Review-Driven Generation of Visual Stories for Products

Using word embeddings to explore user-generated reviews

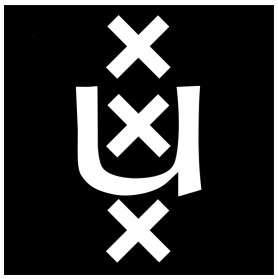
SUBMITTED IN PARTIAL FULFILLMENT FOR THE DEGREE OF
MASTER OF SCIENCE

MARC VORNETRAN
11569565

MASTER INFORMATION STUDIES
HUMAN CENTERED MULTIMEDIA

FACULTY OF SCIENCE
UNIVERSITY OF AMSTERDAM

July 23, 2018



1st Supervisor
Dr. Stevan Rudinac
University of Amsterdam

2nd Supervisor
Gosia Migut
University of Amsterdam

Review-Driven Generation of Visual Stories for Products

Using word embeddings to explore user-generated reviews

Marc Vornetran
University of Amsterdam
Amsterdam, The Netherlands
marc.vornetran@gmail.com

ABSTRACT

The prevalent success of e-commerce platforms has established ratings and reviews as the de-facto mechanism for consumers to build a knowledge base about products. Review aggregation websites such as *RateBeer* collect thousands of reviews for each product (i.e., beer) on their platform. Extracting the actual characteristics of a product from an abundance of reviews becomes a challenging task for users. This paper proposes the *Beerlytics* system which aims to generate visual stories of these products summarizing their core essence. It uses word embedding algorithms to train vector space models for each product using their associated user-contributed reviews. Using cluster centroids to create representative yet diverse summaries of a set of reviews. The vector space models provide the basis for nearest neighbor keyword extraction. Valuable insights into a product are provided by finding the closest words of a query in vector space. User testing on a frontend prototype using the generated information measures the effectiveness of the system.

KEYWORDS

Product summarization, visual stories, word embeddings, product reviews, user testing, clustering, user-centered design, natural language processing

1 INTRODUCTION

The widespread adoption of e-commerce has established consumer-generated reviews and ratings of products as a standard feature on merchant websites. These reviews provide guidance for potential new customers and help to meet their expectations after receiving an ordered product. Besides creating a knowledge database for consumers reviews contain valuable insights and measurements for both merchants and producers. While not every single customer authors a review especially popular products amass numerous reviews on large marketplaces during their life cycle. Reviews themselves differ in length, level of detail, quality of writing, structure, language, and opinion features [16].

However, e-commerce platforms of producers, merchants and large-scale retailers are by no means the only place for users to express their opinion about products. Review aggregation sites collect user reviews about products and services as a third party which does not offer the reviewed subjects themselves. Well established household names include: *Metacritic*[1] (entertainment), *Rotten Tomatoes*[2] (movies, tv), *IMDb*[3] (movies, tv, actors) and *TripAdvisor*[4] (vacation, flights, restaurants). Aggregators act as independent platforms where users can find averaged reviews in a uniform fashion.

As a more specific instance with a much more narrow focus *RateBeer*[5] collects information about (craft) beer and primarily consumer-generated reviews of beers and breweries. Established in 2000 it has since remained a popular exchange platform garnering more than five million reviews in total[6] with a rate of roughly 1 million reviews per year since 2016[7]. While more recent and exact figures remain hidden it is estimated that nearly 500,000 visitors view the site within a single month[8].

Due to the following factors, *RateBeer* acts as a testbed for this thesis project. Firstly, the large quantity of historical reviews and the constant stream of new content provides an abundance of data for processing and experimental testing. Secondly, according to *RateBeer's* quality assurance principles, low quality and nonsensical entries are swiftly removed from the platform through an extensive administration system. Furthermore, although *Anheuser-Busch InBev* acquired a minority stake of *RateBeer*[9] the site claims and emphasizes its mission to remain an independent platform which prohibits ratings by breweries and their affiliates. Finally, reviews are of a semi-structured nature containing free-form text but also scaled ratings in 5 distinct categories: (1) aroma, (2) appearance, (3) taste, (4) palate, and (5) overall.

The following parts of this paper are structured as follows. Firstly, section 2 examines related work in the fields of word embeddings and (visual) summarization. Secondly, section 3 contains the problem statement alongside the pursued research questions. Thirdly, section 4 details the approach taken for the realization of the *Beerlytics* system going from data collection and analysis, over the system design of the API and frontend prototype, to the final evaluation. Fourthly, section 5 discusses actual and potential shortcomings of the research project. Finally, section 6 concludes this research by recapping the proposed system and laying out recommendations for future work.

2 RELATED WORK

This section establishes a bird's-eye view of past and current research projects connected to the topics touched by the *Beerlytics* system. It covers different word embedding algorithms (Section 2.1 and the field of (visual) summarization (Section 2.2).

[1] http://www.metacritic.com
[2] https://www.rottentomatoes.com
[3] http://www.imdb.com
[4] https://www.tripadvisor.com
[5] https://www.ratebeer.com
[6] https://www.ratebeer.com/RateBeerBest/default_2013.asp, accessed: 16-03-2018
[7] https://www.ratebeer.com/ratebeerbest/default_2016.asp, accessed: 16-03-2018
[8] https://www.similarweb.com/website/ratebeer.com, accessed: 16-03-2018
[9] https://www.nytimes.com/2017/06/18/business/media/anheuser-busch-inbev-ratebeer.html, accessed: 16-03-2018

2.1 Word Embeddings

The distributional hypothesis from linguistics states that words which are used and appear in the same context tend to have similar meanings [14]. J.R. Firth popularizes the fundamental idea that *"a word is characterized by the company it keeps"* [11] which is picked up by the natural language processing (NLP) research community. Understanding the meaning of a word on a human-level is a core research objective of NLP [18]. Even though processing natural language on such a semantic level is still out of reach word embedding algorithms have significantly improved word similarity tasks [36]. These algorithms create a vectorial space for a vocabulary. Low-dimensional vectors containing real numerical values represent words from the vocabulary. It is shown that vectors in close vicinity to each other have a similar semantic meaning [18]. For example, the work from Mikolov et al. [24] demonstrates that calculating *vector("King") - vector("Man") + vector("Woman")* obtains a vector in close proximity to *vector("Queen")*. Similarity between vectors is commonly calculated using measures such as cosine similarity [36].

2.1.1 word2vec. The field of word embeddings was first revolutionized by the work of Mikolov et al. [22, 23]. Their software tool popularized *word2vec* as the umbrella term for a group of related approaches to produce word embeddings [13]. It provides a highly efficient way to calculate state-of-the-art embeddings which enabled companies such as Google to create pre-trained vectors for large-scale vocabularies [36]. The researchers propose the continuous bag of words (CBOW) and skip-gram models. Given a surrounding context c with a window size of k, CBOW calculates the conditional probability for a target word t within the context. Conversely, skip-gram predicts the words for the surrounding context c when given the central word t. Further work by Mikolov et al. [22] implements extensions on the original skip-gram model increasing performance and accuracy of the representations of less encountered words. It also seeks to reduce the limitation of word representations trying to express idiomatic phrases. Using vectors to represent full sentences improves the expressiveness of the skip-gram model for these phrases. Lastly, it also introduces negative sampling instead of the previous hierarchical softmax for training embeddings. It draws k samples from a negative noise distribution to achieve higher accuracy. Explaining the reasons for why negative sampling produces higher quality embeddings is still a question of ongoing research [13].

2.1.2 GloVe. Global vectors (GloVe) for word representations builds upon the ideas of previous work such as latent semantic analysis (LSA) [10] and word2vec [24]. Pennington et al. [27] argue that the model families used in these works each feature specific shortcomings. Supposedly, LSA suffers from a sub-optimal vector space structure while the skip-gram model disregards the global co-occurrence count in a corpus. The authors argue that by combining the model families and leveraging their advantages, previous embedding techniques can be outperformed [27]. Fusing global matrix factorization and local context windows produces a global log-bilinear regression model that makes efficient use of global corpus statistics. The performance of the newly proposed model is measured on word analogy tasks and demonstrates 75% accuracy [27].

2.1.3 fastText. Bojanowski et al. [5] from Facebook AI Research (FAIR)[10] create fastText which iterates on the skip-gram model. Their new approach features state-of-the-art training speed and enables the calculation of vectors for words which do not appear in the vocabulary. Previous techniques represent words using a single vector without taking morphological variations into account. fastText introduces subword information to vector representation by computing character n-grams for each morphological form of a word. The word is then represented as the sum of these n-gram vectors. Further research by Conneau et al. [8] employ fastText to outperform other methods training cross-lingual word embeddings. Their unsupervised method eliminates the need for bilingual corpora which is a requirement for supervised counterpart methods.

