabi nos ami pour toujours! If only we could return for dinner! Next time...

5 star review from 2014-01-08 reads:
I really liked it. I had the steak sandwich. The rest of my partly liked their food in general. The bread was good too.

Figure 4.8: The result from querying five reviews for restaurant 'Mon Ami Gabi'. All five reviews shown are from they year 2014, are five star reviews, and contain the word 'food'.

The first line of every review entry is always in the same format. It contains information about the star the reviewer gave and the date of issue. Afterwards, the review itself starts. Only five reviews are shown per page because it was considered too overwhelming to show more on a page. As not to always show the same set of reviews for search criteria the results are intentionally randomised. A random number generator is used to shuffle the rows of the data frame containing the reviews. This operation is performed on the Python side of the tool. When the JavaScript code gets an answer from the post request, it loops through that JSON and fills the pre-defined invisible text box with the respective information.

The last part of this page is the temporal heatmap on the left-hand side. It represents a matrix of the reviews over time, similar to the one on Figure 3.11. D3.js standard heatmap functionalities are used for the creation of this matrix graph. Its linear colouring scale function is employed for the colours of the cells.

Hovering over a cell reveals a tooltip that informs the user of exactly how many reviews there are in the selected cell. Hovering over cells also highlights the cell and changes the mouse cursor into a pointer. Thus once again indicates that single-click and double-click options are available for this element. An example of this can also be seen in Figur 4.9. Clicking on a cell once will bring up a tooltip with the ten most common words among the reviews for the particular cell (see Figure 3.12).

Double-clicking on a cell gives us the ten TF-IDF lemmas with the highest rating. An example is shown in Figure 3.13. Retrieving of TF-IDF words is performed again via post request to the Flask server. While the request is being processed, the mouse cursor is changed into a loading sign. Retrieving TF-IDF lemmas, might take around half a second longer than just requesting the top words. This is because the calculation algorithm for this task is slower. More information on that is available in Section 4.3 and Section 5.1.

If a user hovers over the values on the y-axis the numbers turn red. Another indication that the values can be clicked. Clicking will give us a tooltip with the TF-IDF words for the particular star score. In this case, five collections of reviews, each containing only reviews with the same ratings, are created. That means there can be a maximum of four collections to compare to the main collection of the chosen score. Hovering over the x-axis values of the matrix once again turns their colour red. Clicking on a year will again send a request for a TF-IDF of the words for the specific year. This time all reviews for one year are are formed into a single collection.

Hovering on a year, however, will also turn the mouse cursor into a pointer. Double-clicking on a specific year will transport us to the third and final page of the tool. This page is almost identical to the restaurant page. The only difference is that all the information on the screen will be regarding the same restaurant, but exclusively for the year selected. An example for such a page is seen in Figure 4.9.

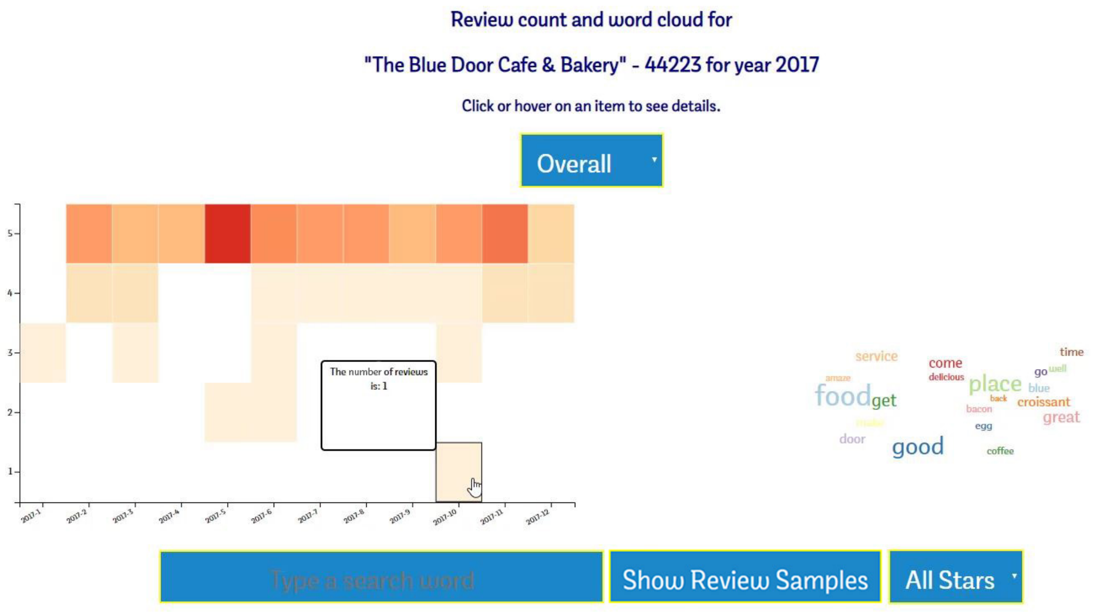

Review count and word cloud for
"The Blue Door Cafe & Bakery" - 44223 for year 2017
Click or hover on an item to see details.
Overall
The number of reviews is: 1
Type a search word
Show Review Samples
All Stars
service
come
delicious
place
time
go
well
food
get
blue
back
croissant
bacon
great
egg
door
good
coffee

Figure 4.9: Review Watcher third page. It shows data about restaurant 'The Blue Door Cafe & Bakery' for the year 2017.

The first drop-down button, that previously had different years as options, now has only the current year, but with different months. The word cloud is only showing words from the chosen year and changing the option values of the drop-down button will switch the word cloud to be restricted to words from the selected month. Search queries are only within the chosen year. Restricting the queries to a specific month of the year, is again possible from the drop-down button on the top of the screen.

The data on the matrix graph is only for the selected year, where each value is a different month. All the functionality is structured the same way as in the previous page. JavaScript and D3 codes are reused for these parts. The x-axis cannot be double-clicked and the mouse does not change into a pointer upon hovering over a value. It was considered that introducing a consecutive page with week-by-week or day-by-day information about reviews will be too specific and the information will be too sparse to be of use. However, this option can easily be added in the future, if deemed necessary.

The tool was developed and tested on Google Chrome 80.0.3987.163. For properly using the tool, the browser needs to have cross-origin resource sharing (CORS) enabled. This function is necessary so that the tool can send and retrieve information and data with Flask and the different parts of the tool. For this purpose a plugin to the Chrome browser called Allow CORS: Access-Control-Allow-origin was installed and enabled [She01].

4.3 Back-end

The JavaScript part of the tool often sends Ajax post requests to the back-end server. Without this communication, the tool will not be operational, since the server performs some of the major operations with the database. Most of the operations such as data manipulation and queering are not able to be performed on the JavaScript side of the tool, because of data size restrictions, time-consuming processing, and generally a lack of proper libraries for natural language processing. Therefore, the tool is linked to a Flask server, which is able to take care of the different requests issued by the client-side of the tool. All the following operations are performed through Python version 3.6.

When the server is originally started, the data is copied from the MongoDB storage. The restaurants' dataset is immediately sent via post request to the tool. It is sent as a JSON and is from then on stored and used by the JavaScript side of the tool. The reviews' dataset in the MongoDB is split into two similar Pandas data frames. This is done because queering in Pandas is faster than doing it with MongoDB requests, as we show in Section 5.1. One of the datasets contains the actual textual reviews. The other dataset contains just the list of lemmas. This approach makes sending and receiving post requests from the tool between half a second and two seconds faster. The downside to this approach is that the tool uses a lot of resources because it saves and interacts with a lot of data by storing it in the operational memory. This issue was disregarded as unimportant since the main load will be taken by the server and not by the user's machine.

Standard Python libraries are used in order to access and work with the data. The data is being always stored into Pandas data frames allowing for easy data filtering, queering, and other ways of data manipulation. Python is using the Flask libraries to host the different pages of the Review Watcher and its resources. Flask provides the ability to build multiple endpoints for application program interface, which when called will provide different kinds of data. A few of these endpoints are set to have post and get methods. Their functions are to find out what kind of information the user is interested in and return it as a JSON file or as a string, depending on the necessity. Post request with arguments is sent to the endpoints from the JavaScript side.

The main Python script references multiple other scripts, which perform different functions. One of the often-used scripts is used for filtering the dataset. The script has a main function that accepts as arguments a Pandas data frame, name of a restaurant, rating of a review, period starting date, and period ending date. Not all of the arguments need to be provided to the script to return an answer. Only the data frame argument is mandatory. If for example a request comes from the site, that data for a specific restaurant is needed, Flask's get method receives the name and postal code of the restaurant. That value is passed as an argument to the filter script along with the full database. Which database, i.e. the text reviews or the lemmas, is passed as an argument, depends on the type of the request and the endpoint to which it is sent. The filter script then produces a dataset where only rows, with matching restaurant name and postal code to the ones provided as an argument, are present. All the arguments passed in the post and get requests are of type string. So in the case of rating filtering, the string values need to be transformed into integer type.

The last two filter arguments are two dates. Those are the earliest date and the latest date from which the reviews should be shown. After being passed as strings in the form of YYYY-MM-DD, the values are transformed into DateTime type. The filter function then returns reviews that were issued between these dates and on these dates. This is used, if restaurants are examined on an annual or monthly basis.

The Flask server has an endpoint, which returns a string of the most common words for a set of reviews. When a request to this endpoint is sent, a few arguments are passed on in order to filter the reviews of interest. This is done via the filter script explained in the previous paragraphs. From the resulting data frame, only the column with the lists of lemmas is passed as a series to a separate script. This script counts how many times a lemma appears in the list and returns a string of the ten most common words. For this task, the Python script uses Itertools [Agg01] to iterate all the lists in the series and then feeds them to a Counter [Fou01]. The series value types is already a list of strings by default, therefore no additional transformations are needed. The resulting string is returned as an answer to the post request.

If a request for the TF-IDF words is sent, it is sent to a different endpoint, compared to the most common words endpoint. To this endpoint, we have an extra argument passed, which is called granularity. It is a mandatory argument that needs to be filled with a string for the request to be properly processed. The value of this argument depends on what kind of calculation we are making and what we need to compare the data to. Since the TF-IDF algorithm can compare review collections to different types of collections, this argument makes sure, the proper comparison is made. This argument is referred to as a granularity argument. The five acceptable values for this argument are: Year, Month, Rating, YearPlus and MonthPlus.

The Year granularity argument indicates that all reviews for a restaurant will be split into separate collections of reviews, where one collection is only consisting of reviews from one given year. The Month granularity argument means we will be concerned with a collections separated based on months. The Rating means we are looking at a collections of reviews with the same star rating as a single collection. The YearPlus means we are looking at collections, which are for a particular year and a particular star rating. And lastly, the MonthPlus is about collections for one particular month and one particular rating.

Here we are using the Filter script twice in a row. At first, we are requesting to filter the data by restaurant name. We might also filter by year, depending on the granularity argument we use (month or year). We save the result as a separate variable and we filter again, on top of it, based on all the posted arguments. In that case, if, for example, we have an argument set to Year, we get a data frame with all the reviews for a restaurant and another data frame only with the reviews of that restaurant for a particular year. In the case of Month, we get a data frame with reviews for one restaurant for one specific full year and a separate data frame with reviews for that same restaurant and that same year, but just for a single specific month. We do this because the second dataset is the one we are interested in and the first dataset is the one that will be compared to when performing the TF-IDF scoring. Afterwards, we send the data to another script that performs the TF-IDF calculations.

The TF-IDF script is custom made and incorporates standard Python libraries with Pandas, Counter and iteration tools. The main function of the script accepts as arguments two data frames and a string parameter. The string parameter is the type of granularity. The types are the same as the ones from the granularity argument from the endpoint. First, the script counts how many times a word appears in the lemmas column of the second data frame. This calculation part is the TF. A Counter type is returned as a result.

Then, for the calculation of the IDF, we have two parts. For the first part, we take the first data frame and drop the second one from it. Then we create an additional column to the resulting data frame, based on what the granularity argument was. Then we strip down the other columns, so that we have a data frame with just two columns: the lemmas and the newly formed granularity column. We use the 'group by' function to sum the lists of lemmas, who have the same values in the new column together. Now we have a properly structured data frame with a new column with unique values and a lemmas column, with all the lemmas for each respective group. An example of how the data frame would look like is shown in Table 4.1.

Table 4.1: Example of how a structured data frame with different collection of lemmas would look like. In the example the granularity argument is Year. Therefore the values in the id column are the different years for which there are restaurant reviews. If we want to compare the year 2016 to the rest of the reviews, we will not have the year 2016 as an id value. Each value in the lemmas column is a list of all the lemmas from all the reviews for the respective year. In the table, only the first two and the last two lemmas from each list are visible.

id	**lemmas**
2014	[list, lemma,..., here, show]
2015	[word, example,..., lemma, nice]
2017	[good, food,..., pizza, eat]

Lastly, we convert the lists in the lemmas columns into a type set. This will remove duplicate entries in the list. We are not interested in duplicates.

Now we are ready to compute the IDF and the TF-IDF. We cycle over each word in the previously defined TF Counter and we calculate its IDF score of each word there. We loop over the different sets and count, in how many of them does the word, we are calculate the IDF of, appears. Then we calculate the IDF getting the logarithm of the overall number of granularity based collections + 1 (for the example from Table 4.1 that would be three, for each unique id, + 1), divided by each word's appearances in these collections. The TF-IDF is just the multiplication of the TF score and the IDF score of each word.

The word cloud portion of the tool also requires a separate endpoint from Flask. This endpoint is the same for all types of word clouds. As post parameters, the word cloud endpoint requires the name of a restaurant, which is again combined with respective postal code. It also requires a granularity string parameter. The possible values for the granularity parameter can only be Year or Month. This depends on whether the user is sending the request by opening the second Review Watcher page or the third Review Watcher page, respectively. The last two arguments are the starting date and the ending date of the review issue period. These fields can be left empty if the general overview of a restaurant has been opened. Post requests to this endpoint are evoked only on opening the web page.

If a word cloud request is received in Python, first the filter script is evoked, to select only the needed restaurant review lemmas, for the needed time. This data, along with the granularity parameter, is passed onto a separate script. That script utilises iteration tools with Counter to find common words among different periods. Whether the periods of interest are months or years is decided by the granularity parameter. In the end, the script produces a data frame. The data frame is a table with multiple columns of two main types. Every odd-numbered column being of type string and containing one word each. Every even-numbered column is of type integer and specifies how many times its corresponding word to the left appeared in a specific time frame. The first two columns are always named the same: ’Overall:word’ and ’Overall: time’. They show the most common words in the whole data and the number of word occurrences. Every other couple of columns are for a specific period, either a year or a month. This table is then transferred into JSON format and returned to JavaScript, where it is translated and used for the word cloud.

The word cloud endpoint also sends another JSON file. It is the transformed result from the filter script. It is used for constructing the temporal heatmap.