2.1.4 StarSpace. Wu et al. [35] iterate and expand on the previously discussed fastText model creating a general-purpose embedding technique called StarSpace. It can be applied to a multitude of use cases: (1) text/sentiment classification, (2) relevance ranking in information retrieval, (3) collaborative-based filtering recommendation, (4) embedding multi-relational graphs, and (5) training embeddings on a word/sentence/document level. The introduced model is a significant improvement over previous embedding techniques which exclusively train textual embeddings. StarSpace embeds entities of varying types in a common vectorial space. Entities are represented as bag-of-features which differ depending on the entity type. Most importantly, StarSpace enables the comparison of entities with different types.

2.2 Summarization

Summarization problems exist in a variety of domains including text, image, and video [3]. The nature of summarization is divided into the main categories of extractive [1] and generative summaries. Extractive summaries use excerpts or representative examples as a way to summarize the original content. In a survey on automatic text summarization [9], 3 essential aspects of summaries are reiterated: (1) single or multiple documents may be used for summarization, (2) important information needs to be preserved, and (3) a summary should be short.

Successful summarization projects have been implemented for different application domains.

Ren et al. [30] focus on selecting a summary set of tweets based on user interests. Their proposed model for personalized time-aware tweets summarization emphasizes the characteristics novelty, coverage, and diversity.

Touristic routes are summarized in an automatic, multi-modal approach by providing visual and textual descriptions [34]. It enables interactive city exploration in a prototype application using the multi-modal summaries.

Rudinac et al. [31] present a method to create visual summaries of geographic areas based on user-contributed images. Their approach values the selection of representative and diverse images from the dataset. Furthermore, automatic evaluation is presented using a novel protocol that eliminates the need for human annotations.

[10] https://research.fb.com/category/facebook-ai-research, accessed:18-07-2018

User preferences for visual summaries are researched using a large-scale crowdsourcing approach via the *Amazon Mechanical Turk (MTurk)*[11] platform [32].

Product review summarization is a field of ongoing research as well. With an abundance of reviews at hand readers are easily overwhelmed by the sheer number of expressed opinions, and thus extracting meaningful information becomes a challenging task. Existing solutions to this problem focus on different aspects in their summarization process. In feature-based summaries [17] product features are extracted and reviews are classified according to their sentiment towards these features. This approach aims to reduce opinion bias when only a subset of reviews is read. Similarly, this goal can also be achieved by ranking a set of questions about a product and matching reviews that are able to answer their respective question [19]. As other research has pointed out [2, 19] diversification is an important and sometimes overlooked characteristic of review summarization. Besides including representative reviews into the summary diverse sentiments and aspects should be included as well.

Bearing this in mind, the summarization problem can be approached with a wider scope taking into account the dominance of social multimedia [3]. The authors argue that summarization should be revisited in the light of modern multimedia to connect research areas which have been working in isolation from each other. In order to meet today's information needs of users tapping single media sources is not sufficient anymore.

3 PROBLEM STATEMENT

Visitors of beer review websites often receive recommendations for beers to try out. However, with an abundance of available recommendations to sample from the following question arises:

"Why should I drink this beer? What is it about?"

Given a specific recommendation, users are left behind with official product descriptions and a multitude of reviews to read through. Marketing descriptions rarely uncover the actual characteristics of a product as their goal is to instigate attention and demand. Assuming that reviewers talk about these characteristics revealing them is possible by mining the free-form text. Word embedding algorithms create vectorial models for a vocabulary by taking the co-occurrence statistics of a word into account.

"You shall know a word by the company it keeps." [11]

Building vectorial representations for the reviews of a product would allow performing numerical operations on them. The proposed *Beerlytics* system aims to exploit these representations for the creation of visual product stories. It targets consumers and visitors of beer review websites. Ideally, the system is capable of conveying the most representative and diverse review while also highlighting the essential characteristics of the product. The visual product stories should be both factual and stylistically pleasing. Creating such a system would satisfy the short attention span of the current generation by supporting cognition of relevant information [12]. Using different word embedding algorithms in an explorative and experimental nature for modeling products through their reviews is the primary concern of this work. Evaluating the system through user testing provides valuable insights into the preferences of users regarding visual product stories.

Research question:
To what extent can a product be summarized by a visual story based on associated metadata and user-contributed reviews? (RQ)

Further research questions:
How can user-contributed reviews support the generation of visual stories for a product? (SRQ1)
In which way is the quality of a visual story affected by using different word embedding techniques? (SRQ2)

4 METHOD

This section describes the methodological approach taken to realize a system for the generation of visual product stories and subsequent evaluation through user testing. It is composed of the following consecutive steps:

(1) Data collection from the *RateBeer* platform (Section 4.1)
(2) Data analysis through clustering algorithms (Section 4.2)
(3) System design of the frontend prototype (Section 4.3)
(4) Evaluation by the means of user testing (Section 4.4)

4.1 Data Collection

The process of *data collection* concerns itself with obtaining or building a dataset which will be used for all further work on the project. As *RateBeer* does not publish official datasets, third-party datasets are of limited availability, and the API only covers specific use cases *web scraping* is used to build a full useable dataset. Web scraping is an extraction technique which gathers structured data from HTML pages. Therefore, the data collection process consists of 3 consecutive steps:

(1) Collecting links for relevant pages (Section 4.1.1)
(2) Scraping pages for structured data (Section 4.1.2)
(3) Data persistence (Section 4.1.3)

4.1.1 Collecting Links. As the first step in the data collection pipeline lists of links will be created for relevant pages. These will be consumed by the scraping process in the next step. Relevant pages include data which is considered to be potentially useful for further processing. Since the goal is to build visual stories for the beer pages on *RateBeer* the original pages should be scraped to gather metadata about beers and their respective user-contributed reviews. The platform offers a way for users to contribute *places* where beers can be purchased. A place is essentially any entity with a locality which also distributes beers (e.g., bar, restaurant, brewery, etc.). Links for place pages will be collected as well in order to form a relationship between a beer and multiple localities.

The link scraping tool is implemented in *Python*[12] and uses the *Beautiful Soup*[13] open-source library for parsing HTML documents, traversing their structure, and finding link tags. Its associated source code is available on GitHub[14].

[11] https://www.mturk.com, accessed: 17-07-2018

[12] https://www.python.org/, accessed: 11-07-2018
[13] https://www.crummy.com/software/BeautifulSoup/, accessed: 11-07-2018
[14] https://github.com/SaschaVanEssen/Thesis/tree/master/Scraper

4.1.2 Data Extraction. Once the first step in the data collection pipeline is finished the lists of beer and places links are used by the second step for data extraction. The web scraping process is implemented in *JavaScript*[15] using the *Node.js*[16] [33] runtime and relies on *cheerio*[17] as an open-source library for parsing HTML into the Document Object Model (DOM)[18] which can be programmatically traversed and searched for data extraction purposes (Similar to the *Beautiful Soup* library used in 4.1.1). Respective source code can be found on GitHub[19].

It reads a list of links and transforms it into a queue of scraping tasks. These tasks can be processed concurrently as the program has to wait for responses of its HTTP requests to the *RateBeer* website. Depending on the type of page the scraper has 4 strategies to extract different information: (1) beer metadata, (2) beer reviews, (3) place metadata, and (4) relationships between a place and the beers it distributes. Each strategy works in a similar fashion by searching specific HTML tags using their name, IDs, CSS classes, attributes, and location in the DOM tree. As these elements are found their data is extracted and converted to the correct type if necessary (e.g., integer or decimal numbers instead of raw strings). All single extracted properties of a page are brought together in the strategy's respective model. This step transforms the unstructured, raw data on the page into structured, semantic data models. These models are passed to the third and final step of the data collection process which is responsible for persisting the models to the database (Section 4.1.3).

4.1.3 Data Persistence. Finally, the third and last step of the data collection process accepts structured data models as its input and stores them in the database. Persisting the data models to the database builds the dataset which is thereby made available for later usage. Access to the dataset is provided by the database management system (DBMS)[6] which acts as a sophisticated gateway to the underlying dataset.

Concerning the choice of DBMS, one has to consider whether a relational database management system (RDBMS)[7] or a non-relational (NoSQL) is more appropriate for the task at hand. The scientific nature of the project promotes a large-scale data collection approach which captures as many properties as possible. This approach is further justified due to the fact that 3 researchers in total are working with the dataset while having different targets in mind. It becomes problematic to judge whether certain data might be useful at a later point in time which speaks for a broader collection. Capturing a wide range of different properties with various types renders NoSQL databases as a favorable choice. These database systems do not impose a strict relational schema on their collections (respectively *tables* in relational databases). Despite the tendency towards a NoSQL storage system MySQL[20] has been chosen as the DBMS in question. This choice is backed by multiple factors:

- All researchers working with the dataset are already comfortable with using a relational SQL database.
- A relational schema encompassing all available properties on the scraped pages (Section 4.1.2) has been created with minimal complexity in mind (no foreign keys, no relations, minimal indexing, and lax type/length restrictions).
- Working with a relational database provides the structured query language (SQL) as a powerful mechanism to access and manipulate data which is capable to perform joined queries (aggregations) with low complexity.[21]

The final dataset schema can be inspected in the appendix (see Appendix C: MySQL Database).

As for the persistence process itself, the architecture is designed to mimic the data extraction (Section 4.1.2) system. Each *extraction* strategy has a corresponding *insertion* strategy which transforms the particular data model into one or multiple *INSERT*-statements and executes them[22]. In cases where multiple statements are required relations between models (e.g., place and beer) are saved into different tables.

Finally, the following entries are available in the dataset (limited to the main models):

- 106,088 beers (56 MB)
- 13,035 breweries (26 MB)
- 57,738 places (19.4 MB)
- 2,940,890 reviews (1.7 GB)
- 67,820 users (4.7 MB)

The collected data is preprocessed and used in the *data analysis* pipeline (Section 4.2) before it is consumed by the frontend prototype (Section 4.3).

4.2 Data Analysis

The *data analysis* process builds upon the result of the previous *data collection* (Section 4.1) process. Its main goal is to transform raw data into useful information. More specifically it is crucial for building a visual product story which summarizes the original data and provides engaging insights. Data analysis is composed of 3 sequential steps which are covered in greater detail in the following sections:

(1) Data cleaning and preprocessing (Section 4.2.1)
(2) Training Vector Space Models (VSM) using the 4 selected word embedding algorithms: word2vec, GloVe, fastText, and StarSpace (Section 4.2.2)
(3) Clustering of reviews and keyword extraction using the k-means [20] and k-nearest neighbors (k-NN) [28] algorithms (Section 4.2.3)

Due to the relatively large dataset (Section 4.1.3), which contains mostly textual data, all computation is to be executed on a more capable and powerful system than personal consumer devices. As a support in this matter, access has been granted to the Distributed ASCI Supercomputer (DAS) in its fourth generation [4]. Being designed for research areas with heavy demands for computational power and memory it is capable of handling the devised tasks. Furthermore, it is a system specifically built for running algorithms in parallel and on distributed nodes with an easy to use interface.

[15] https://www.w3.org/standards/webdesign/script, accessed: 11-07-2018
[16] https://nodejs.org/en/, accessed: 11-07-2018
[17] https://github.com/cheeriojs/cheerio, accessed: 11-07-2018
[18] https://www.w3.org/DOM/, accessed: 12-07-2018
[19] https://github.com/marc1404/msc-thesis/tree/master/beer-scraper
[20] https://www.mysql.com/, accessed: 15-07-2018
[21] https://dev.to/jignesh_simform/comparing-mongodb--mysql-bfa, accessed: 15-07-2018
[22] https://github.com/marc1404/msc-thesis/tree/master/beer-scraper/src/insert

4.2.1 Preprocessing. Data cleaning and preprocessing is an essential first measure between collection and analysis. It aims to reduce noise and improve the quality of the underlying dataset. Furthermore, the original data is transformed into a format which can be consumed by the actual analysis process. The main goal of this technique boils down to the creation of training text files for each beer containing the respective preprocessed reviews.

Prior to preprocessing the language of all reviews is identified. Multilingual word embeddings are out of scope for this research project which is the reason why only the subset of English reviews will be considered. Language detection is realized using the Python *py-googletrans*[23] library which relays calls to the official Google Translate API[24]. As this is a one-time procedure it has been implemented as a relatively simple script[25]. After detection succeeded it becomes apparent that 91.73% (2,716,159 entries) of all reviews are written in English. This justifies the usage of the English subset as the available corpus is still large enough for training vector space models.

With language identification in place, the actual cleaning and preprocessing tool can be used to generate the required training files. The transformation consists of 9 operations[26] working towards unification and removing unnecessary noise from the original reviews: (1) convert text to lowercase, (2) remove newline characters, (3) remove HTML tags, (4) remove digits, (5) remove text automatically added by *RateBeer*, (6) tokenization, (7) stopword removal, (8) stemming, and (9) convert British spelling variations to their American counterpart (e.g., colour to color).

While most operations are relatively straightforward step (6) tokenization, (7) stopword removal, and (8) stemming are of greater interest. Firstly, tokenization takes a character sequence (string) as its input and splits it up into discrete tokens [20]. There exists a range of different strategies regarding the splitting points. This project uses a word tokenizer which breaks up sequences on anything but alphabetic characters, digits, and underscores[27]. Secondly, common words with no semantic value are excluded in a process called stopword removal [20]. A list of 109 stopwords[28] is used for filtering the tokens. Finally, word variations are reduced through the stemming operating using the algorithm by M.F. Porter which was originally published in 1980 [29]. Review authors may use different words for the same semantic meaning (e.g., taste, tasteful, tastefulness). The suffixes of these words can be safely removed without altering the semantic meaning to one common root.

An example for the full preprocessing operation is provided here:

> **Original review:**
> *"Bottle. Poured a very dark brown with a medium off-white head. Fruity aroma, raisins the most. Big warm sweet alcohol taste with currants and raisins. Has a very smooth aftertaste"*
>
> **After preprocessing:**
> *"bottl pour dark brown medium off white head fruiti aroma raisin big warm sweet alcohol tast currant raisin smooth aftertast"*

It becomes clear that while the preprocessed version is better suited for machines it is less readable for humans. The word embedding algorithms (Section 4.2.2) work exclusively with the preprocessed version. Any output gathered from the trained vector space models only contains truncated words. As the results are to be used in a frontend prototype (Section 4.3) and subsequently shown to real users during user testing (Section 4.4) a mechanism is required which is able to transform stemmed works back into their original form. Naturally, such a mechanism does not exist due to the fact that stemming deliberately loses information. The relationship between a stemmed word and its origin counterparts is one-to-many. Therefore, a dictionary is created during the stemming operation which keeps original words for each stemmed token. This dictionary is saved to the database at the end of preprocessing (see Table 14).

4.2.2 Vector Space Models. Using the training text files from preprocessing (Section 4.2.1) vector space models (VSM) can be trained for each beer using the 4 selected word embedding algorithms: word2vec [23], GloVe [27], fastText [5], and StarSpace [35]. The selection of word embedding techniques is guided by the following criteria: covering all state of the art algorithms, different research organizations, covering different years, popularity, and paper citations. The training procedure and their parameters are shortly covered for each word embedding algorithm in the order of publication year.

word2vec: Training a word2vec model is realized using the Python *gensim*[29] library. The training script[30] reads the training file line by line, every line being a preprocessed review. Each line is split up on whitespace into an array. These arrays are appended to an aggregation array essentially creating a matrix of reviews. This matrix is used by the gensim library to train a word2vec model which is then saved to the disk. The parameters used for this training algorithm are as follows: vector dimensionality = 100, initial learning rate (alpha) = 0.025, sliding window = 5, continuous bag of words (CBOW) is used as the training algorithm, negative sampling is used as the loss function with 5 noise words, and 5 iterations (epochs) are performed on the corpus.

GloVe: Training of a GloVe model is possible using the 4 provided command-line interfaces (CLI)[31] of the tool. Firstly, a unigram count vocabulary is created from the training file with a minimum size of 10 and a maximum size of 100,000 entries. Secondly, word co-occurrence statistics are calculated using the training file and the previously generated vocabulary. Only the left side of the context is considered (symmetric = 0) with a window size of 10. Thirdly, the co-occurrence statistics file is shuffled and saved back into a new file. Finally, using the vocabulary and shuffled co-occurrence

[23] https://github.com/ssut/py-googletrans, accessed: 16-07-2018
[24] https://cloud.google.com/translate/docs/, accessed: 16-07-2018
[25] https://github.com/SaschaVanEssen/Thesis/blob/master/Preprocessing/detectlangs.py
[26] https://github.com/marc1404/msc-thesis/blob/master/embedding/src/cleanText.js
[27] https://github.com/NaturalNode/natural#tokenizers, accessed: 16-07-2018
[28] https://github.com/fergiemcdowall/stopword/blob/master/lib/stopwords_en.js, accessed: 16-07-2018
[29] https://radimrehurek.com/gensim/models/word2vec.html, accessed: 16-07-2018
[30] https://github.com/marc1404/msc-thesis/blob/master/clustering/train_word2vec.py
[31] https://github.com/marc1404/msc-thesis/blob/master/misc/train_glove.sh

file training of the GloVe model is triggered. Parameters for this training are as follows: vector dimensionality = 100, initial learning rate (alpha) = 0.75, and 25 iterations (epochs) are performed on the corpus.

fastText: In a similar fashion to GloVe training a fastText model is performed using a single CLI[32]. It creates a skip-gram model which considers character n-grams with a minimum size of 3 and a maximum size of 6. Further parameters used for training are: vector dimensionality = 100, minimal number of word occurrences = 5, initial learning rate (alpha) = 0.05, sliding window size = 5, negative sampling as the loss function with 5 noise words, and it performs 5 iterations (epochs) on the corpus.