The last endpoint that is used is for showing the reviews. This one is evoked if a user clicks on the button Show Review Samples in the tool. The get and post methods here expect arguments such as the name of a restaurant (and postal code, as usual), star rating, starting date, ending date, and a search string. Only the restaurant name (and postal code) argument is essential for the successful function of this endpoint. The arguments that are present after a post request is sent, are forwarded to the filter script. The filter script produces a data frame with the reviews that fulfil the given criteria. If there are reviews and the search string parameter is not empty, all the reviews are searched for containing the string. If there are five or less such reviews they are converted into a JSON format with their respective star rating, date of issue, and corresponding text. The JSON is returned to JavaScript to be read and transformed into the page. In the case that there are more than five reviews fulfilling all the filter criteria and the search word check, a NumPy randomiser selects five random rows from the data frame. These five results are subsequently passed onto the tool.

CHAPTER 5

Results

This chapter of the thesis will be about what was the end product of the work. Initially, there is a description of some technical performance benchmarks and how they were fulfilled by the created tool. They will also serve as a justification for making some design and implementation decisions. Then follows a section about how the tool was tested with users and compared to other tools in order to decide its usability. The results from the created user study are analysed and discussed in the last parts of this chapter. They will serve us as main arguments to answering the already established research questions and proving, or disproving the hypotheses.

5.1 Performance Benchmarks

During the development of the tool, there were a few established criteria to determine how the tool should function, as already mentioned in Chapter 1. One criterion was that the tool should be lightweight and possibly used from a browser. This, of course, justified the use of implementation tools such as D3.js and Flask. Another crucial criterion was that the tool's separate functions are executed fast. There should be no unnecessary delays that might make the tool seem cumbersome and create loading times that will drive any potential users away. Lastly, an optional goal of the thesis was to create a tool that is capable of loading large amounts of big data. This part of the thesis will be comparing different implementation decisions and giving justifications to why one approach was chosen over another one.

All the development, testing, and user study of the tool were executed on the same machine: a laptop with 32 gigabytes of random-access memory (RAM) and a solid-state drive. The central processor is an Intel Core(TM) i7-6700HQ CPU 2.60GHz. The laptop is running the 64-bit Windows 10 operating system.

Using JavaScript and D3 did help Review Watcher to quickly execute commands and draw graphs due to JavaScript and D3's asynchronous capabilities. Performing almost all JavaScript-based commands was instantaneous. However, one of the first issues that was present was the inability to load big data into JavaScript at once. Depending on the browser and the machine it runs on, not more than a few hundred megabytes of data can be cached by an internet browser at a time [Sou01]. This, of course, could have been problematic, considering we are expecting to deal with between three and four gigabytes of data. This naturally led to the decision for the data to be formatted in a different manner, split into parts, and heavily pre-processed before being supplied to the tool. How the data was changed, has already been elaborated in Chapter 4. As a result, MongoDB was chosen for storing the data, mainly because of its NoSQL capabilities. Fetching the data from the database, however, also proved a challenge. When dealing with a dataset with more than three million rows and columns containing sometimes full pages of text and a column with list values some downtimes are inevitable. And since constantly sending queries to MongoDB was taking too long (see Table 5.1 for reference), a different approach was taken.

Table 5.1: Average time in seconds it takes to query a table with four columns and 3 346 024 rows using Pandas and MongoDB. The queried dataset is the same dataset as the one shown in Figure 4.2, except the index, the business_id, and the postal_code columns. 50 separate runs of the code were performed with different number of search parameters.

Method	**Number of parameters**	**Time in seconds**
Pandas	1	0.185
Pandas	2	0.183
Pandas	3	0.18
MongoDB	1	1.84
MongoDB	2	1.81
MongoDB	3	1.73

The whole data is dumped into Pandas and queries are only made to the Pandas data frames. The time to fetch the data and convert it to a proper data frame, however, takes between 3 and 3.5 minutes, as seen in Table 5.2. This time does, however, count towards tool loading time, since it is done before the server is started. One downside to this approach is that the server side uses up to 18 gigabytes of RAM to complete this operation. Later, queries sent to Pandas take milliseconds to complete as shown in Table 5.1.

Table 5.2: Loading times for each separate operation on server start-up in minutes.

Operation	**Time in minutes**
Fetch and convert dataset with lemmas to Pandas	1.7
Fetch and convert dataset with restaurants to JSON	0.01
Fetch and convert dataset with reviews to Pandas	1.4
Overall loading time	3.11

If a result is present, it is stored into a JSON format and sent to the tool's page. This process is executed by Ajax post request and Flask endpoints. The time it takes to send and receive an answer from such a request depends also on what Python does with the request. In the case of Review Watcher, it was recorded that it could take between one second and three seconds. This is considered a passable delay. Around one second is spent even for performing the simplest request. A three-second delay was most commonly encountered if a TF-IDF script was run on collections of 5 000 reviews or more. When the tool was tested with a small database of just 30 restaurants, the request time was always close to one second or one second and a half. This illustrates that the amount of data explored is the main cause of downtime.

All these tests, however, were performed if some of the information has been calculated in advance. Calculating the Trend of restaurant reviews, for example, is one such information. Calculating 38 894 different trends from a collection of 3 346 024 reviews were done with a simple linear regression method. Whilst a simple method, the large dataset led to the whole database being processed in 2 hours and 52 minutes. Performing such calculations in real-time when using the tool is inacceptable. For this reason, the calculations were performed only once, before the database was loaded into the tool. Adding additional data to the set in the future might be problematic, so ways to re-calculate the trend accordingly should be sought. Preparing the data before using the tool is necessary, because performing trend calculation in real-time will be too slow to be comfortable. Alternatively, the trend parameter can be ignored entirely if constantly changing its value after every update is not a viable option in regards to its benefits to the user.

Another problem with computational time saving was the calculation of the common words. The initial idea for the tool was that it is able to dynamically find the most common word tokens in a text. The initial test runs showed that depending on the size of the text, tokenization could take minutes. Incorporating this with lemmatization and finding common phrases or sets of words, would increase the downtime of the application to tens of minutes. This kind of delay is considered unacceptable for a tool that is supposed to provide data quickly. Libraries such as NLTK and SpaCy were used for lematization, until it was revealed that incorporating them and using them together would produce a result in the fastest way possible. Python tests were run to measure how the calculation time changes for different scripts and combinations of libraries. The fastest approach was tested on different data sizes with different functions.

Figure 5.1, for example, is a comparison of how long it takes to convert a number of reviews to tokens and how much additional time it takes to convert these same tokens to lemmas.

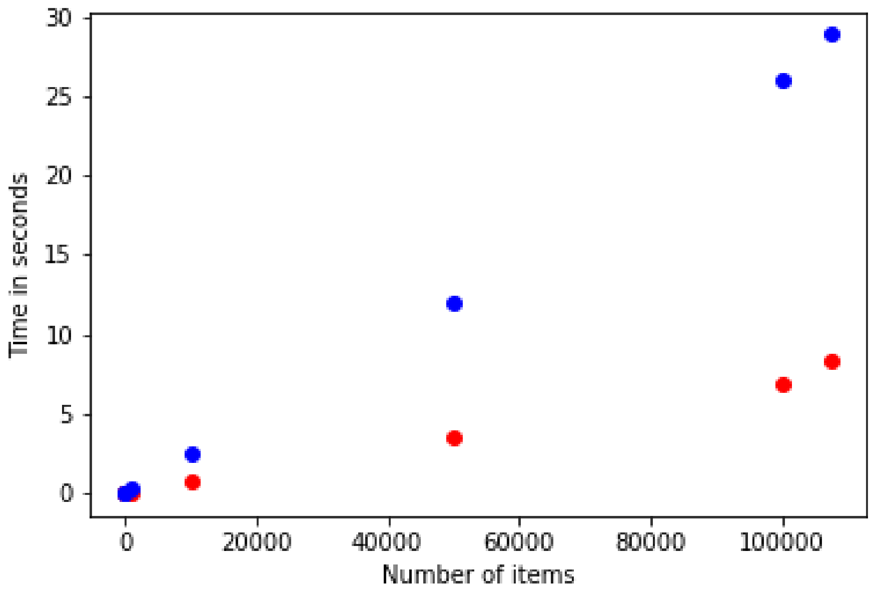


Figure 5.1: Time it takes in seconds for extracting different number of tokens and lemmas. The red dots represent the tokens and the blue ones the lemmas.

Both, the tokenization and the lemmatization, include the removal of stopwords. As visible from the graph, increasing the amount of the data is strongly correlated with the loading times. Additionally, common phrases and token sets algorithms were run on a test dataset for comparisons. This process took the longest amount of time as evident from Table 5.3. Sentiment analysis was ran on all the words from the text. This feature was ultimately not featured in the tool due to unsatisfactory results.

Table 5.3: Time it takes to extract tokens, lemmas, phrases and determine sentiment from 107 401 text reviews. What these processes do, is described in Chapter 3. The Phrases extraction is done using SpaCy's standard noun chunks functionalities.

Method	**Time**
Tokens extraction	8 seconds
Lemmas extraction	37 seconds
Stems extraction	35 seconds
Phrases extraction	16.2 hours
Sentiment analysis	2.2 minutes

For 107 401 reviews it took 16 hours and 20 minutes to find a viable combination of tokens. This kind of delay is far from the desired time to process information. In combination with the fact that the resulting sets of phrases were not very illustrative of the content of the reviews, or at least not more telling than the tokens and lemmas were, this approach to text processing was not used in the tool. The method is regarded as too time-consuming and not beneficial enough.

Another disregarded method, as mentioned in Chapter 3, was the word stemming, which was transforming words after they were tokenized. The results here were not particularly pleasing with some stems not making sense. The stemming process was also taking almost as much time as the lemmatization. The time was calculated using different stemming methods, such as Porter [KG13], Snowball [PB] and Stanford. Subsequently, the lemmatizaion approach was considered superior. Tokenizaton and lemmatization of the entire dataset still requires almost a minute for the whole dataset. Therefore, this is done in a preprocessing step, and the results are stored in the database.

While this initial pre-processing takes some time for the whole database, it is expected it would be performed rather quickly if the data is updated with new text entries. Extracting the lemmas will not be as complicated as the trend recalculation. The time it takes for extracting the lemmas from a new collection of text reviews can be predicted, based on the information in Figure 5.1.

It also needs to be noted that the time depends on the length of each review and the number of stopwords present. Longer reviews will take more time to tokenize and lemmatize. Reviews with fewer tokens, which are stopwords, would naturally take more time to lemmatize.

Once the list of lemmas for each review is predefined, it can be used to figure out how often a word appears in a collection of reviews and even the TF-IDF scores can be calculated. This part of the tool is calculated dynamically in real-time. The lemmas are intentionally saved as a list in order to make counting and looping through the list as quick as possible. The time it takes for the finished tool to produce the words with the highest TF-IDF scores is measured and presented in Figure 5.2. All the measured times include the time it takes for the application to send and receive post requests to the server.

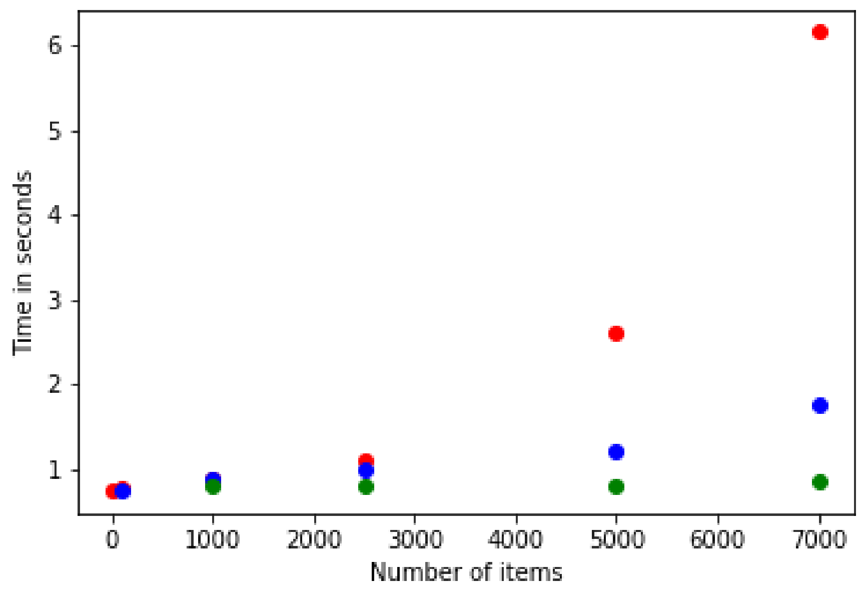


Figure 5.2: Average time it takes Review Watcher to complete a TF-IDF for a specific number of reviews and specific number of separate collections of lists of lemmas. The red dots are for 5 collections, the blue ones are for 10 collections and the green ones are for 50 collections.

Increasing the separate collections of reviews reduces the computational times of the process. This is most apparent when dealing with a large number of reviews.

The process of showing the most common words for a collection of reviews is naturally quicker than calculating the TF-IDF scores. The recorded times are present in Figure 5.3.

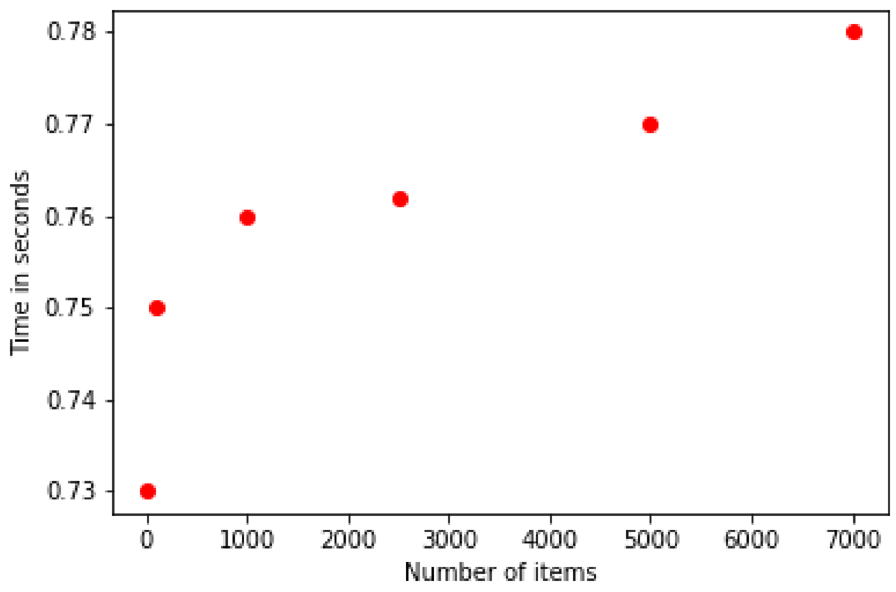


Figure 5.3: Average time it takes Review Watcher to complete a top words request for a specific number of reviews.

Another problem was that the TF-IDF calculation is performed very quickly on a small dataset, but the times are getting progressively worse if the data size is increased. Since no viable already build TF-IDF library was available, a custom algorithm was built specifically for this tool. Another problem with computing the IDF is how dependent the calculation time is on the number of separate review collections the data is split into. Having one review be one separate data collection slowed the processing algorithm. Tested on 7 000 reviews, as 7 000 separate collections of reviews, it takes nearly 32 seconds to compute the TF-IDF. Therefore, the algorithm was set to regard a collection of multiple reviews as one and stack together multiple lists of lemmas from adjacent reviews into one set of lemmas. This way of computing TF-IDF reduces the loading times.