StarSpace: Being the successor to fastText training a StarSpace model follows a similar pattern. Using the StarSpace CLI a vector model can be trained from a given corpus[33]. Parameters used for this training are as follows: vector dimensionality = 100, training mode = 5 for unsupervised training using only input, initial learning rate (alpha) = 0.01, sliding window size = 5, and 5 iterations (epochs) are performed on the corpus.

The trained vector space models for each beer and word embedding algorithm are used in the final step of the data analysis process (Section 4.2.3). Models created by GloVe and StarSpace are subject to a transformation due to their internal format being different from word2vec and fastText. This will be discussed in the next section.

4.2.3 Clustering. The final process of the data analysis pipeline uses *clustering* algorithms to compute information which is used for the realization of visual product stories in the frontend prototype (Section 4.3). Namely, 2 objectives are pursued:

- Clustering of beer reviews in order to identify a representative and diverse subset covering the main aspects.
- Keyword-based insight into the way how users write about different aspects of a beer.

Before the actual cluster computation, a standard interface is required for using the different vector space models (Section 4.2.2). As previously mentioned the models generated by GloVe and StarSpace differ slightly from the shared format used by word2vec and fastText. Furthermore, the gensim library offers the *KeyedVectors*[34] module which implements such a standard interface. It can directly load model output from word2vec and fastText which speaks for converting GloVe and StarSpace models to the shared format. Regarding the GloVe model gensim provides a built-in conversion script[35]. Due to the recent publication of StarSpace, a custom solution[36] converts the tab-separated values file (.tsv) generated by the tool into the original word2vec format. The differences are minimal: whitespaces separate values instead of tabs and the first line specifies both the vocabulary size and dimensionality of the vectors: *<vocabulary size><whitespace><vector size>*

Clustering of a beer's reviews is performed using the *k-means* algorithm. It is a standard clustering technique dividing the input data into k disjoint groups [20]. Reviews from the same cluster are similar to each other while being different to reviews from other clusters. The assumption that k-means allows selecting cluster centroids as representative (for their cluster) and yet diverse (centroids are different to each other) reviews justifies the use of this algorithm for this project. Reviews are clustered through the following process:

(1) Randomly select k reviews as the initial cluster centers
(2) Repeat:
(3) Calculate distance between each review and cluster center, assign reviews to their nearest cluster center
(4) For each review cluster re-calculate the cluster center
(5) until the center of all clusters remains unchanged

As k-means works by calculating distances between data points all textual reviews of a beer are converted into their vector representation[37]. This vector representation is used as input for the k-means algorithm implemented by the *scikit-learn* Python library [26]. The hyperparameter k equals 5 in this project to support the goal of selecting a subset of reviews that summarizes the product. Once the clustering algorithm converges pairwise distance calculation between cluster centers and reviews identifies the centroid review for each cluster[38]. The last step of review clustering is saving the computation result to the database (see Table 11).

Keyword-based insights are realized using the *k-nearest neighbors (k-NN)* algorithm. Provided with a test sample (i.e., a keyword) the algorithm calculates the Euclidean distance to all other data points [28]. The k nearest data points are selected and returned. The KeyedVectors gensim module already provides the calculation of k-NN through its interface. Using the official *RateBeer* rating aspects in combination with a hand-picked selection of frequent terms in reviews creates a list of 10 query keywords: (1) taste, (2) palate, (3) overall, (4) aroma, (5) appearance, (6) pour, (7) bottle, (8) price, (9) color, and (10) smell. Each model is probed for the 10 nearest neighbors for each query from the keyword list. list[39]. The resulting output is saved to the database (see Table 2).

4.3 System Design

The *Beerlytics* system is composed of multiple subsystems (Figure 1). These subsystems communicate with each other and provide downstream information which their dependent systems utilize for further work. There are 4 primary servers involved: (1) the *RateBeer* server, (2) a private server for data collection and database hosting (Section 4.1), (3) the computation server (Section 4.2), and (4) the frontend server which runs the API. In order to evaluate the proposed system's effectiveness through user testing (Section 4.4) a frontend prototype is implemented as a web application. The application runs inside a web browser on the user's device and communicates with the API using the HTTP protocol. First, the API is explained in section 4.3.1 which is followed by section 4.3.2 covering the frontend prototype more thoroughly.

[32] https://github.com/marc1404/msc-thesis/blob/master/misc/train_fasttext.sh
[33] https://github.com/marc1404/msc-thesis/blob/master/misc/train_starspace.sh
[34] https://radimrehurek.com/gensim/models/keyedvectors.html, accessed: 16-07-2018
[35] https://radimrehurek.com/gensim/scripts/glove2word2vec.html, accessed: 16-07-2018
[36] https://github.com/marc1404/msc-thesis/blob/master/clustering/starspace_tsv2txt.py, accessed: 16-07-2018
[37] https://github.com/marc1404/msc-thesis/blob/master/clustering/run_kmeans.py
[38] https://github.com/marc1404/msc-thesis/blob/master/clustering/kmeans.py
[39] https://github.com/marc1404/msc-thesis/blob/master/clustering/query_nn.py

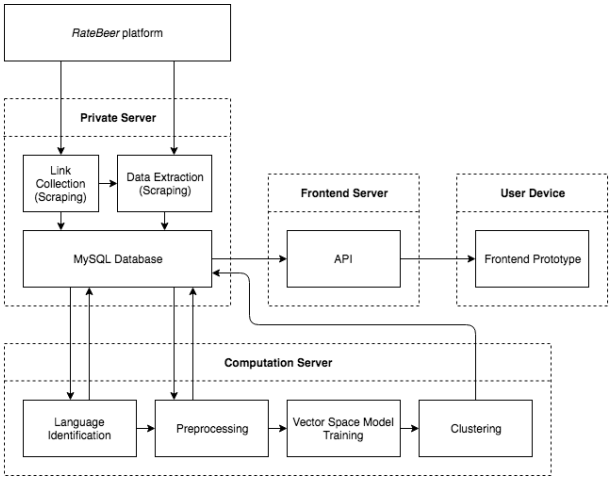


Figure 1: Beerlytics system architecture diagram

4.3.1 API. The API acts as the central communication gateway between the private database server and the frontend prototype (Figure 1). It is implemented in JavaScript using the Node.js [33] runtime and relies on *micro*[40] as a HTTP microservice framework. Each API endpoint serves a specific purpose for the frontend prototype (Table 1).

Route	*Purpose*
/search/(:query)	Most popular beers without query or search beer by name
/beer/:id/nn	Nearest neighbor keyword insights for a beer
/beer/:id/places	Locations of a beer's distribution places
/beer/:id/reviews	Reviews of a beer grouped by word embedding algorithms
/beer/:id	Metadata about a beer, e.g., name and description
/user/:id/locations	Locations of a user's reviewed beers

Table 1: API endpoints and purposes

The following paragraphs shortly discuss the available endpoints:

Search: Its route contains an optional query parameter. Depending on the presence of the query parameter the endpoint switches behavior. Provided no query parameter is present the API returns a list of the 16 most popular beers ordered by the count of their total ratings. Given a query, all beers are searched by their name and sorted according to the relevance towards the query. The search index is generated by the *Lunr.js*[41] full-text search library.

Beer Nearest Neighbors: This endpoint returns the nearest neighbor terms per word embedding and query term as computed in section 4.2 using the k-NN algorithm. The similarity for neighboring terms is a percentage decimal between 0 and 1. As the words in the database are in their stemmed form (see section 4.2.1) the token dictionary enables a translation back to a list of original words (Table 14).

Beer Places: Gathers geocoordinates (latitude, longitude) of all places selling a specific beer. These coordinates are lazily geocoded from the address information captured from *RateBeer*. Google provides a geocoding service[42] which is used to translate a place's address into geocoordinates if it has not been geocoded before.