One of the potential disadvantages that the developed tool has, is its high usage of memory. The tool was ran on a Chrome 80.0.3987.163 internet browser and when fully operational and executing tasks, the tool was using nearly 20 gigabytes of RAM. Most of this memory was used by the MongoDB server and the different Python-related script operations and data containment. This would indicate that for running the tool successfully on a different machine, first, an appropriate remote server will need to be assigned for deployment. The server side is taking most of the heavy load. After that is secured, the tool will be practically fully operational on any machine with Chrome that has CORS enabled. Review Watcher could cache up to 200 or 250 megabytes browser data, depending on the use. Having the necessary deployment equipment provided, the tool is expected to properly keep up to the recorded loading and execution times. The tool has only been hosted on a local server. Using it in a real-life setting might potentially, slightly influence the responsiveness of the tool, depending on the quality and speed of the user's internet connection.

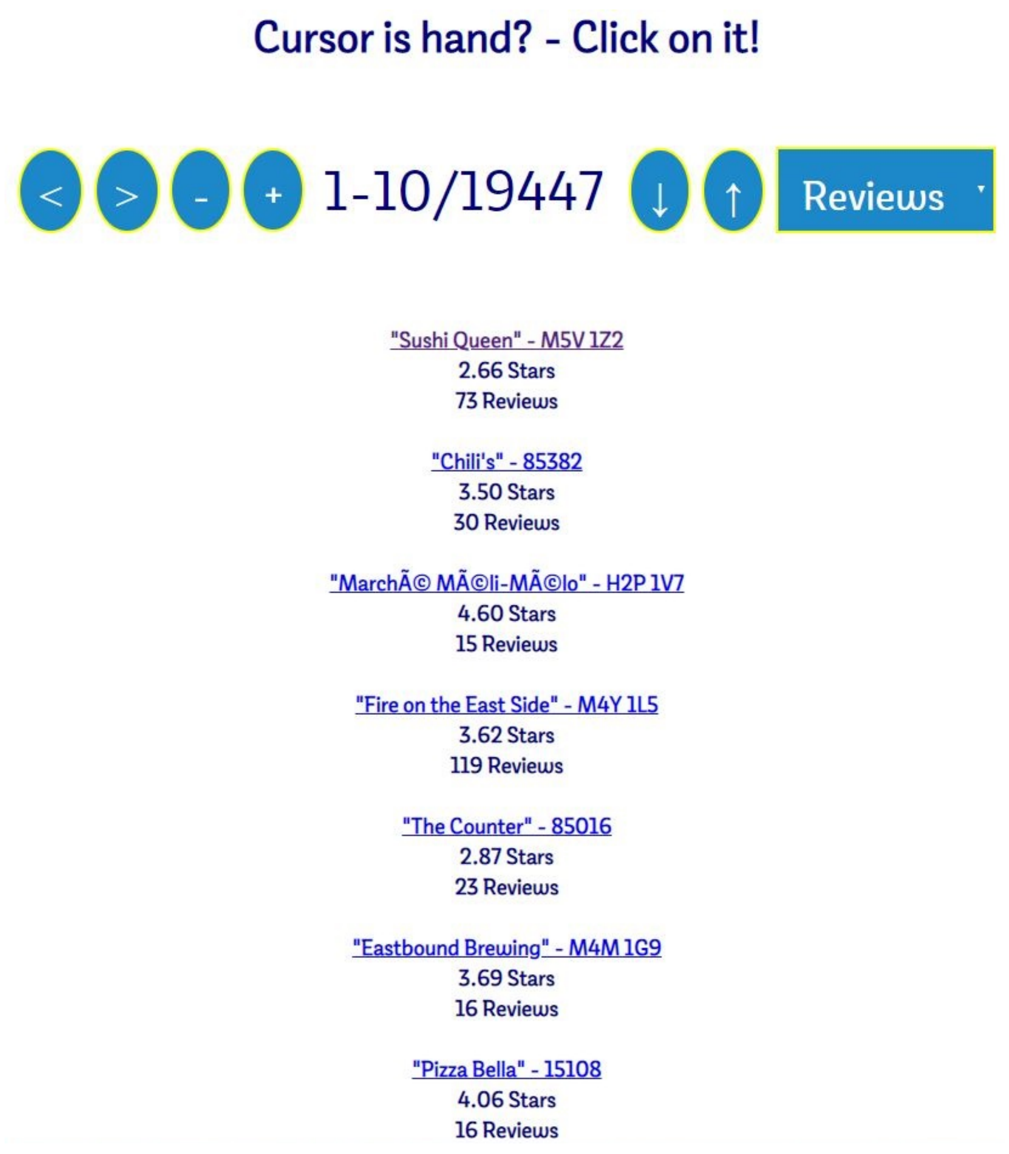
5.2 User Study

Having built a fully functioning tool that satisfies our set of performance benchmarks, we continue by testing the tool with users. We are performing a user study to determine if the tool is capable of answering the research questions, which were already listed in Chapter 1. Typically, there are two main evaluation methods, i.e., qualitative and quantitative [Car08]. For this particular thesis, we do not have predefined tasks that can be simply solved and measured. This is why we are performing a qualitative evaluation.

We employ qualitative evaluation method to gain a deeper understanding of the user's exploration process. We conduct an insight-based evaluation. It follows a think-aloud protocol, which is the basis for a comparative evaluation and qualitative insights analysis. We provide the test users with two tools, Review Watcher and the baseline (referred to as the Sample Tool), in order to also make a comparison between how users explore the data and which insights they gain with the two tools.

The Sample Tool was a simple copy of some of the functionalities of the Yelp site. Sample Tool's design is missing some major functionalities of Yelp because they were not considered important for the comparison with Review Watcher. The tool depends heavily on some of the programs, libraries and scripts already built for the Review Watcher. It was also built using a combination of Python scripts, hosted on the same Flask server. HTML mixed with JavaScript and Ajax post requests was used for the web pages and requests for data to the server. D3.js was not used here since no graphs, charts or word clouds were created.

The Sample Tool contains two pages. The first page is shown in Figure 5.4.


Cursor is hand? - Click on it!

"Sushi Queen" - M5V 1Z2
2.66 Stars
73 Reviews

"Chili's" - 85382
3.50 Stars
30 Reviews

"MarchÃ© MÃ©li-MÃ©lo" - H2P 1V7
4.60 Stars
15 Reviews

"Fire on the East Side" - M4Y 1L5
3.62 Stars
119 Reviews

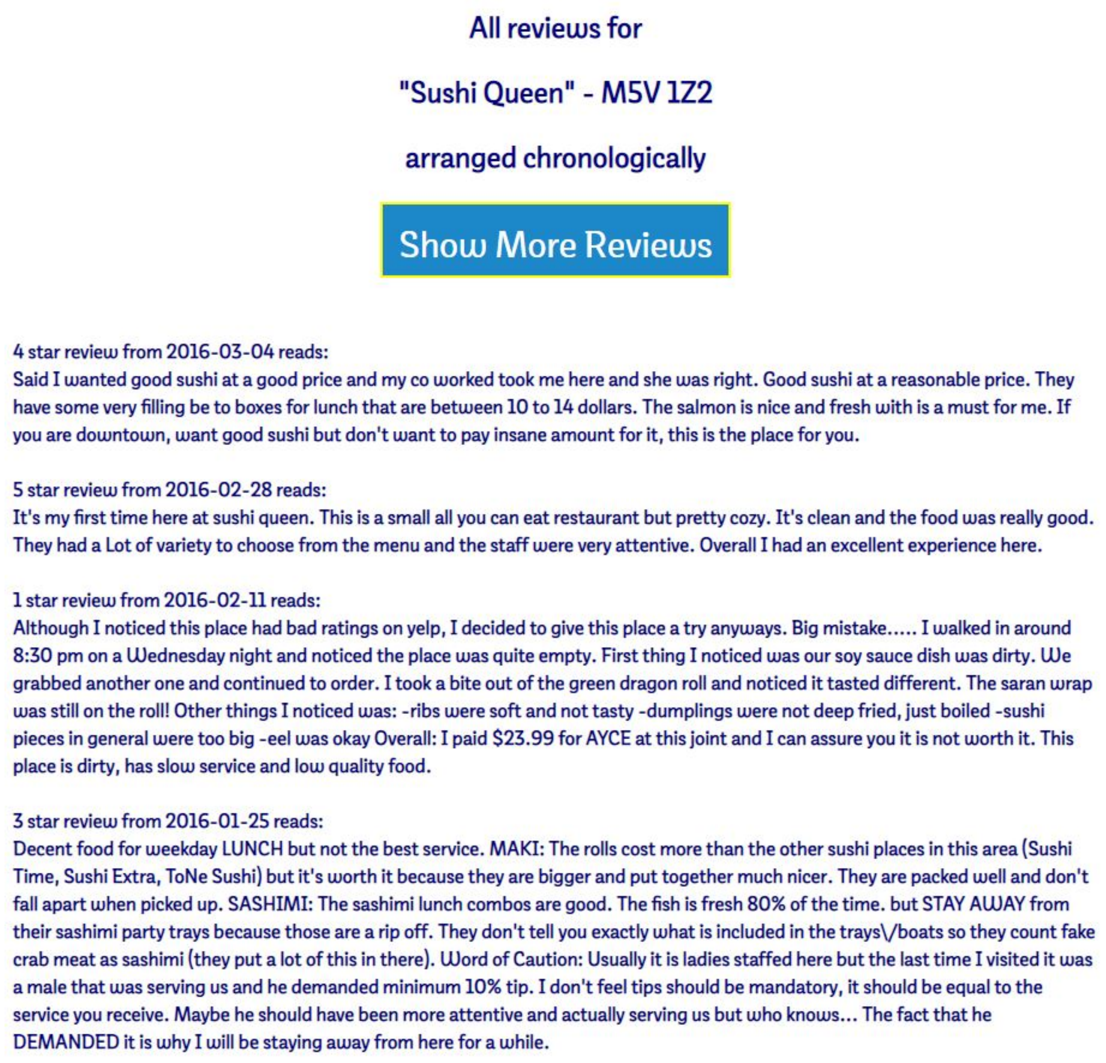
"The Counter" - 85016
2.87 Stars
23 Reviews

"Eastbound Brewing" - M4M 1G9
3.69 Stars
16 Reviews

"Pizza Bella" - 15108
4.06 Stars
16 Reviews

Figure 5.4: Sample Tool landing page.

On the page can be seen a list of restaurant names and their postal codes. Below each name we see the average star score of restaurants and their number of reviews. The list of restaurants can be reordered using the buttons on the top of the page. Their functionality is the same as in Review Watcher and was already explained in Chapter 1. Clicking on the restaurant links takes us on the second page. An example of the second page is visible in Figure 5.5.



All reviews for

"Sushi Queen" - M5V 1Z2

arranged chronologically

4 star review from 2016-03-04 reads:
Said I wanted good sushi at a good price and my co worked took me here and she was right. Good sushi at a reasonable price. They have some very filling be to boxes for lunch that are between 10 to 14 dollars. The salmon is nice and fresh with is a must for me. If you are downtown, want good sushi but don't want to pay insane amount for it, this is the place for you.

5 star review from 2016-02-28 reads:
It's my first time here at sushi queen. This is a small all you can eat restaurant but pretty cozy. It's clean and the food was really good. They had a Lot of variety to choose from the menu and the staff were very attentive. Overall I had an excellent experience here.

1 star review from 2016-02-11 reads:
Although I noticed this place had bad ratings on yelp, I decided to give this place a try anyways. Big mistake..... I walked in around 8:30 pm on a Wednesday night and noticed the place was quite empty. First thing I noticed was our soy sauce dish was dirty. We grabbed another one and continued to order. I took a bite out of the green dragon roll and noticed it tasted different. The saran wrap was still on the roll! Other things I noticed was: -ribs were soft and not tasty -dumplings were not deep fried, just boiled -sushi pieces in general were too big -eel was okay Overall: I paid $23.99 for AYCE at this joint and I can assure you it is not worth it. This place is dirty, has slow service and low quality food.

3 star review from 2016-01-25 reads:
Decent food for weekday LUNCH but not the best service. MAKI: The rolls cost more than the other sushi places in this area (Sushi Time, Sushi Extra, ToNe Sushi) but it's worth it because they are bigger and put together much nicer. They are packed well and don't fall apart when picked up. SASHIMI: The sashimi lunch combos are good. The fish is fresh 80% of the time. but STAY AWAY from their sashimi party trays because those are a rip off. They don't tell you exactly what is included in the trays\/boats so they count fake crab meat as sashimi (they put a lot of this in there). Word of Caution: Usually it is ladies staffed here but the last time I visited it was a male that was serving us and he demanded minimum 10% tip. I don't feel tips should be mandatory, it should be equal to the service you receive. Maybe he should have been more attentive and actually serving us but who knows... The fact that he DEMANDED it is why I will be staying away from here for a while.

Figure 5.5: Sample Tool's second page. Showing the last few reviews for restaurant 'Sushi Queen'. Only the latest four reviews are visible in the figure.

On this page we see the last ten reviews the chosen restaurant has received, arranged chronologically, starting from the most recent one. Information, about the date of issues of each review and its score, is also visible. Clicking on the Show More Reviews button will list the next ten reviews for that restaurant.

The Sample Tool aims to present to the user all reviews for a restaurant in a simplistic way, without any graphs. No pictures or additional data has been used for this tool, that has not been used for Review Watcher as well. The design of the Sample Tool, despite mirroring the design of sites such as Yelp or TripAdvisor, is supposed to be simplistic and to point the user to exclusively the review data.

Test users are asked to verbally describe as detailed as possible what they find out from the data while working with the two interfaces. While there is no formal definition of insight, it is described as ‘individual observation about the data by the participant’ [SND05]. Some key characteristics of insights [Nor06] might be that they are:

• Complex - Insights are complex since they involve not only single data values but large amounts of data.

• Deep - Insights are not present from the beginning, but build up over time, bringing up more questions and thus generating depth.

• Qualitative - Insights can be subjective and ambiguous and are thereby not exact.

• Unexpected - Insights are often unforeseeable.

• Relevant - Insights give domain knowledge relevant meaning because they are embedded in the data and can connect the data to the existing domain.

The open-ended, think-aloud protocol ensures users take their time exploring the data and the tool. Afterwards, insights are coded and quantified to serve as a measure of the tools’ quality. Coding, in this case, means gathering all the findings from all the test users and determining similar themes and ideas among them. They can be later used to form thematic categories, thus, enabling the creation of quantifiable metrics and further statistical analyses. We use the gathered data from this user study to verify our hypotheses described in Chapter 1. After the users are finished exploring the data with both tools, they can share their overall opinion on their experiences in a semi-structured interview. Then, they are additionally prompted to answer a questionnaire, determined to measure the system usability score (SUS) [B$^+$96].