Beer Reviews: The review cluster information computed during data analysis (Section 4.2) using the k-means algorithm is retrieved by this endpoint. It loads all review centroids per word embedding algorithm for a specific beer.

Beer Metadata: Returns static metadata about a beer (e.g., name, image, description) as scraped from its *RateBeer* page. It also computes the average rating using all reviews of a beer.

User Locations: Similar to the beer nearest neighbors endpoint as it also lazily geocodes addresses into geocoordinates. It gathers all beers which have been reviewed by a specific user and returns a list of their respective locations.

4.3.2 Frontend Prototype. The frontend prototype is a JavaScript application based on modern frameworks. *Nuxt.js*[43] for component-based application development and *Bulma*[44] as a CSS[45] framework which provides basic styling. It uses asynchronous HTTP requests to communicate with the API backend (Section 4.3.1). The various components of the frontend prototype fulfill their specific data requirements by sending requests to respective endpoints (Table 1).

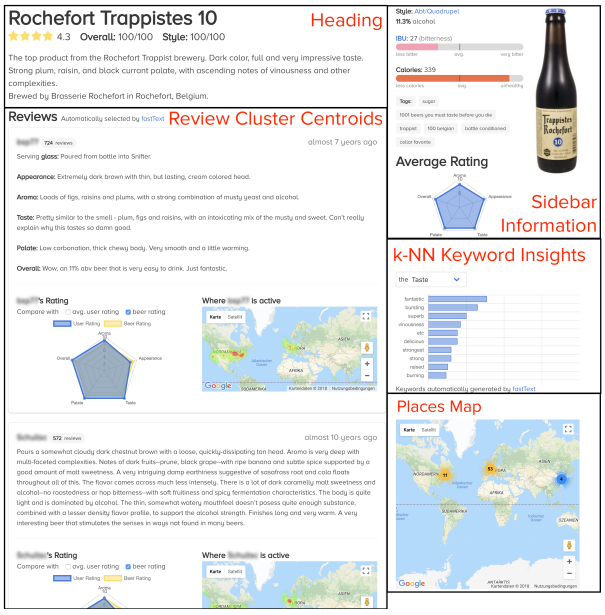


Figure 2: Components on the Beerlytics beer page

[40] https://github.com/zeit/micro, accessed: 17-07-2018
[41] https://github.com/olivernn/lunr.js, accessed: 17-07-2018
[42] https://developers.google.com/maps/documentation/javascript/geocoding, accessed: 17-07-2018
[43] https://nuxtjs.org/, accessed: 17-07-2018
[44] https://bulma.io/
[45] https://www.w3.org/Style/CSS/Overview.en.html, accessed: 17-07-2018

A user's journey begins on the index page (Figure 5) which features a prominent search bar and a list of beers. Beers are displayed using their name, amount of ratings, and a star rating from 0 to 5 stars. Typing into the search input automatically triggers search requests against the API's search endpoint. The list is automatically updated using the search results sorted by relevance.

Clicking on a beer entry redirects the user to the corresponding beer detail page (Figure 8). At the top of the page resides the heading component which renders static data as gathered from *RateBeer* (Figure 6). It includes the beer's name, previously mentioned star rating, and scores ranging from 0 to 100 for the categories *overall* and *style*. Below is the official description text provided by the brewery. The description follows a sentence stating the brewery name and location.

Continuing the page downwards reveals the reviews component which takes up the remaining space of the primary content column (Figure 3). This component is particularly interesting in the sense that it utilizes existing data from *RateBeer* (such as username, review text, and user rating) and merges it with dynamic data computed by data analysis. It restricts the set of visible reviews to the currently selected word embedding algorithm and is thereby only showing the cluster centroid reviews for this technique. In order to educate the user about this behavior the heading of the reviews component contains a disclaimer stating:

> *"Reviews automatically selected by <word embedding algorithm>"*

The bottom part of a review contains visual components working towards building a visual product story and supporting data comprehension of the user. Instead of displaying a textual description of the user rating a radar chart is used to visualize the strength of each aspect. Moreover, the radar chart allows overlaying the average rating of the current beer or the average rating of the review author (aggregated and averaged from all reviews authored by this user). Next to the radar chart is a heat map which indicates the locations of all beers reviewed by the same author. It essentially summarizes a user by visualizing their geospatial interest in relation to the reviewed beers.

The sidebar component provides factual information about the beer (Figure 7). Textual descriptions are used for the beer style and alcohol by volume (ABV) percentage with the beer style linking back to the corresponding *RateBeer* page. In contrast to that

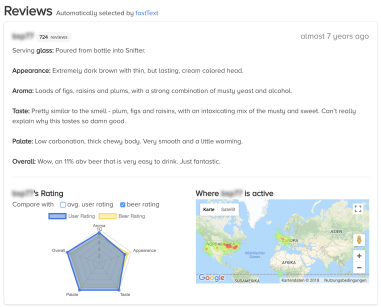


Figure 3: User review on Beerlytics

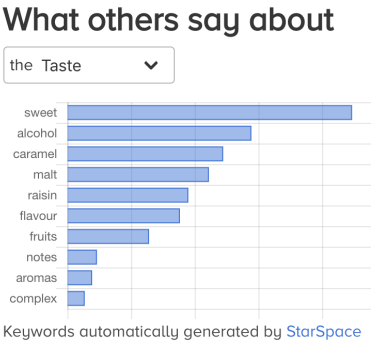


Figure 4: Nearest neighbor keywords on Beerlytics

stand the scales for international bitterness units (IBU) and calories per serving below. Displaying these measurements as simple values provide little to no use for users missing the appropriate background knowledge. Therefore, employing scales creates the necessary context to judge where these values lay in comparison to other beers. Labels below the scales annotate minimum, average, and maximum marks. Furthermore, color gradients of the scale bar support cognition by featuring different colors and varying intensity in correlation to the underlying value. The selected gradients are ensured to be safe for color blindness as they stem from the *ColorBrewer2*[46] tool which provides color gradients for sequential, diverging, and qualitative data based on research [15]. Users on *RateBeer* crowdsource sets of tags for beers which reappear as a simple list of formatted links in this prototype. Another radar chart concludes the bottom part of the sidebar information box which visualizes the aggregated, average rating of the current beer. In comparison to the radar chart in a review component, this chart does not offer interactivity besides hovering on data points to inspect their precise value.

Beneath the previously discussed sidebar information is the keywords component or nearest neighbors visualization (Figure 4). Similarly to the reviews component, it builds on the result of the k-NN data analysis computation (Section 4.2.3). The visualization itself is a horizontal bar chart with the y-axis displaying neighboring terms. Its x-axis does not have a legend annotating discrete values. Intentionally hiding the x-axis legend is based on the assumption that users are not interested in reading similarity values in percent (This assumption is picked up again in section 4.4.2). Suppressing this information reduces noise and clutter on the page and emphasizes the length of bars in relation to each other. The currently selected query keyword is selectable using a dropdown list above the bar chart. This list contains the query terms as discussed in section 4.2.3. Switching the query keyword or selected word embedding algorithm automatically adapts the displayed neighboring terms in the bar chart. In accordance with the disclaimer in the reviews component, another annotation below the visualization states:

> *"Keywords automatically generated by <word embedding algorithm>"*

[46] http://colorbrewer2.org, accessed: 05-07-2018

Finally, a map underneath the keyword chart summarizes the distribution places of a beer. Instead of displaying a marker at the location of each place it dynamically clusters markers in the vicinity of each other. Zooming into the map breaks up these clusters until only single markers remain. Clicking on a place marker opens an info-box in the map which renders name and an exemplary image of the place (if available). Further information about the place appears below the map and includes address, opening times, telephone number, and social media links.

4.4 Evaluation

The proposed *Beerlytics* system aims to summarize products (i.e., beers) by generating visual stories using primarily user-contributed reviews and product metadata for displaying static information. It automatically selects a subset of reviews aimed at satisfying the requirements for representativeness and diversity. Furthermore, it provides keyword-based insight into different aspects of the product by exploiting word cooccurrence characteristics in reviews. Evaluating such a system using automated methods is unfeasible. There exists no ground truth data about the quality of reviews or extracted keywords. Consequently, user testing with the frontend prototype (Section 4.3.2) intents to reveal opinions, improvements, suggestions, and criticism from real users regarding the different aspects of the system.

The following section 4.4.1 lays out the devised approach for user testing. Finally, section 4.4.2 concludes the system evaluation by discussing the user testing results.