The questionnaire contains ten statements, which are only in regard to Review Watcher and not the Sample Tool. The user rates the statements using five-point a Likert-scale with ranges ordered from Strongly Disagree, Somewhat Disagree, Neutral, Somewhat Agree to Strongly Agree. The ten statements are as follows:

1. I think I would like to use this tool frequently.

2. I found the tool unnecessarily complex.

3. I thought the tool was easy to use.

4. I think that I would need the support of a technical person to be able to use this system.

5. I found the various functions in this tool were well integrated.

6. I thought there was too much inconsistency in this tool.

7. I would imagine that most people would learn to use this tool very quickly.

8. I found the tool very cumbersome to use.

9. I felt very confident using the tool.

10. I needed to learn a lot of things before I could get going with this tool.

The score of every statement, in a range from 0 to 4, contributes to the calculation of the final score. For every statement with an odd number, the contribution is the scale position minus 1 and for every statement with an even number, the contribution is 5 minus the scale position. The sum of all ten statement scores multiplied by 2.5 gives the final SUS. This score ranges from 0 to 100 where a higher score indicates better usability. An example final grading would be: Worst Imaginable (0-12), Awful (13-20), Poor (21-36), OK (37-51), Good (51-70), Excellent (71-85) and Best Imaginable (86-100) [BKM09]. A user score below 50 would be considered unsatisfactory, while a score above 75 would be a great indicator of quality [BKM08]. In our evaluation, we analyse not only the final scores, but every statement score separately.

The targeted users for the study were people with a strong background in economics and/or finance with great interest in data analysis. During the recruitment process, all test users were sought as non-profit volunteers. All volunteers were filtered based on their higher education and their connection to business and economics. Interest in gastronomy businesses and/or big data analysis was an optional criterion for candidates. The candidates' age, sex, and gender were not considered as criteria during recruitment. Participants were acquired using the word-of-mouth recruitment strategy.

Nine test users were gathered for the evaluation process. All the tests were conducted on the same machine with the technical specifications mentioned in Section 5.1. All users used Chrome as a preferred test browser. Four of the test participants were female and five were male. The youngest participant was 24 years of age, and the oldest one was 41. The average age was 29 years. Before starting the test, every user gave consent to their speech and interactions on the screen to be captured and recorded for further examination. They also agreed all their data from these recordings to be used, printed and discussed in this thesis, if deemed necessary. At the beginning of every test, each user had to verbally specify their professional qualification and previous experiences in working with tools such as TripAdvisor, Yelp, or other similar tools for internet reviews. All the participants had at least one Bachelor's degrees as a professional qualification. Four participants are Masters of Business Administration. Two are completing Master degrees in Business Informatics. One participant is a Master of Microeconomics. One is a Bachelor of Financial Mathematics. One is a Bachelor of International Business. All the users were conversing in the English language throughout the test.

In order to enable test subjects to learn by using both Review Watcher and the Sample Tool, the dataset of restaurants has been split. A random split of the restaurants was performed once. All 38 894 restaurants were split into two datasets containing 19 447 items each. These datasets were named dataset A and dataset B. The users were then presented with the two tools, i.e., Review Watcher and the Sample Tool, respectively. The order of the tools was counter-balanced, as well as the assignment of dataset to the tool. This is done in an effort to eliminate learning biased opinions and proof that insights from any data can be gathered. This way we have four different sequences of tests, as seen in Table 5.4.

Table 5.4: Test sequences: the order of the tools was counter-balanced, as well as the assignment of data set to the tool.

Sequence	**Initial tool**	**Initial dataset**	**Sequential tool**	**Sequential dataset**
1	Review Watcher	A	Sample Tool	B
2	Sample Tool	B	Review Watcher	A
3	Review Watcher	B	Sample Tool	A
4	Sample Tool	A	Review Watcher	B

The four different scenarios were performed twice each with the first eight people. The last person had the opportunity to blindly pick a test scenario. The first one was chosen. Prior to using the tools, no information on how the tool functions was given to the users. The only explanation given was about the nature of the data, how it was collected and provided by Yelp. The users were not given tasks before the start of the test. They were encouraged to explore on their own and to ask questions if necessary. During the tests, all the users had different questions of how the tool functions and needed little help from the interviewer. Sometimes users were assisted by the interviewer with questions in order to help them express their opinion easier and to a relevant point of the study. On average, every user spent about an hour in total working with both tools and answering the questionnaire.

After transcribing all recordings, we performed open coding on the users' insights. We grouped utterances into the following categories:

• Restaurant selections - This code was assigned whenever a user mentioned the name of a restaurant or selected it for closer inspection.

• Numbers - Mentions and counts of reviews, restaurants, or other types of numeric values, excluding ratings.

• Unexpected findings - Unexpected findings or astonishments.

• Unknown entities - Entities that were unknown to the user. Things that were never heard before. Restaurant names are excluded.

• Reviews read - Reviews read or skimmed through by users. Presented as a number of reviews.

• Temporal development – People reasoning about the development of restaurants over time.

• Restaurant comparisons - Comparisons between restaurants.

• Restaurant descriptions – Users summarizing the perceived data to infer opinions about the restaurant.

• Tool comparisons - Comparisons between different tools. They can be not only between Review Watcher and Sample Tool, but also between other tools the test users have used.

This category, unlike the rest, is not about insights about the underlying data. It is more of an indication of usability issues.

We use the number of insights per category to check our hypothesis with a further statistical analysis. After gathering and counting all the mentions for the different categories for the two tools, we created graphs to visualise the results (Figures 5.6 - 5.14).

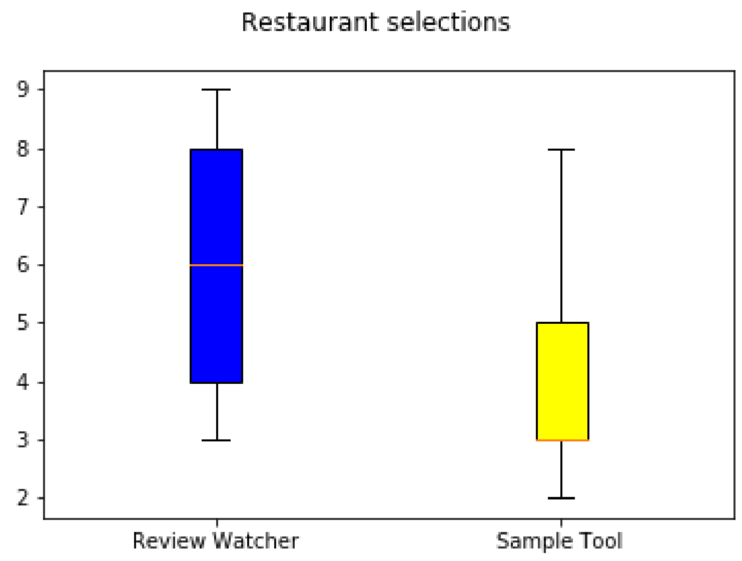


Figure 5.6: Mentions of restaurant selections per user for the two tools.

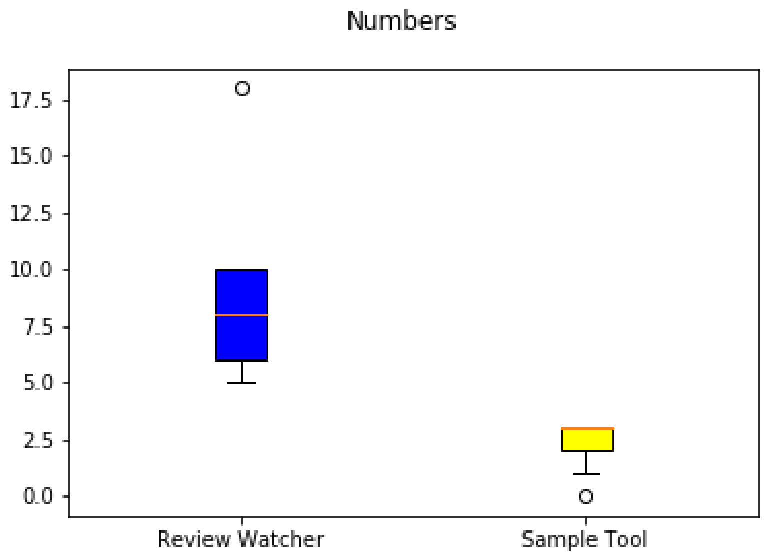


Figure 5.7: Mentions of numbers per user for the two tools.

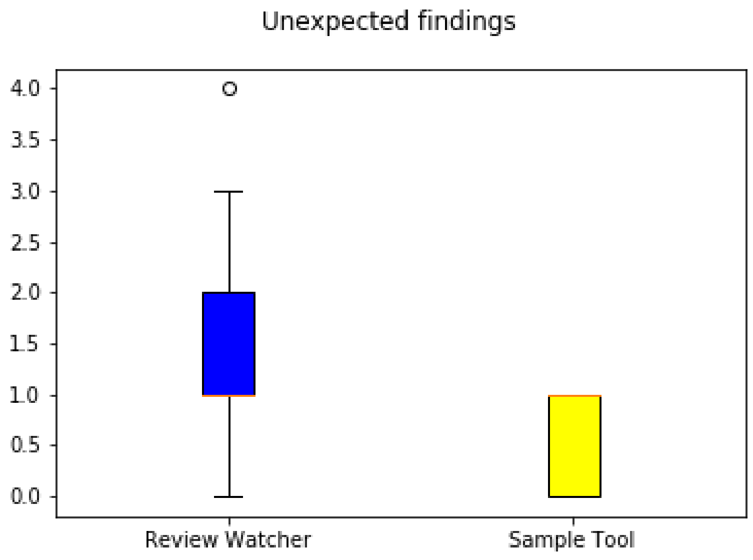


Figure 5.8: Mentions of unexpected findings per user for the two tools.

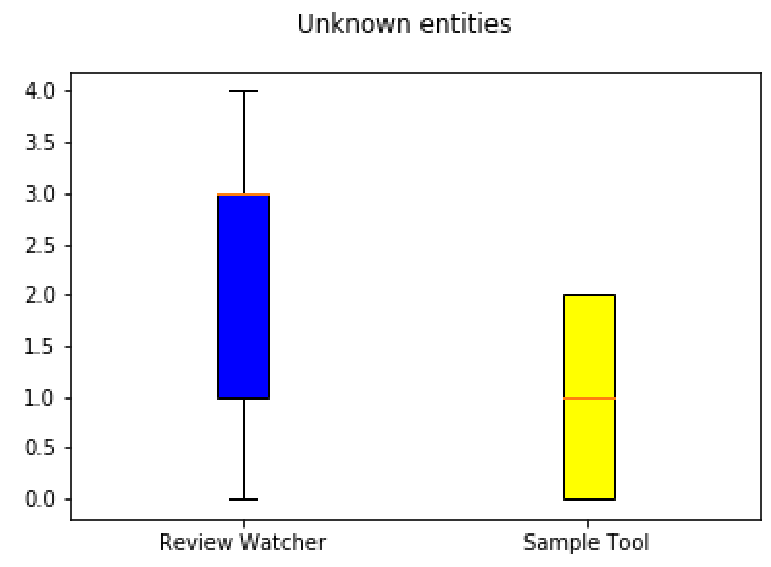


Figure 5.9: Mentions of unknown entities per user for the two tools.

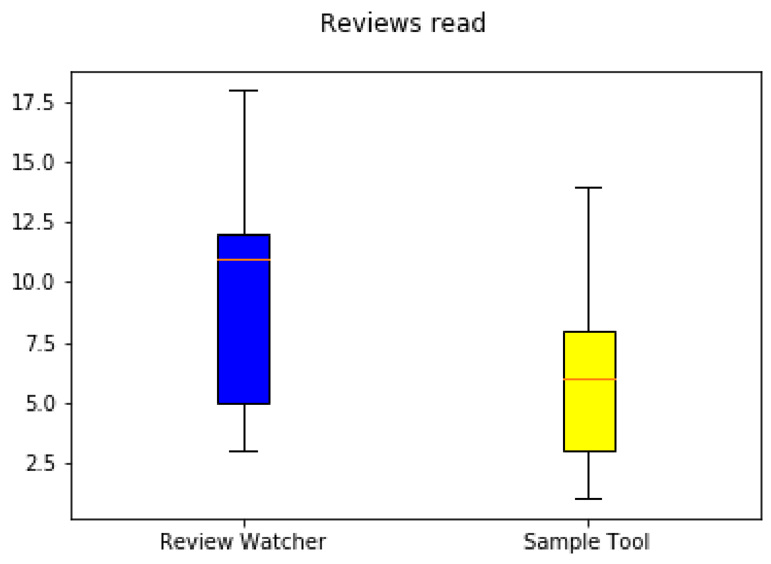


Figure 5.10: Mentions of reviews read per user for the two tools.

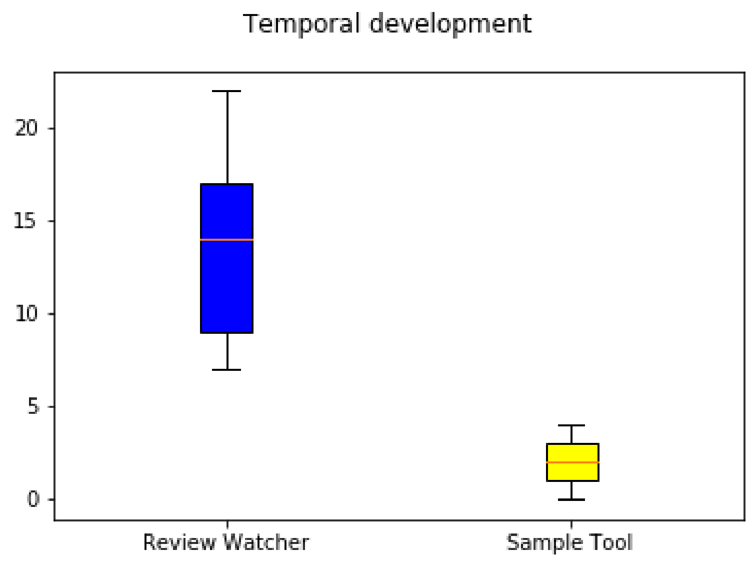


Figure 5.11: Mentions of temporal development per user for the two tools.

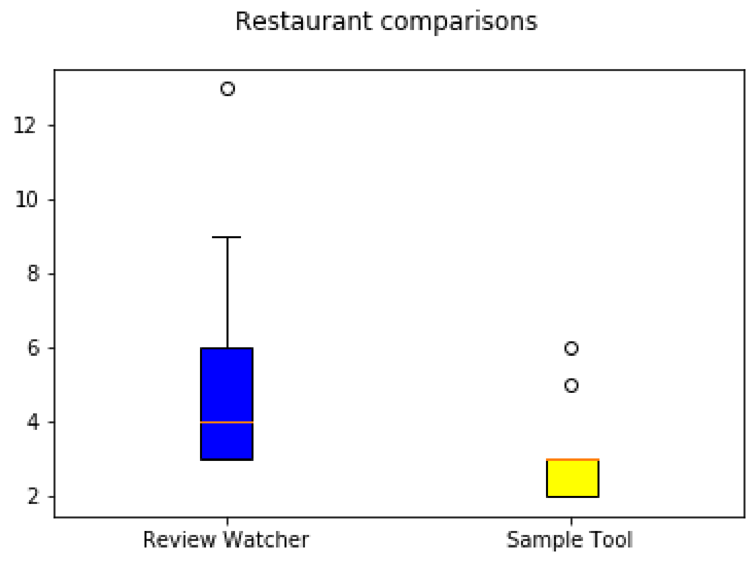


Figure 5.12: Mentions of restaurant comparisons per user for the two tools.