4.4.1 Approach. The user study is planned as qualitative research although quantitative results might provide interesting insights as well (see also section 6). Convenience sampling selects participants who are easily accessible for the researcher [21]. These are usually subjects in close vicinity. However, the subject group is broadened by probing participants from different countries: Germany and the Netherlands. User testing takes place in 2 iterations with 6 participants in each group. Accordingly, a total of 12 participants took part in user testing the frontend prototype.

Research suggests that employing 3 to 5 evaluators in an evaluation is sufficient for gathering the main findings [25]. Indeed, theoretical saturation has been reached in each iteration which justifies equally sized groups of 6 participants.

The first iteration of testing is performed in Karlsruhe, Germany. Preliminary results are gathered between iterations which provide the basis for implementing improvements prior to the second round. The second iteration of the user study continues in Amsterdam, Netherlands.

Participants are in the age group between 20 and 30. Their backgrounds are as follows: 5x information studies students, 2x business informatics students, 1x design student & marketing, 2x software developer, and 2x business & product management. It should be noted that personal information is intentionally decoupled from Appendix B: User Tests to respect privacy.

The user testing procedure is semi-structured. Duration of a full test lies between 15 to 45 minutes. An initial briefing educates participants about the purpose, the concept of the original *RateBeer* page, and the basic functionality of the *Beerlytics* prototype. The briefing emphasizes that users should feel free to think out loud and comment on anything they find confusing, missing, poorly realized, or well done. Following up on the briefing is the primary part of the study where participants are free to use both *RateBeer* and *Beerlytics*. They receive guidance from the researcher only if necessary (e.g., components in the prototype might not have been explored yet). Feedback is not strongly directed in a predefined way. Solely the attention of users is directed to focus on the reviews and keywords components. After finishing the testing phase participants can provide further comments in an open-feedback round.

4.4.2 Results. The preliminary results gathered after the first iteration suggest a range of improvements. Implementation of these suggestions before the second round has the primary benefit that participants focus on different aspects instead of pointing out the same shortcomings again.

Firstly, adding search functionality to the index page provides users with the ability to search for other beers matching their interest (Figure 5). The search input was initially missing thereby restricting users to the fixed set of most rated beers.

On the beer detail page participants point out a missing mechanism for navigating back to the index page (apart from the browser's native navigation). The top left corner features an unobtrusive button for navigating back now.

Regarding the rating and scores visible in the heading component of a beer (Figure 6) users feel confused by the different rating schemes on the page (star rating 1-5, overall and style score 0-100, categorical rating 0-10). It is clarified that the varying schemes stem from the original *RateBeer* page but could indeed potentially be unified for better cognition. The maximum value of 100 is added next to the overall and style score which was absent before the second iteration.

Moving onto the reviews component (Figure 3) a trivial improvement is the addition of a "Reviews" heading which improves the separation between the heading and reviews section. The review count next to the author's name lacked a clear label as well. These seemingly small additions were wished for by nearly all participants of the first testing round. Thus, clarity is valued over minimalistic design in these cases. The rating radar chart in a user review initially only displayed the static user rating. Multiple test subjects expressed their wish of comparing the user rating with the average beer rating and the average rating of the review author in the same chart. This suggestion is realized while also adding a radar chart for the average beer rating in the sidebar component (Figure 7). Another small improvement in the review component is the rephrased wording for the rating radar chart and user heatmap. The relationship between these visualizations to the review author was not clear before using possessive phrasing in their heading labels.

Focussing on the sidebar component (Figure 7) most participants in the first iteration suggested to employ color gradients for the IBU and calories scale. These gradients have been implemented while also adding a descriptive legend below the scales.

Lastly, disclaimer statements are added in the heading of the reviews component (Figure 3) and below the keywords component (Figure 4). Before the disclaimers where present users could easily feel perplexed by the quality of selected reviews or extracted

keywords. Placing reminders helps to create awareness about the utilization of automated, unsupervised algorithms.

After discussing the improvements spurred by the first iteration, the results are used to revisit the research questions from section 3. The sub-questions *SRQ1* and *SRQ2* lead the way with a subsequent final look at the primary research question *RQ*.

SRQ1: The generation of visual stories for a product can be supported by using a word embedding algorithm to process user-contributed reviews. Training vector space models from product reviews allow performing clustering and keyword extraction.

Cluster centroids point to the most representative review of their cluster while different clusters ensure diversity. Participants confirm that showing only a subset of reviews is beneficial for the product story (*"preselection of reviews is a good idea"*, *"less is more"*). Suggestions to improve this component include: helping to judge the trustworthiness of a review, working on the explainability why reviews are selected, and finding a way to pick the most neutral reviewers. Furthermore, the structural quality of reviews should be considered as well (*"There's a sweet spot for length, neither too short nor too long"*, *"formatting matters, should be easy to read"*). An interesting comment is also to *"recognize average and deviating reviews, is it controversial?"*.

The keywords component is perceived as an intriguing mechanism for exploring reviews based on different query keywords (*"useful and interesting"*, *"fun to play around with and explore"*). However, a majority of participants criticize the quality of extracted keywords. Noise should be reduced and duplicates filtered out.

SRQ2: The quality of a visual product story varies between different word embedding algorithms. Naturally, the vertical page size is affected by the length of the selected reviews. Reviews selected by GloVe tend to be only a single or a few sentences long (*"favors short, one-line reviews"*). Conversely, StarSpace seems to pick up the internal text structure displaying reviews ranging from well-structured stories, over long consecutive text blocks, to short one-line texts. The word2vec and fastText algorithms select reviews of medium to short length.

Furthermore, the quality of the terms visible in the keyword chart affects the overall sentiment towards the product story. Multiple participants mention StarSpace as the preferred embedding regarding the quality of reviews and keywords (*"StarSpace keywords are rather good"*, *"seems to offer more interesting ... and relevant keywords"*). In case of the keyword component displaying nonsensical, duplicate, or irrelevant terms users immediately pointed out degraded usefulness.

RQ: A product can be partially summarized by a visual story when information is condensed and displayed using appropriate visualizations, only the relevant subset of reviews is shown, context is provided for standalone values, and the product can be explored from different aspect angles. However, the summarization appearance of the *Beerlytics* prototype leaves room for improvement. While participants unanimously prefer the prototype over the original *RateBeer* page, it does not meet the expectations of a summary in terms of perception. Participants state that the size of the page and the number of visible elements lean towards a certain degree of clutter which could be kept more succinct (*"reduce amount of keywords to reduce clutter"*, *"collapse lower part of reviews to reduce amount of visible charts and maps"*, *"requires a lot of scrolling"*).

Displaying user ratings in a radar chart with the ability to compare them with the average beer or user rating is an appropriate choice (*"... makes it easier to understand"*, *"being able to compare ratings in radar chart is useful"*). Moreover, providing context through the use of scales and color gradients for IBU and calories measurements is a welcome improvement (*"good addition"*, *"... context is very useful"*, *"allows to avoid very bitter beers"*).

5 DISCUSSION

This section discusses the shortcomings of the proposed *Beerlytics* system. Decisions made during the design, development, and evaluation of the project can impact the effectiveness of the system and the quality of the results. The experimental nature of the project hints towards a critical interpretation of the results as well.

Firstly, the dataset stems exclusively from the *RateBeer* platform. The website features specific characteristics and restrictions which its data inevitably mirrors explicitly and implicitly. Namely, the different rating schemes (weighted average, overall and style score, categorical rating) have an impact on the rating data but also the way users write about beers in their reviews. Moreover, rating assistance is displayed while authoring a review on *RateBeer*. While this assistance is undoubtedly helpful for users, it offers predefined descriptors in each rating category which shape the review text. It would be interesting to consider other beer rating platforms such as *BeerAdvocate*[47] and *Untappd*[48]. Perhaps it could be possible to create an aggregated dataset from a multitude of platforms and unifying the data from each source into a shared scheme.

Secondly, the current system solely processes the textual part of user reviews. Considering further aspects of reviews before or after the word embedding training is a sensible option. Furthermore, StarSpace can embed entities of varying types in a shared vectorial space [35]. Using StarSpace other aspects could already be included during training of the vector space model. Participants of the user testing also suggest considering the bias of review authors and the age of reviews (*"consider bias of reviewers, try to pick the most neutral ones"*, *"age of a review matters, old ones are not as interesting"*).

Thirdly, the k-means algorithm used for clustering reviews is well proven and commonly used in data analysis [20]. However, the choice of clustering algorithm has a fundamental impact on the computed clusters. Other techniques should be reviewed and tested for their performance in clustering reviews represented in vectorial space. It may be that different algorithms produce better results than the currently employed algorithm. Additionally, the current system does not satisfy the requirement of explainability. It is not able to explain the reasons for including a review in the final subset. Users feel the need to understand why a review is automatically selected (*"Is it relevant for me?"*).