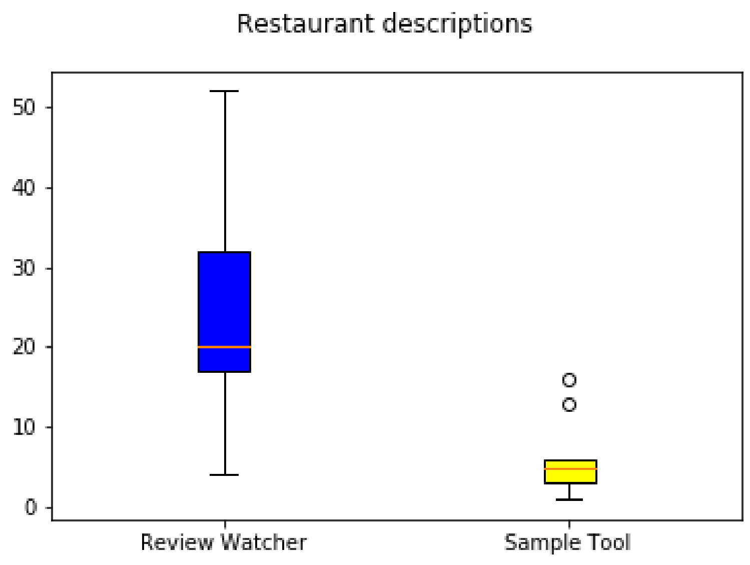


Figure 5.13: Mentions of restaurant descriptions per user for the two tools.

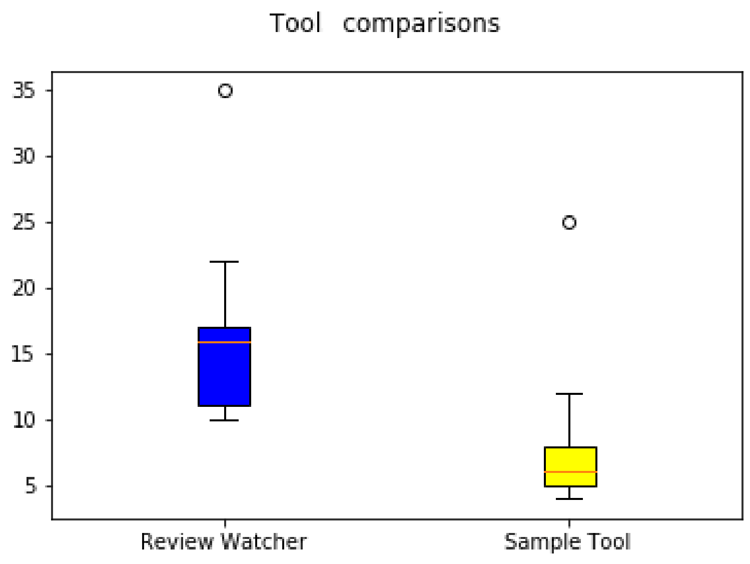


Figure 5.14: Mentions of tool comparisons per user for the two tools.

We also show in Figure 5.15 the time users spent with the tools.

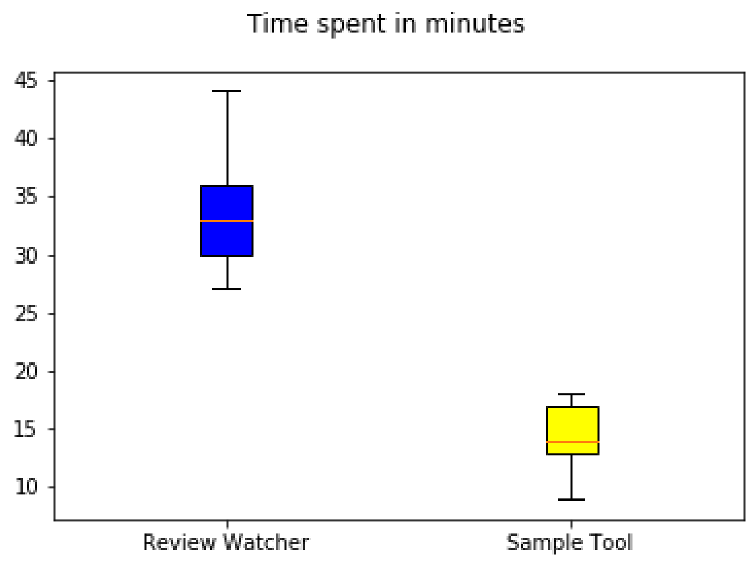


Figure 5.15: Time users spent using the two tools in minutes.

Wilcoxon signed-rank tests were performed on the results as well, in order to see differences in the results for the two tools for each category. That can be seen in Table 5.5.

Table 5.5: The results from the Wilcoxon signed-rank test on the different codes of the user evaluation.

Code	**Z**	**p**
Restaurant selections	8.5	0.18
Numbers	0	0.01
Unexpected findings	0	0.04
Unknown entities	0	0.02
Reviews read	3	0.04
Temporal development	0	0.01
Restaurant comparisons	2	0.04
Restaurant descriptions	2.5	0.02
Tool comparisons	1	0.01
Time spent with the tools	0	0.01

From the charts and the statistical analysis, it is apparent that users generally made significantly more note-worthy references using Review Watcher. This is true for all the different codes, safe for the restaurant selections.

From the results in Figure 5.15 it is apparent users spent significantly more time working with Review Watcher than with the Sample Tool. This might contribute to why there was not a lot of similarities in the number of utterances, based on the Wilcoxon signed-ranked test results. To reflect on this we recompute the values of utterances on a per-minute basis (Figures 5.16 - 5.24).

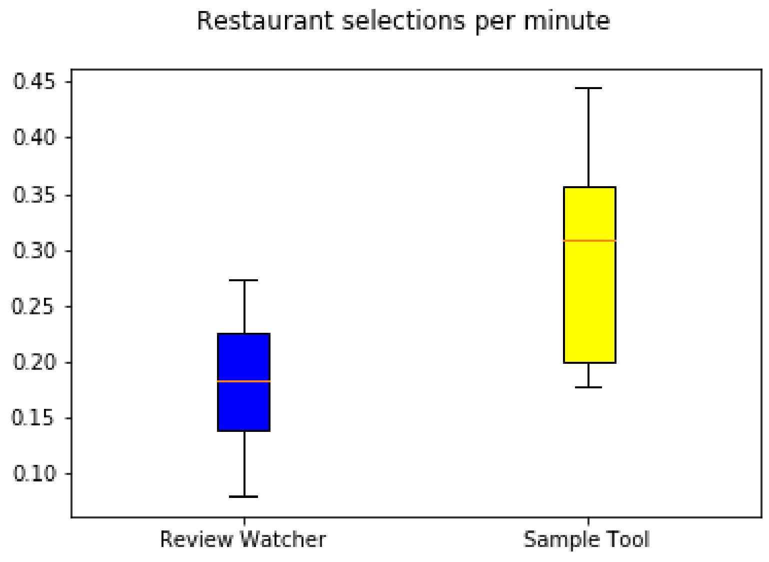


Figure 5.16: Mentions of restaurant selections per user, per minute for the two tools.

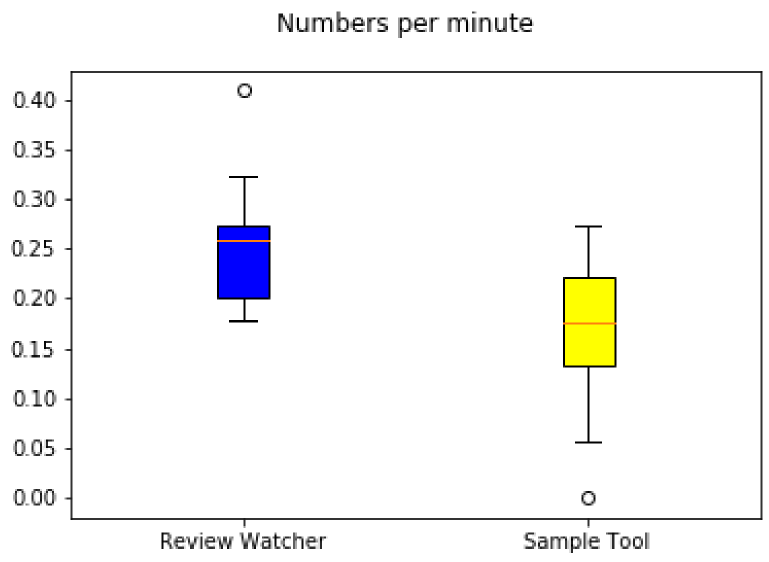


Figure 5.17: Mentions of numbers per user, per minute for the two tools.

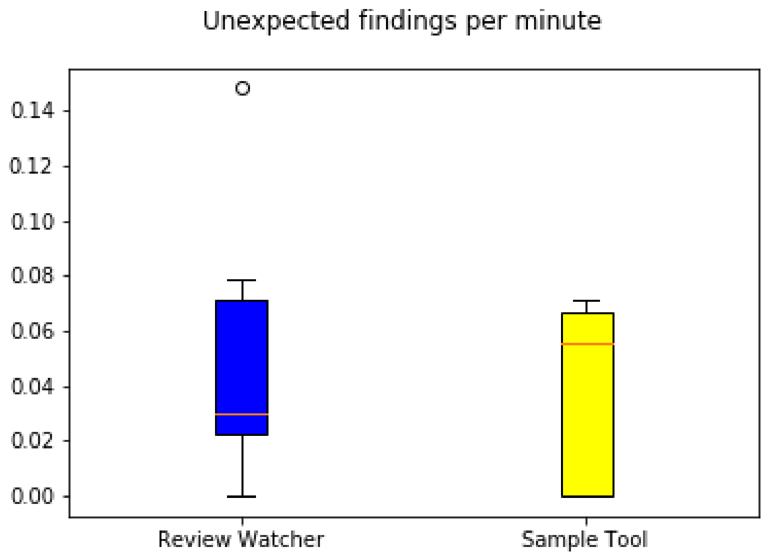


Figure 5.18: Mentions of unexpected findings per user, per minute for the two tools.

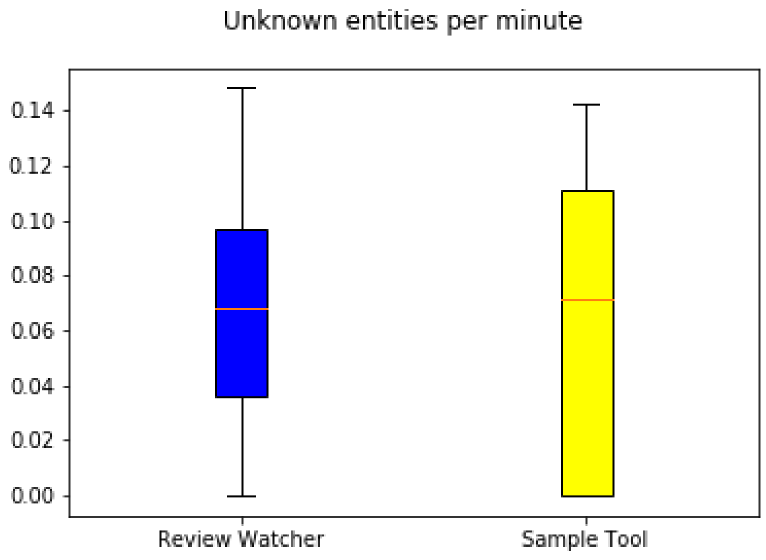


Figure 5.19: Mentions of unknown entities per user, per minute for the two tools.

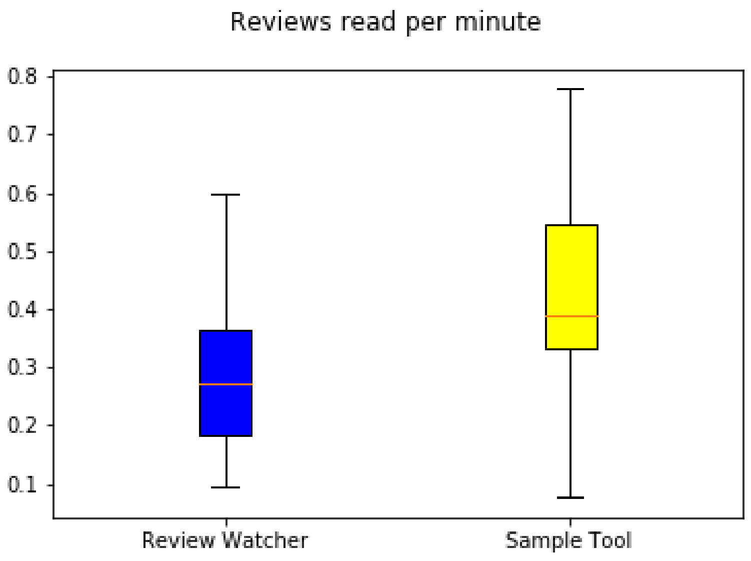


Figure 5.20: Mentions of reviews read per user, per minute for the two tools.

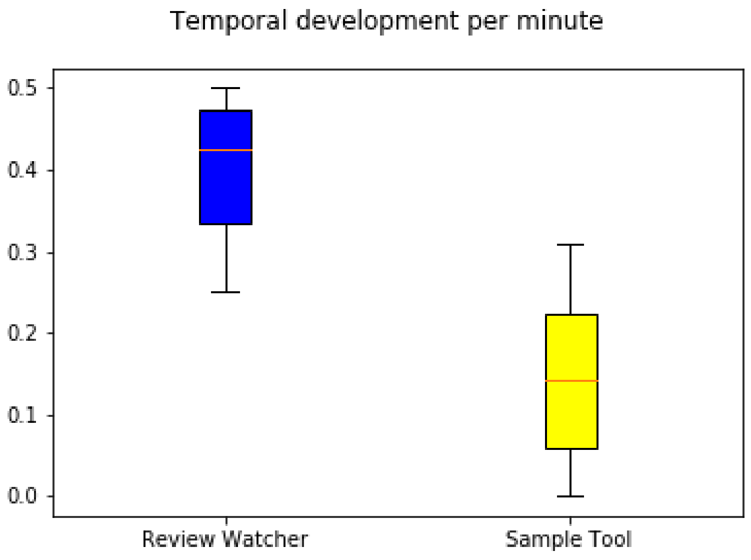


Figure 5.21: Mentions of temporal development per user, per minute for the two tools.