Finally, using convenience sampling to gather participants of the user study can be criticized as well. It limits the subject group to a particular subgroup from the researcher's environment. This subgroup might not be representative of a potential real user group.

[47] https://www.beeradvocate.com, accessed: 17-07-2018
[48] https://untappd.com, accessed: 17-07-2018

Findings are therefore signals for further research and development directions. Furthermore, combining qualitative and quantitative research could have produced more valuable insights. It would also allow comparing the different word embedding algorithms using quantitative data (see also section 6). Increasing the test group size is a requirement for valid quantitative research.

6 CONCLUSION & FUTURE WORK

The proposed *Beerlytics* system succeeds in generating visual stories of beers while exploiting vector space models of their reviews trained by different word embedding algorithms.

In order to build its dataset, the system scrapes relevant links from *RateBeer* and scrapes their corresponding pages thereafter. The data collection process persists the data to a MySQL database that is shared by all researchers working on the larger *Beerlytics* platform.

Subsequently, data analysis transforms raw data into valuable information through multiple steps. Raw data is cleaned and preprocessed before being fed into the different word embedding algorithms. This process trains vector space models for each beer and algorithm in use. Clustering techniques are applied the resulting models which identify representative and diverse reviews. Additionally, keyword extraction provides insights into a beer product from the angle of different aspects.

Finally, the frontend prototype turns the output from the computation process into visual product stories. It communicates with the API to satisfy its data requirements. Evaluation of the frontend prototype in 2 iterations with 6 participants in each round tests the effectiveness of the systems and gathers useful feedback.

Considering all findings, participants prefer *Beerlytics* over the original *RateBeer* platform. However, the system leaves room for improvement in regards to the summarization aspect of the page and ensuring a high standard for the quality of extracted keywords.

Future work includes the implementation of suggested improvements, exploring the problem using quantitative methods, and potential benefits for interactive multimodal learning scenarios.

Firstly, multiple participants in both user testing rounds recommend a sentiment scoring for extracted keywords. As they currently stand, it is challenging to judge whether a positive or negative context surrounds a keyword. Realization requires localizing the keywords in their original review and performing sentiment analysis on the context. Given the localization of keywords, they could also be linked to the review component and highlighted in the text.

Secondly, applying the same approach for training models from beer reviews to breweries and also users is a future possibility too. The models for breweries and users would allow their exploration and potentially comparisons between them.

Thirdly, a highly requested functionality during user testing is regional filtering of the dataset (e.g., restricting the data to the Netherlands only). This regional filtering would require computation on varying regional levels.

Fourthly, quantitative and automated methods could potentially be used for the evaluation. Currently, no ground truth data exists about the quality of reviews in the dataset. Crowdsourcing platforms like *Amazon Mechanical Turk (MTurk)*[49] offer human intelligence through programmable interfaces. Utilization of human intelligence could establish the necessary ground truth data by creating human intelligence tasks (HITs) judging the quality of reviews.

Finally, the compact representations generated by the system could benefit interactive learning scenarios where quick relevance judgment of an item is necessary [37, 38]. Representation of an item should be as condensed as possible while most of the original information content is preserved.

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude towards all people that helped to shape, progress, and support the successful completion of this thesis project. Dr. Stevan Rudinac for his invaluable supervision over the course of the past months. Gosia Migut for being the second reader of this thesis paper. Dr. Frank Nack for his dedication towards students and the constant guidance as the programme director of Information Studies. Sascha van Essen for the excellent collaboration on the *Beerlytics* system. Katharina Lott for assisting in brainstorming and unwavering support throughout the writing process. Finally, all participants of the user testing for taking their time and providing valuable insights.

REFERENCES

[1] Charu C Aggarwal and ChengXiang Zhai. 2012. *A Survey of Text Clustering Algorithms.* Springer US, Boston, MA, 77–128. https://doi.org/10.1007/978-1-4614-3223-4_4

[2] Mohammed Al-Dhelaan and Abeer Al-Suhaim. 2017. Sentiment Diversification for Short Review Summarization. In *Proceedings of the International Conference on Web Intelligence (WI '17).* ACM, New York, NY, USA, 723–729. https://doi.org/10.1145/3106426.3106525

[3] Stevan Rudinac B, Tat-seng Chua, Nicolas Diaz-ferreyra, Gerald Friedland, Tatjana Gornostaja, Benoit Huet, Rianne Kaptein, and Krister Lind. 2018. MultiMedia Modeling. *International Conference on Multimedia Modeling* 10704 (2018), 632–644. https://doi.org/10.1007/978-3-319-73603-7

[4] Henri Bal, Dick Epema, Cees De Laat, Rob Van Nieuwpoort, John Romein, Frank Seinstra, Cees Snoek, and Harry Wijshoff. 2016. A Medium-Scale Distributed System for Computer Science Research: Infrastructure for the Long Term. *Computer* 49, 5 (2016), 54–63. https://doi.org/10.1109/MC.2016.127

[5] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2016. Enriching Word Vectors with Subword Information. *CoRR* abs/1607.0 (2016). arXiv:1607.04606 http://arxiv.org/abs/1607.04606

[6] P. Christensson. 2006. DBMS (Database Management System) Definition. (2006). https://techterms.com/definition/dbms

[7] P. Christensson. 2017. RDBMS (Relational Database Management System) Definition. (2017). https://techterms.com/definition/rdbms

[8] Alexis Conneau, Guillaume Lample, Marc'Aurelio Ranzato, Ludovic Denoyer, and Hervé Jégou. 2017. Word Translation Without Parallel Data. *CoRR* abs/1710.0 (2017). arXiv:1710.04087 http://arxiv.org/abs/1710.04087

[9] Dipanjan Das and André F.T. Martins. 2007. A Survey on Automatic Text Summarization. *Eighth ACIS International Conference on Software Engineering Artificial Intelligence Networking and ParallelDistributed Computing SNPD 2007* 4 (2007), 574–578. https://doi.org/10.1016/B0-08-044854-2/00957-3 arXiv:7182216

[10] Scott Deerwester, Susan T. Dumais, George W Furnas, Thomas K Landauer, and Richard Harshman. 1990. Indexing by latent semantic analysis. *Journal of the American Society for Information Science* 41, 6 (1990), 391–407. https://doi.org/10.1002/(SICI)1097-4571(199009)41:6<391::AID-ASI1>3.0.CO;2-9 arXiv:arXiv:1011.1669v3

[11] John R Firth. 1957. A synopsis of linguistic theory, 1930-1955. *Studies in linguistic analysis* (1957).

[12] Alyson Gausby and Others. 2015. Attention Spans. Consumer Insights. *Microsoft Canada* (2015).

[13] Yoav Goldberg and Omer Levy. 2014. word2vec Explained: deriving Mikolov et al.'s negative-sampling word-embedding method. *CoRR* abs/1402.3 (2014). arXiv:1402.3722 http://arxiv.org/abs/1402.3722

[49] https://www.mturk.com, accessed: 17-07-2018

[14] Zellig S Harris. 1954. Distributional Structure. *WORD* 10, 2-3 (1954), 146–162. https://doi.org/10.1080/00437956.1954.11659520

[15] Mark Harrower and Cynthia A. Brewer. 2011. ColorBrewer.org: An Online Tool for Selecting Colour Schemes for Maps. *The Map Reader: Theories of Mapping Practice and Cartographic Representation* 7041 (2011), 261–268. https://doi.org/10.1002/9780470979587.ch34

[16] Minqing Hu and Bing Liu. 2004. Mining and Summarizing Customer Reviews. In *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '04)*. ACM, New York, NY, USA, 168–177. https://doi.org/10.1145/1014052.1014073

[17] Minqing Hu and Bing Liu. 2004. Mining opinion features in customer reviews. In *AAAI*, Vol. 4. 755–760.