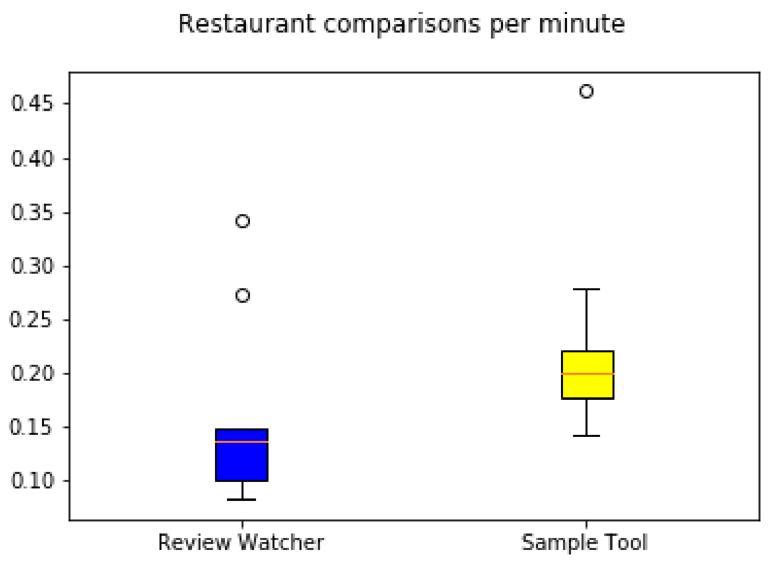


Figure 5.22: Mentions of restaurant comparisons per user, per minute for the two tools.

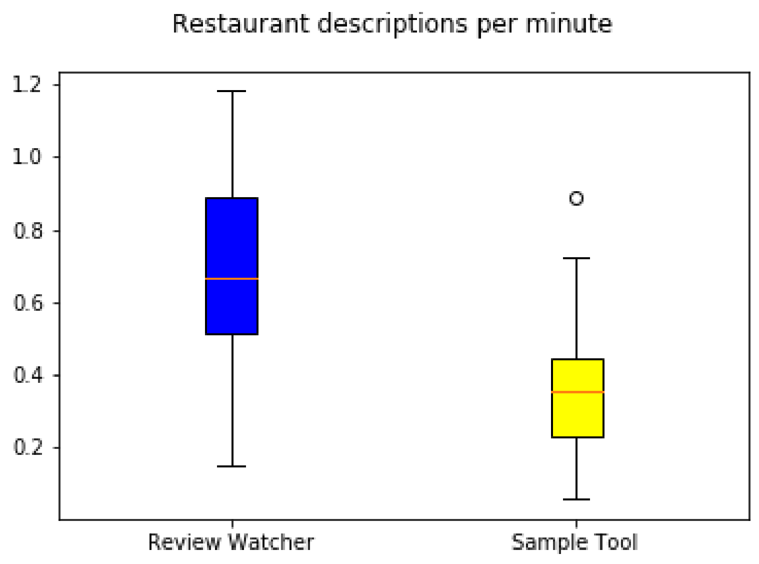


Figure 5.23: Mentions of restaurant descriptions per user, per minute for the two tools.

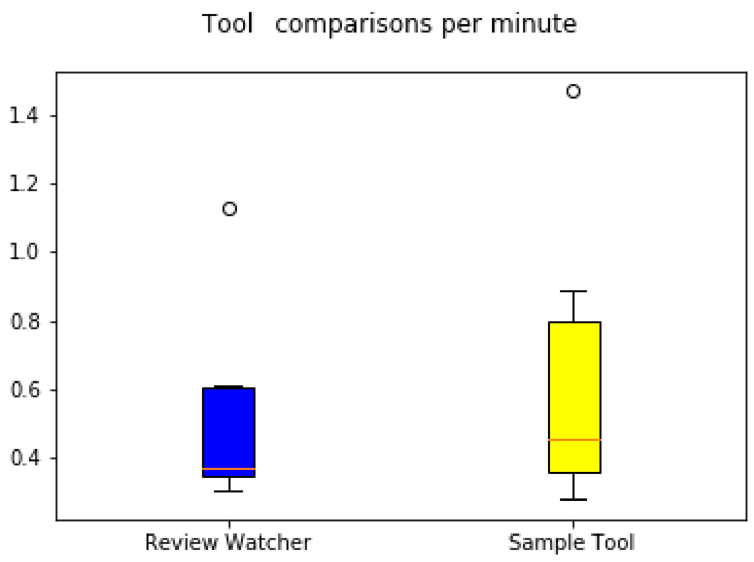


Figure 5.24: Mentions of tool comparisons per user, per minute for the two tools.

As observed from the temporally normalised figures, the values of the Sample Tool have increased and caught up with the Review Watcher for most of the time. It shows that the higher number of insights, was primarily because users spent more time exploring the data. This could imply that users had the feeling, there is more information to explore using the Review Watcher interface. Notably, users were mentioning temporal aspects and describing restaurants more frequently using Review Watcher. They did, however, took their time tinkering with some of the unfamiliar functionalities introduced by Review Watcher. This lead to the number of insights per minute to be somewhat biased in the final results.

Wilcoxon signed-rank test results for the per-minute results is also available in Table 5.6. The per-minute Wilcoxon signed-rank results have greatly changed in comparison to those in Table 5.5. As expected, only the values for the numbers and the temporal development remain unchanged.

The results from the SUS were rated according to the norm and their satisfaction level is present in Figure 5.25. The scores from all the users, for all the statements, rated from 0 to 100, is present on the graph. The arithmetic mean of all the user scores is 75, which can be described as an excellent score. The averages for each of the ten different questions is also shown in the graph. These averages are, again, given on a 0 to 100 scale. While some outliers are noticeable for some of the statements, the average scores for every statement is as follows:

1. I think I would like to use this tool frequently: 58.33

2. I found the tool unnecessarily complex: 72.22

Table 5.6: The results from the Wilcoxon signed-rank test on the different codes of the user evaluation on a per-minute basis.

Code	**Z**	**p**
Restaurant selections	3	0.04
Numbers	0	0.01
Unexpected findings	7	0.46
Unknown entities	17	0.89
Reviews read	11	0.17
Temporal development	0	0.01
Restaurant comparisons	3	0.02
Tool comparisons	16	0.44
Restaurant descriptions	9	0.11

3. I thought the tool was easy to use: 63.89

4. I think that I would need the support of a technical person to be able to use this system: 83.33

5. I found the various functions in this tool were well integrated: 86.11

6. I thought there was too much inconsistency in this tool: 83.33

7. I would imagine that most people would learn to use this tool very quickly: 69.44

8. I found the tool very cumbersome to use: 72.22

9. I felt very confident using the tool: 69.44

10. I needed to learn a lot of things before I could get going with this tool: 91.67

Some test subjects tend to be unsure whether or not they will use this tool frequently and how easy it is to work with it. This is indicated by some lower SUS scores as a result. This hints at the tool's limitations and potential areas for future improvements. The mean SUS from all the users are still however described as good. On average, the best imaginable scores, received the statements about whether the tool requires additional, beforehand knowledge and whether the tool is well integrated. This indicates that the specific features , that were developed for Review Watcher, are perfectly operational and accessible to new potential users.

From the restaurants that people selected for analysis and discussion, there are no restaurants that were chosen more than two times by different people. Test participants were extensively using the filter features to select which restaurants to view and explore further. Participants were often sorting restaurants by trend, number of reviews, and overall rating. These often preferred sorting criteria led them into viewing the same sets of restaurants. Users were often deciding to view a single restaurant from a filtered set

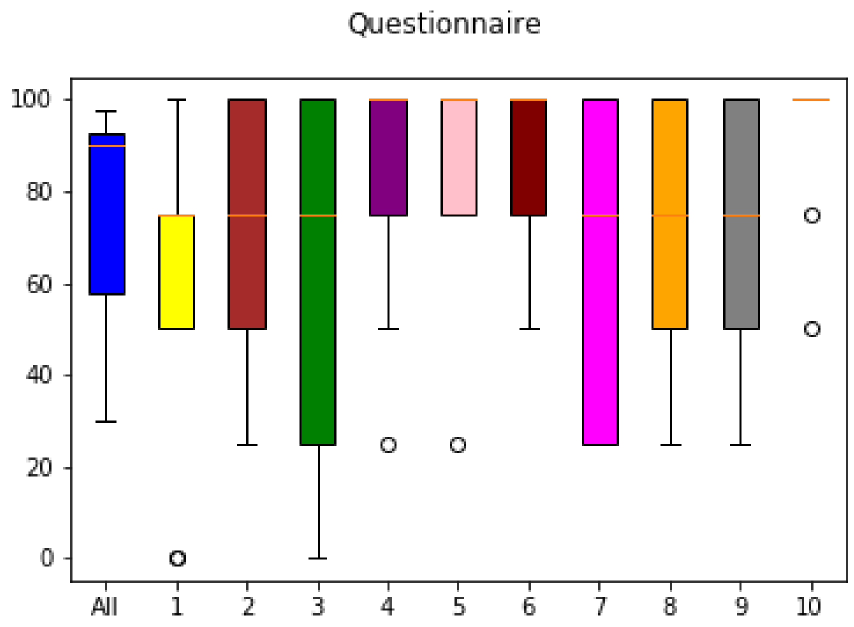


Figure 5.25: Questionnaire results. Overall results are displayed to the left. The results for the separate questions are also presented on a 0 to 100 scale. The median values are marked with a red line.

based on random principles. Since they were left dealing with a vast restaurant collection, choosing the same restaurant was rarely an option.

Having a more concrete view at the topics that users were discussing, we notice some reoccurring ones. Some user quotes about their experience with Review Watcher are listed below. The list of quotes is not complete, but it represents a sample of the more common types of insights people gathered.

• 'They say the food is excellent. Service has improved, so they have five stars now.'

• 'So I see that the price for this restaurant is good for the clients. They like this one.'

• '7000 reviews! Maybe they do not let you leave this restaurant, unless you write an opinion? That is why it has many reviews.'

• 'It seems that the more it is progressing the time the more people are interested in making reviews. I think that has to do with culture in general. People now are used to seek all kinds of information online. Even where to eat.'

• 'So they are complaining mostly about the service, when they make reservations. Again a line by the door, waiting for the tables.'

• 'If I would want to invest in some restaurant I would probably choose some that has not so good review. But the reviews are getting better and better.'

- ‘I see that the most commonly used words in the five star reviews are meat, place, good. Whereas people who gave it one star have used words such as time, table, food. Maybe they had issues with the service or something like that.’

- ‘There are months with five star reviews, for example February and March, and then two months without five star reviews, then once again featuring five star once. It tells me that it is probably some seasonal business, which is visited by a certain type of customers during this exact periods.’

- ‘That is very interesting actually. Burger, wait, good, get. That is something actually quite useful, because I can see what the most common words was. So for instance if this was potentially a good restaurant, because I can see a lot of positive reviews, I can see if people have waited long.’

- ‘2011 is where I suppose they had the most business. Plenty of customers. Because I can see a lot of reviews. 2012 less reviews.’

- ‘And 2016 just one review. I guess before the restaurant got shot down.’

- ‘If you own an Italian restaurant, you can see restaurants with Italian food, for example. How their reviews change over time.’

- ‘This is actually pretty useful because it is visual. You do not have to read through all the reviews, you can look at it and know, not on average, but just to see how many star reviews each one has in general.’

- ‘For example if you look at here you can see the words that are horrible. It has ‘good’ as well, but if you have ‘horrible’ as a top word, you would reconsider going there.’

- ‘Because you basically see immediately, approximately how many reviews the restaurant has and what is the distribution of the stars the other people gave. But, again it is also useful that you can see the words that appear in those reviews.’

- ‘800? What does that mean? In four star reviews and is the year 2017. Let us see. Aha, this is the place prepares all of its pizzas in! Alright, this is useful, I like this.’

- ‘For example the comment on the meat loaf was cool and underwhelming. That is something we have to change. And also the service was slow and unfriendly.’

- ‘Definitely, there is change. The number of reviews year by year is increasing and the number of positive reviews and also the number of negative reviews. But the highest peak for the negatives was in 2018 versus the most positive in 2017. Overall the trend is positive.’

- ‘Philly cheese steak sandwich - Of course, this must be the hit.’

Some restaurants, that users found interesting while using the tool, are shown as examples below. The restaurant shown in Figure 5.26, for example, was regarded as a restaurant with polarising reviews.

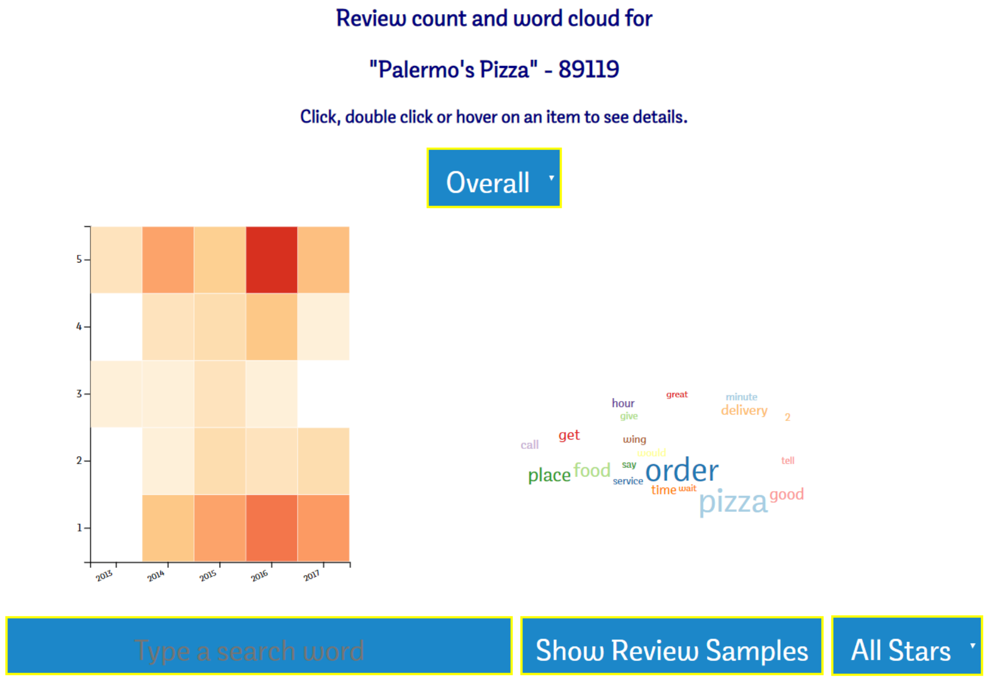


Figure 5.26: Showing information about the restaurant ‘Palermo’s Pizza’.

A test participant was able to spot the polarising reviews using the matrix graph and later on, to determine the reasons for the different scores by looking at the TF-IDF words and the common words. Digging deeper into the reviews, using the review search options, the user uncovered people, who like the restaurant, praise the prize-food quality ratio and people, who dislike it and criticise the slow service. Arriving at such conclusions using the Sample Tool would have be a lot more cumbersome. Spotting a restaurant with polarising ratings is also made simple using Review Watcher’s bar-chart on the tool’s first page.

Another example restaurant is ‘Mon Ami Gabi’. Using the detail view, shown in Figure 5.27, a user concluded that the restaurant’s growing popularity can be mostly attributed to people getting accustomed to publishing reviews online. The fact, that the number of reviews was growing over time, was not apparent when using a ranked list of reviews.

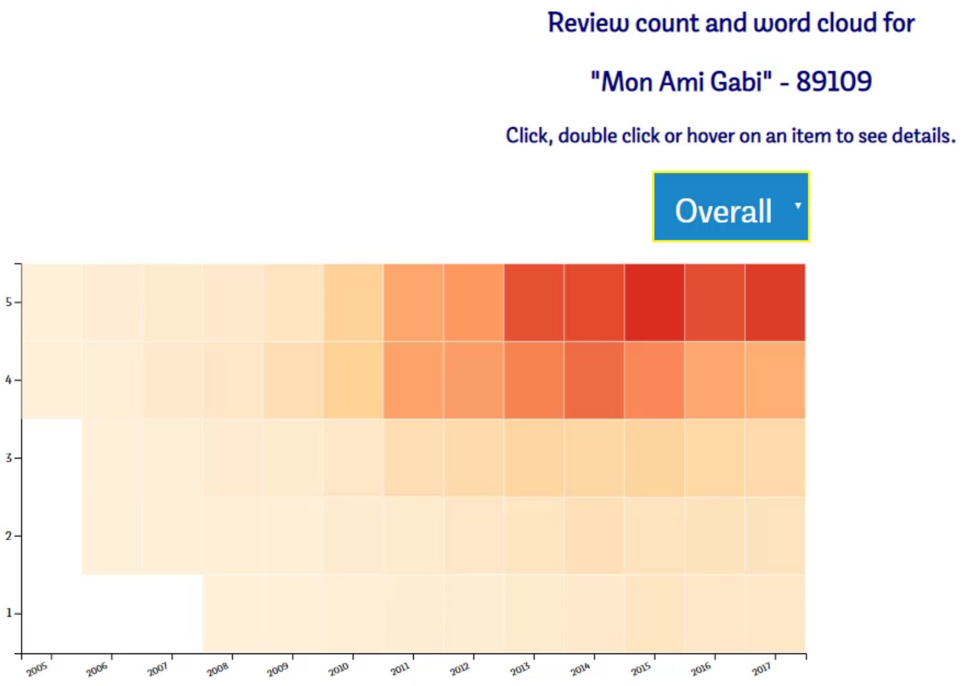
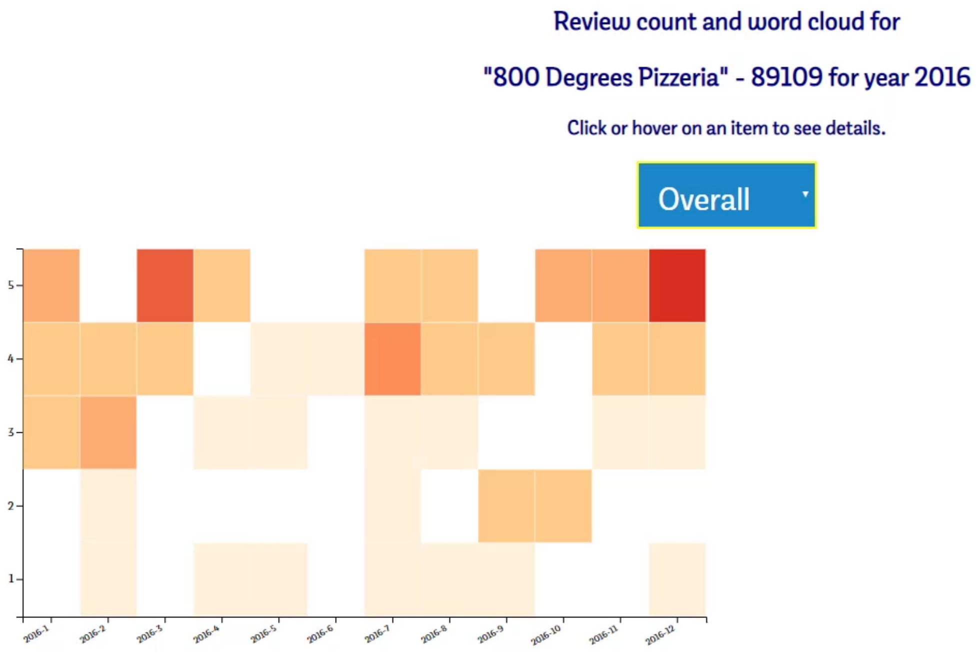


Figure 5.27: Showing information about the restaurant 'Mon Ami Gabi'.

Another example of gathered insights using the tool was discovering seasonality in the review scores. Such an example is shown in Figure 5.28. The user employed the monthly detail view to notice that the restaurant was more popular during the colder months of the year. Using the keywords and the sample reviews, the user concluded that the menu of the restaurant was one of the major reasons the place was a preferred destination during some seasons. Noticing seasonal trends is close to impossible when using the Sample Tool.


Review count and word cloud for

"800 Degrees Pizzeria" - 89109 for year 2016

Click or hover on an item to see details.

Figure 5.28: Showing information about the restaurant '800 Degrees Pizzeria' for the year 2016.

Noticing changes in overall review scores is also made more apparent using Review Watcher. An example of a restaurant with a trend of changing reviews is shown in Figure 5.29.

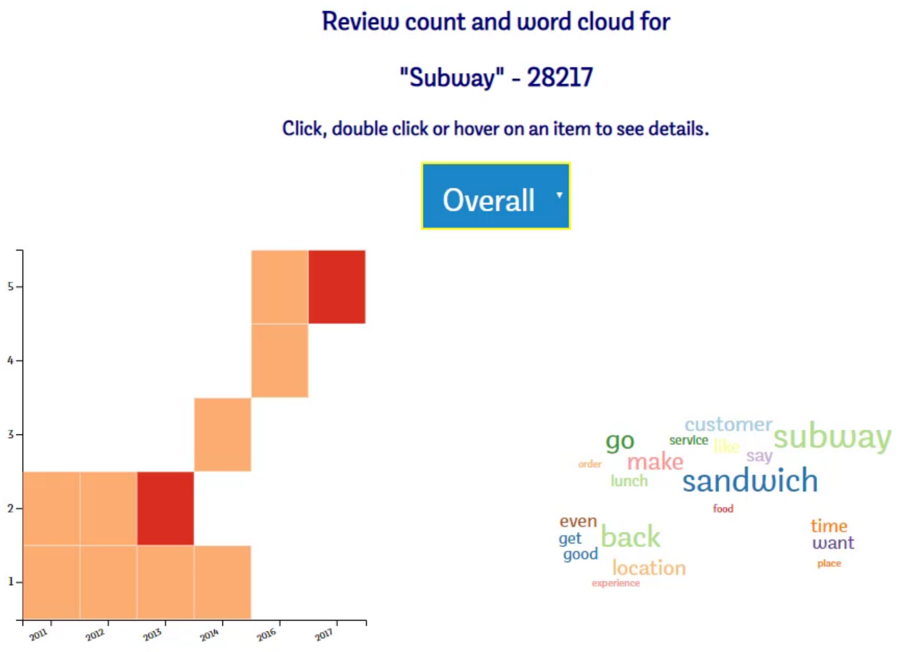


Figure 5.29: Showing information about the restaurant ‘Subway’.

5.3 Discussion

We use the results from the evaluation to justify the work in this Master thesis. For this purpose, we return to the main research questions and try to answer them one by one. We try to either prove or disprove the already built hypotheses, from Chapter 1, based on the collected information so far.

For the question of how to identify a change in the reviews we try to prove if Review Watcher is able to present identifiable changes to test users. This hypothesis is proven correct by the test results. This should come as no surprise, considering Review Watcher’s special features that focus and highlight changes in rating scores, text reviews, and number of reviews. Features that explicitly show changes were not present in the Sample Tool. Users tend to spend a lot more time with Review Watcher, examining and discussing temporal developments, compared to the Sample Tool. Test participants are mostly using the temporal heatmap feature of the tool to notice rating changes and common words in order to spot rating changes easily. TF-IDF and word clouds are also described by users as useful in pinpointing specific topics and content changes over the years. Regular review evaluation tools, already present in Yelp and TripAdvisor might be difficult to apply when trying to spot a content change if working with a lot of reviews. Some of the users were also impressed with the tool’s capabilities to highlight seasonal and monthly changes in ratings. Something that both restaurant owners and clients could find useful.

The second hypothesis is that the created tool can be used to identify categories of items.

This hypothesis is being largely disproved. While users did tend to make more restaurant comparisons using Review Watcher, the difference is not that vast and mostly attributed to the more time spent with the tool. The per-minute analysis from Figure 5.22 supports this claim. There, the average restaurant comparisons per minute is even higher for the Sample Tool. Users also tend to provide more restaurant descriptions if using Review Watcher, compared to using the Sample Tool. Some users were able to make up to 50 restaurant descriptions for a given time. Having that information in mind, drawing only five comparisons on average with the Sample Tool is a relatively low number.

Users often pointed out that the tools would benefit from a restaurant location or type meta data. Such features were not present in Review Watcher. Adding them at later stages however, is a real possibility and a potential improvement for the future. Users generally had mixed opinions about presenting restaurant ratings in a stacked bar chart. While some users liked the idea of finding similarities between restaurants based on the charts, others pointed out that just having an average rating indicator and review counter is a simpler and easier way to gather the same information. Some users found the Trend sorting useful when trying to categorise restaurants. However, the feedback from them suggests that this sorting algorithm alone is not sufficient for a practical categorisation of items.

The next hypothesis states that the created tool is preferred to identify major topics with. This hypothesis is proven correct. As already mentioned, the created tool was a great way for users to get an overview of a restaurant's review themes. Users were able to make around five times more restaurant descriptions using Review Watcher in comparison to the Sample Tool. Users often pointed to the word cloud for a quick way to get an overview of what a particular restaurant is about. The ability to afterwards search reviews, containing specific terms, led to test participants to better grasp what reviewers are concerned about. The number of reviews read per person for Review Watcher is not much bigger than with the Sample Tool. This fact might indicate that people were able to identify topics with less effort using Review Watcher. When comparing the two tools, users often stated that they can see them being employed in different scenarios by different people. Test users sometimes acknowledged that Review Watchers gives them a quick overview of a particular restaurant topic.

The fourth hypothesis states that the created tool is able to better distinguish between permanent and transient topics. This statement is proven right by both the high number of restaurant descriptions and the high number of temporal development insights. Users were often recorded filtering specific restaurant features and attributing them to a specific point in time. This behaviour was not present at all when they were using the Sample Tool. Users were also mentioning numbers a lot more often when using Review Watcher. They were gathering a lot more statistical insights about the majority of restaurant reviews and were able to spot sudden appearances of topics. Furthermore, users were able to also reason why these topics were transient. They were mostly helped by the time-based word clouds and the TF-IDF functions of the tool. Some users attempted to find transient topics using the Sample Tool. They were either discouraged by the need to read through a large number of reviews, or alternatively were uncertain of whether just a couple of reviews are a solid indicator for an actual topic change.

The following hypothesis states that the created tool is able to identify sudden increases or decreases in review scores. This is largely proven correct. Many users resorted to using the Trend and the Variance sort to track either gradual or polarising changes, respectively. Then, they were often using the matrix view to pinpoint the time and the direction of the score change. All test users were able to find restaurants with fluctuating review scores and pinpoint the time of the changes. Since users were seen examining only around six restaurants on average, their ability to find sudden rating changes among these restaurants is indicative of the tool's efficiency. In comparison, the Sample Tool conversations rarely involved temporal development. On the other hand, it has to be noted, that users are very often picking different restaurants. Some users, while using the Sample Tool, were also able to spot sudden rating changes just by quickly skimming through the chronologically arranged list of reviews. In turn, we can conclude that Review Watcher is able to identify sudden score changes quicker, easier, but with about a similar accuracy, as compared to the Sample Tool.

The last hypothesis is that the tool is able to identify topics of great importance. This hypothesis is being somewhat accepted by the results from the user evaluation. Users were able to make significantly more descriptions of a particular restaurant using Review Watcher. Test participants were noticing particular important topics for a restaurant and the identity of a restaurant as a whole. Figuring out why reviews change and how a particular topic might be the reason for it, is something that a lot of users found out by using the tool. Users stated that finding a general topic that might be important is doable with Review Watcher. On the other hand, some test participants revealed that delving deeper into a topic is easier with the already existing tools. This could be attributed to the fact that the Sample Tool does provide immediately all the text reviews for a restaurant, whereas Review Watcher provides only a sample of the reviews. This can be easily rectified in the future by introducing to Review Watcher the functionality to list all reviews. In the rare cases when users were comparing restaurants, they explained that there are a few topics which are often common for all restaurants. Such topics were the taste of the food, the client services, and occasionally the prices and the available places. While users did find these topics of interest using Review Watcher, they were mostly unsurprised by the results. The number of unexpected findings and unknown entities encountered using both tools is considered very low, usually not more than three per person and sometimes even zero. An example for an unexpected finding was that people were surprised at the sheer number of reviews some restaurants have and how that number is, usually, gradually increasing over time. As unknown entities test participants often regarded specific food recipes or cooking techniques. Recipes were regularly mentioning by users in comments. This might indicate that users rarely learned about the existence entities that are influential to review scores.

Having analysed the SUS results and the general feeling the people have about the created tool, we can definitely conclude that it left a positive impression on the users.

The majority of test users gave the tool a very good usability score. They pointed out that Review Watcher is not difficult to work with and could be useful in a few different scenarios. As evident from the number tool comparisons test participants made during the evaluation interviews, users were not shy of examining how Review Watcher compares to similar tools they have used. Most users were agreeing that the tool can be very beneficial to investors or restaurant owners, because it provides them with a restaurant history, gives them a quick overview of potential restaurant strengths and weaknesses, and lets them scout and compare other businesses. Users were also quick to point out that the Sample Tool and the already existing tools are more appropriate and targeted towards potential clientele and tourists, rather than business analysts. Few users suggested that combining the two tools available for testing would be an ideal possibility for an in-depth analysis of review data.

5.4 Limitations and Future Work

Review Watcher is naturally not without its limitations. Some of the drawbacks of the tool were already pinpointed during development. One is the need to spend additional time and resources in order to load new data into the existing database. Appropriate formatting, recalculating the trend parameters, and extracting lemmas are all necessary operations that need to be performed properly before data is being loaded into the tool. The tool also requires sufficient computational resources to run the server, as described in Section 5.1. Reducing the memory usage of the server should be considered. A few minutes of server starting time is also required if the database has been updated. Such technical issues could be solved in the future as technology progresses.

One optional research question that this Master thesis asked was whether the tool can be used to identify categories of restaurants. A few different types of clustering methods were used during development to help with this problem. They, however, did not produce satisfactory results and were left out from the final stages of development. Additional research in this direction could still be beneficial. The sorting algorithms that users tested did spark some sort of restaurant comparison discussions. However, the sorting was still not convincing enough to convince that the tool is indeed capable of categorising restaurants better than already existing tools.

The user test did, however, provide numerous suggestions to how the tool can be further developed in the future. Some users suggested to include more filters for restaurants. Filters based on the restaurant's location, menu possibilities, establishment type or whether the restaurant is still functioning, were some common suggestions. Adding all these features is theoretically possible, because information about them already exists in the Yelp databases. Some users also expressed suggestions for minor visual changes to make the tool more appealing.

One additional feature that was considered during development and later suggested by testers, was the polarity distinction of words. The idea was to use sentiment analysis tools to interpret word meanings. Depending on the sentiment polarity of a word, we can colour code it in the word cloud or in the TF-IDF word results. This idea is still considered doable as a potential future improvement.

Another idea was combining the Yelp data with some Google Maps functionalities. This could help users to gather more information as to where the specific restaurants are located. This could potentially add to further insights, based on the new information. It could potentially provide a unique look into how whole areas with restaurants are changing over time and why. It could also prove useful to restaurant owners or investors to locate dangerous competitors that might be influencing their regional market share.

The different user suggestions is a sign of Review Watcher‘s potential for future improvement. While the tool is already completely functional and suitable for a working environment, additional research and functionalities can lead to more benefits.

CHAPTER 6
Conclusion

In conclusion, this Master thesis managed to develop a novel approach for the visual exploration of textual and numerical reviews and provide a functioning tool that can be used for temporal analysis. This thesis manages to answer the questions of how the presented approach and tool handle the answering of given research questions.

It could be shown that Review Watcher is easier for users to identify changes in the reviews and determine the reason for these changes, compared to other existing tools. Review Watcher is also capable of revealing contrary trends in review scores and assist users in finding the reasons for these trends. Finding topics of great importance to reviewers is also made easier with the tool. Review Watcher would indicate the general reasons why restaurants receive positive reviews and why they receive negative reviews.

Review Watcher is able to provide a quick overview of review topics, spot score changes over time, and determines reasons for it. The presented approaches result in a lightweight tool, capable of loading big data, granted it is structured in a specific way. The tool showcases the use of visualisations of the review scores for each restaurant as a superior alternative to a ranked list. The tool provides an intuitive way of visualising review score changes over time. Its way to extract common words and essential words for a given period makes it accessible to spot trends and content changes. Pre-processing steps are still required, however, in order to make word extraction possible.

Review Watcher did not facilitate the identification of restaurants with similar customer opinions or similar review trend developments over time. Ways to do this in the future could be to use additional geographic data, do topic modelling, or do time series clustering of review scores.

From the qualitative user evaluation, it was proven that the tool stands out among the existing common interfaces tailored for end users. The user tests also revealed some of the tool's limitations that can be found solutions to in the future. The take-away message from this thesis would be that the created tool and its visualisation approaches can be beneficial to business data analysts and business owners. This message was derived from the test participants, who were all people with buiness backgrounds. Combining Review Watcher with already existing functionalities for examining restaurants and reviews, could be of great benefit. Adding more functionalities and combining them with the existing features, in order to overcome some limitations, is a testament to the tool's potential for further future improvement.

List of Figures

List of Tables

Bibliography

[Agg01] Nikhil Aggarwal. Itertools - https://www.geeksforgeeks.org/python-itertools/, Accessed Online: 2020-05-01.

[AMST11] Wolfgang Aigner, Silvia Miksch, Heidrun Schumann, and Christian Tominski. *Visualization of Time-Oriented Data.* Human-Computer Interaction Series. Springer, 2011.

[Asg16] Nabiha Asghar. Yelp dataset challenge: Review rating prediction. *CoRR*, abs/1605.05362, 2016.

[AYHK11] Basak Alper, Huahai Yang, Eben Haber, and Eser Kandogan. Opinion-blocks: Visualizing consumer reviews. In *IEEE VisWeek 2011 Workshop on Interactive Visual Text Analytics for Decision Making*, 2011.

[B^+96] John Brooke et al. SUS-a quick and dirty usability scale. *Usability evaluation in industry*, 189(194):4–7, 1996.

[BBN12] Eivind Bjørkelund, Thomas H Burnett, and Kjetil Nørvåg. A study of opinion mining and visualization of hotel reviews. In *Proceedings of the 14th International Conference on Information Integration and Web-based Applications & Services*, pages 229–238, 2012.

[Bez01] Jeffrey Bezos. Google maps - https://www.aboutamazon.com/, Accessed Online: 2020-05-01.

[BH01] Cynthia Brewer and Mark Harrower. Colorbrewer 2 0 - https://colorbrewer2.org/, Accessed Online: 2020-05-01.

[BKM08] Aaron Bangor, Philip Kortum, and James Miller. An empirical evaluation of the system usability scale. *International Journal of Human–Computer Interaction*, 24(6):574–594, 2008.

[BKM09] Aaron Bangor, Phil Kortum, and James Miller. Determining what individual SUS scores mean: Adding an adjective rating scale. *J. Usability Stud.*, 4:114–123, 04 2009.

[Bos01] Mike Bostock. D3 data-driven documents - https://d3js.org/, Accessed Online: 2020-05-01.

[BSH$^+$16] Benjamin Bach, Conglei Shi, Nicolas Heulot, Tara Madhyastha, Tom Grabowski, and Pierre Dragicevic. Time curves: Folding time to visualize patterns of temporal evolution in data. *IEEE Transactions on Visualization and Computer Graphics*, 22(1):559–568, 01 2016.

[CAB$^+$20] SM Mazharul Hoque Chowdhury, Sheikh Abujar, Khalid Been Md Badruzzaman, Syed Akhter Hossain, et al. Product review analysis using social media data based on sentiment analysis. In *Innovations in Computer Science and Engineering: Proceedings of 7th ICICSE*, pages 527–533, Singapore, 2020. Springer Singapore.

[Cao15] Qiong Cao. *Some topics on similarity metric learning.* PhD thesis, University of Exeter, UK, 2015.

[Car08] Sheelagh Carpendale. Evaluating information visualizations. In *Information Visualization: Human-Centered Issues and Perspectives*, pages 19–45. Springer Berlin Heidelberg, 2008.

[CC16] Nan Cao and Weiwei Cui. Overview of text visualization techniques. In *Introduction to Text Visualization*, pages 11–40, Paris, 2016. Atlantis Press.

[CDBF10] Fanny Chevalier, Pierre Dragicevic, Anastasia Bezerianos, and Jean-Daniel Fekete. Using text animated transitions to support navigation in document histories. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '10, pages 683–692, New York, NY, USA, 2010. ACM.

[CLT$^+$11] Weiwei Cui, Shixia Liu, Li Y. Tan, Conglei Shi, Yangqiu Song, Zekai Gao, Huamin Qu, and Xin Tong. Textflow: Towards better understanding of evolving topics in text. *IEEE Transactions on Visualization and Computer Graphics*, 17:2412–2421, 2011.

[CTL18] Juntian Chen, Yubo Tao, and Hai Lin. Visual exploration and comparison of word embeddings. *Journal of Visual Languages and Computing*, 48:178 – 186, 2018.

[Dan01] Michal Mimino Danilak. Langdetect - https://pypi.org/project/langdetect/, Accessed Online: 2020-05-01.

[ED07] Geoffrey Ellis and Alan Dix. A taxonomy of clutter reduction for information visualisation. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1216–1223, 11 2007.

[Fou01] Python Software Foundation. Counter - https://docs.python.org/3.6/library/collections.html#collections.counter, Accessed Online: 2020-05-01.

[Gar01] Jesse James Garrett. Asynchronous javascript and xml - https://developer.mozilla.org/en-us/docs/web/guide/ajax, Accessed Online: 2020-05-01.

[GJG$^+$15] Soliman Gad, Waqas Javed, Sohaib Ghani, Niklas Elmqvist, Tom Ewing, Keith Hampton, and Naren Ramakrishnan. Themedelta: Dynamic segmentations over temporal topic models. *Visualization and Computer Graphics, IEEE Transactions on*, 21:672–685, 05 2015.

[Gro] The Stanford Natural Language Processing Group. Stanford topic modeling toolbox - https://nlp.stanford.edu/software/tmt/tmt-0.4/.

[HHWN02] Susan Havre, Elizabeth Hetzler, Paul Whitney, and Lucy Nowell. Themeriver: Visualizing thematic changes in large document collections. *IEEE Transactions on Visualization and Computer Graphics*, 8(1):9–20, 01 2002.

[HLLE14] Florian Heimerl, Steffen Lohmann, Simon Lange, and Thomas Ertl. Word cloud explorer: Text analytics based on word clouds. In *2014 47th Hawaii International Conference on System Sciences*, pages 1833–1842, 2014.

[HLZ08] Nan Hu, Ling Liu, and Jie Jennifer Zhang. Do online reviews affect product sales? The role of reviewer characteristics and temporal effects. *Inf. Technol. and Management*, 9(3):201–214, 09 2008.

[Hon01] Matthew Honnibal. Spacy - https://spacy.io/, Accessed Online: 2020-05-01.

[HY07] Michael Alexander Kirkwood Halliday and Colin Yallop. Lexicology: a short introduction. A&C Black, 2007.

[Inc01] Yelp Inc. Yelp dataset challenge - https://www.yelp.com/dataset/challenge, Accessed Online: 2020-05-01.

[JM14] Daniel Jurafsky and James Martin. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*, volume 3. 2014.

[KA15] Diane Kelly and Leif Azzopardi. How many results per page? A study of serp size, search behavior and user experience. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '15, page 183–192, New York, NY, USA, 2015. Association for Computing Machinery.

[Kau01] Stephen Kaufer. Tripadvisor - https://tripadvisor.mediaroom.com/us-about-us, Accessed Online: 2020-05-01.

[KG13] Wahiba Ben Abdessalem Karaa and Nidhal Gribâa. Information retrieval with porter stemmer: A new version for english. In Dhinaharan Nagamalai, Ashok Kumar, and Annamalai Annamalai, editors, *Advances in Computational Science, Engineering and Information Technology*, pages 243–254, Heidelberg, 2013. Springer International Publishing.

[Kum01] Nikhil Kumar. Linear regression python implementation - https://www.geeksforgeeks.org/linear-regression-python-implementation/, Accessed Online: 2020-05-01.

[LRKC10] Bongshin Lee, Nathalie Henry Riche, Amy K. Karlson, and M. Sheelagh T. Carpendale. Sparkclouds: Visualizing trends in tag clouds. *IEEE Transactions on Visualization and Computer Graphics*, 16:1182–1189, 2010.

[LTL19] Ziyu Lu, Haihui Tan, and Wenjie Li. An evolutionary context-aware sequential model for topic evolution of text stream. *Information Sciences*, 473:166 – 177, 2019.

[McK01] Wes McKinney. Pandas - https://pandas.pydata.org/about/index.html, Accessed Online: 2020-05-01.

[MHR01] Dwight Merriman, Eliot Horowitz, and Kevin Ryan. MongoDB - https://www.mongodb.com/company, Accessed Online: 2020-05-01.

[MKF$^+$02] Carlos Monroy, Rajiv Kochumman, Richard Furuta, Eduardo Urbina, Eréndira Melgoza, and Arpita Goenka. Visualization of variants in textual collations to analyze the evolution of literary works in the Cervantes project. In Maristella Agosti and Costantino Thanos, editors, *Research and Advanced Technology for Digital Libraries*, pages 638–653, Berlin, Heidelberg, 2002. Springer Berlin Heidelberg.

[MYW13] Baojun Ma, Hua Yuan, and Qiang Wei. A comparison study of clustering models for online review sentiment analysis. In Jianyong Wang, Hui Xiong, Yoshiharu Ishikawa, Jianliang Xu, and Junfeng Zhou, editors, *Web-Age Information Management*, pages 332–337, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

[NLT01] Team NLTK. Natural language toolkit - https://www.nltk.org/index.html, Accessed Online: 2020-05-01.

[Nor06] Chris North. Toward measuring visualization insight. *IEEE Computer Graphics and Applications*, 26(3):6–9, 05 2006.

[NPB18] Chakkrit Snae Namahoot, Sopon Pinijkitcharoenkul, and Michael Brückner. Travel review analysis system with big data (tras). In Meikang Qiu, editor, *Smart Computing and Communication*, pages 18–28, Cham, 2018. Springer International Publishing.

[PB] MF Porter and R Boulton. Snowball stemmer. 2001. *URL http://snowball. tartarus. org.*

[Pet01] Gerhard Peters. The American presidency project - https://www.presidency.ucsb.edu/, Accessed Online: 2020-05-01.

[PG01] Fernando Pérez and Brian Granger. Jupyter notebook - https://jupyter.org/about, Accessed Online: 2020-05-01.

[Por06] Martin Porter. An algorithm for suffix stripping. *Program*, 2006.

[Ron01] Armin Ronacher. Flask - https://flask.palletsprojects.com/en/1.1.x/, Accessed Online: 2020-05-01.

[RR01] Jens Eilstrup Rasmussen and Lars Eilstrup Rasmussen. Google maps - https://www.google.com/intl/en-gb/maps/about/, Accessed Online: 2020-05-01.

[Rub18] Yuliya Rubtsova. Reducing the deterioration of sentiment analysis results due to the time impact. *Information*, 9(8):184, 2018.

[She01] Eric Shepherd. Cross-origin resource sharing - https://developer.mozilla.org/en-us/docs/web/http/cors, Accessed Online: 2020-05-01.

[SJ88] Karen Sparck Jones. A statistical interpretation of term specificity and its application in retrieval. In *Document Retrieval Systems*, page 132–142, GBR, 1988. Taylor Graham Publishing.

[SND05] Purvi Saraiya, Chris North, and Karen Duca. An insight-based methodology for evaluating bioinformatics visualizations. *IEEE Transactions on Visualization and Computer Graphics*, 11:443–56, 07 2005.

[Sou01] Steve Souders. Call to improve browser caching - https://www.stevesouders.com/blog/2010/04/26/call-to-improve-browser-caching/, Accessed Online: 2020-05-01.

[SS01] Jeremy Stoppelman and Russel Simmons. Yelp - https://www.yelp.ie/about, Accessed Online: 2020-05-01.

[TBH19] Thang Tran, Hung Ba, and Van-Nam Huynh. Measuring hotel review sentiment: An aspect-based sentiment analysis approach. In Hirosato Seki, Canh Hao Nguyen, Van-Nam Huynh, and Masahiro Inuiguchi, editors, *Integrated Uncertainty in Knowledge Modelling and Decision Making*, pages 393–405, Cham, 2019. Springer International Publishing.

[uhU01] Zeeshan ul-hassan Usmani. Kaggle - https://www.kaggle.com/getting-started/44916, Accessed Online: 2020-05-01.

[WHS+18] Yong Wang, Hammad Haleem, Conglei Shi, Yanhong Wu, Xun Zhao, Siwei Fu, and Huamin Qu. Towards easy comparison of local businesses using online reviews. *Computer Graphics Forum*, 37(3):63–74, 2018.

[WWL+10] Yingcai Wu, Furu Wei, Shixia Liu, Norman Au, Weiwei Cui, Hong Zhou, and Huamin Qu. Opinionseer: Interactive visualization of hotel customer feedback. *IEEE Transactions on Visualization and Computer Graphics*, 16(6):1109–1118, 11 2010.

[Yel01] Inc. Yelp. Kaggle - https://www.kaggle.com/yelp-dataset/yelp-dataset, Accessed Online: 2020-05-01.

[YLK16] Young Seok You, Suan Lee, and Jinho Kim. Design and development of visualization tool for movie review and sentiment analysis. In *Proceedings of the Sixth International Conference on Emerging Databases: Technologies, Applications, and Theory*, EDB '16, pages 117–123, New York, NY, USA, 2016. ACM.