[18] Omer Levy, Yoav Goldberg, and Ido Dagan. 2015. Improving Distributional Similarity with Lessons Learned from Word Embeddings. *Transactions of the Association for Computational Linguistics* 3 (2015), 211–225. https://doi.org/10.1186/1472-6947-15-S2-S2 arXiv:1103.0398

[19] Mengwen Liu, Yi Fang, Alexander G Choulos, Dae Hoon Park, and Xiaohua Hu. 2017. Product review summarization through question retrieval and diversification. *Information Retrieval Journal* 20, 6 (dec 2017), 575–605. https://doi.org/10.1007/s10791-017-9311-0

[20] Christopher D. Manning, Prabhakar Ragahvan, and Hinrich Schutze. 2009. An Introduction to Information Retrieval. *Information Retrieval* c (2009), 1–18. https://doi.org/10.1109/LPT.2009.2020494 arXiv:0521865719 9780521865715

[21] Martin N Marshall. 1996. Sampling for qualitative research Sample size. *Family Practice* 13, 6 (1996), 522–525. https://doi.org/10.1093/fampra/13.6.522

[22] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Distributed Representations of Words and Phrases and their Compositionality. *Advances in Neural Information Processing Systems 26 (NIPS 2013)* 26 (2013), 1–9. https://doi.org/10.1162/jmlr.2003.3.4-5.951 arXiv:1310.4546

[23] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient Estimation of Word Representations in Vector Space. (2013), 1–12. https://doi.org/10.1162/153244303322533223 arXiv:1301.3781

[24] Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. 2013. Linguistic regularities in continuous space word representations. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. 746–751.

[25] Jakob Nielsen and Rolf Molich. 1990. Heuristic Evaluation of User Interfaces. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '90)*. ACM, New York, NY, USA, 249–256. https://doi.org/10.1145/97243.97281

[26] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. 2012. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* 12 (2012), 2825–2830. https://doi.org/10.1007/s13398-014-0173-7.2 arXiv:1201.0490

[27] Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*. 1532–1543.

[28] Leif Peterson. 2009. K-nearest neighbor. *Scholarpedia* 4, 2 (2009), 1883. https://doi.org/10.4249/scholarpedia.1883

[29] M. F. Porter. 1980. An algorithm for suffix stripping. *Program* 14, 3 (1980), 130–137. https://doi.org/10.1108/eb046814 arXiv:http://dx.doi.org/10.1108/BIJ-10-2012-0068

[30] Zhaochun Ren, Shangsong Liang, Edgar Meij, and Maarten de Rijke. 2013. Personalized Time-aware Tweets Summarization. In *Proceedings of the 36th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '13)*. ACM, New York, NY, USA, 513–522. https://doi.org/10.1145/2484028.2484052

[31] Stevan Rudinac, Alan Hanjalic, and Martha Larson. 2013. Generating visual summaries of geographic areas using community-contributed images. *IEEE Transactions on Multimedia* 15, 4 (2013), 921–932. https://doi.org/10.1109/TMM.2013.2237896

[32] Stevan Rudinac, Martha Larson, and Alan Hanjalic. 2013. Learning crowdsourced user preferences for visual summarization of image collections. *IEEE Transactions on Multimedia* 15, 6 (2013), 1231–1243. https://doi.org/10.1109/TMM.2013.2261481

[33] Stefan Tilkov and Steve Vinoski. 2010. Node.js: Using JavaScript to build high-performance network programs. *IEEE Internet Computing* 14, 6 (2010), 80–83. https://doi.org/10.1109/MIC.2010.145

[34] Jorrit van den Berg, Stevan Rudinac, and Marcel Worring. 2016. Scenemash: Multimodal Route Summarization for City Exploration. In *Advances in Information Retrieval*, Nicola Ferro, Fabio Crestani, Marie-Francine Moens, Josiane Mothe, Fabrizio Silvestri, Giorgio Maria Di Nunzio, Claudia Hauff, and Gianmaria Silvello (Eds.). Springer International Publishing, Cham, 833–836.

[35] Ledell Wu, Adam Fisch, Sumit Chopra, Keith Adams, Antoine Bordes, and Jason Weston. 2017. StarSpace: Embed All The Things! *CoRR* abs/1709.0 (2017). arXiv:1709.03856 http://arxiv.org/abs/1709.03856

[36] Tom Young, Devamanyu Hazarika, Soujanya Poria, and Erik Cambria. 2017. Recent Trends in Deep Learning Based Natural Language Processing. *CoRR* abs/1708.0 (2017). arXiv:1708.02709 http://arxiv.org/abs/1708.02709

[37] J Zahálka, S Rudinac, B Jónsson, D Koelma, and M Worring. 2018. Blackthorn: Large-Scale Interactive Multimodal Learning. *IEEE Transactions on Multimedia* 20, 3 (mar 2018), 687–698. https://doi.org/10.1109/TMM.2017.2755986

[38] J Zahálka, S Rudinac, and M Worring. 2015. Interactive Multimodal Learning for Venue Recommendation. *IEEE Transactions on Multimedia* 17, 12 (dec 2015), 2235–2244. https://doi.org/10.1109/TMM.2015.2480007

APPENDIX

Appendix A: GitHub Repository

Development of the project has taken place in a *Git* repository on *GitHub*. All related source code of this work can be found inside the repository. It is available at:

https://github.com/marc1404/msc-thesis

Appendix B: User Tests

I1.T1: 03. July 2018 10:40; Karlsruhe, Germany

- Show total count of reviews for this beer
- Regional search function
- Search for keyword
- Reviews should provide info why a certain beer is bad? How can I judge that it is relevant for me?
- Add a heatmap visualizing where a user is active
- Show regional differences, provide a way to compare
- Preselection of reviews is good
- Would be interesting to differentiate between positive and negative reviews
- Favored embedding: StarSpace

I1.T2: 03. July 2018 11:10; Karlsruhe, Germany

- Make it more clear that heatmap is about a user's reviews
- Show more information about a user, e.g., average rating
- Add label to user review count, number alone doesn't suffice
- Judging trustworthiness of reviews: Is it relevant for me? Is it specific to a region?
- Consider bias of reviewers
- Try to pick neutral reviewers
- User's reviews heatmap is a good addition
- IBU and calories scale should use a color gradient
- Consider deviation of a user's rating
- Build user profiles
- Maybe some users are being paid by companies
- Radar chart for visualizing ratings is helpful
- Link keywords with review text, e.g., highlighting
- Calculate a positivity/negativity score for keywords, in which semantic context are they used?

I1.T3: 03. July 2018 11:30; Karlsruhe, Germany

- "Cool design"
- Would be interesting to compare different beers using the radar chart
- Overall heatmap for a beer's reviews
- It should be possible to compare the user rating with the beer and average user rating in the radar chart
- Not directly clear that reviews are from users
- Should the page support mobile devices? (responsive design)
- Create summary of breweries and compare them

I1.T4: 03. July 2018 15:55; Karlsruhe, Germany

- Consider age of a review
- Visualize development over time
- List ingredients of a beer
- Automatic preselection of reviews is good: "less is more"
- Show maximum for overall and style score
- Quality of keywords could be improved
- Places map is useful
- Link to official beer homepage, could go as far as to include referral links
- Showing user review count and user heatmap is useful to gauge trustworthiness
- Tag quality could be improved
- Formatting of review matters, should be easy to read, should be structured
- Filter low quality reviews

I1.T5: 03. July 2018 16:25; Karlsruhe, Germany

- User review heatmap not necessarily useful, what does it tell me?
- Implement positivity/negativity score for keywords
- There's a sweet spot for review length, neither too short nor too long
- Recognize average and deviating reviews, "Is it a controversial review?"
- Highlight keywords in reviews

I1.T6: 03. July 2018 17:00; Karlsruhe, Germany

- Add a heading above reviews to label them
- Age of reviews matters, old ones are not that interesting
- Length of review matters, neither too short nor too long
- Avoid text blocks
- Places map is very useful
- Keywords chart is interesting
- Calories scale should have a color gradient
- Using a radar chart for visualizing ratings makes it easier to understand
- Show rating of review as a star scale too
- Drinkability should be a keyword
- "Super good idea"
- Rating count of user is important
- User review heatmap is somewhat useful
- Include more media, e.g., higher quality images and videos
- Provide context for ranking, how popular is this beer really?

I2.T1: 09. July 2018 13:50; Amsterdam, The Netherlands

- Keywords need to be refined, StarSpace keywords are rather good
- Using "unhealthy" as maximum for calories scale is a tad aggressive
- IBU scale is useful
- Recognize and filter duplicate keywords
- Keywords chart is useful and interesting
- Some keywords are weird, could be improved
- Default zoom of user's reviews heatmap should be optimized
- Heatmap requires a scale
- Rating radar chart is intuitive to understand

- It's interesting to see differences between user rating and average rating for a beer
- Showing short reviews is good too
- Rating scales should be more consistent
- Show count for how many reviews are visible
- Showing full review blocks is fine since only a subset of the total reviews is shown
- Age of review should be considered
- Places map helps to get and idea where a beer is based
- "Neat!"
- Current layout requires a lot of scrolling
- Calories should be more specified, "Calories per what?"

I2.T2: 09. July 2018 14:15; Amsterdam, The Netherlands

- Layout is good, division between reviews and quick facts
- Keywords chart requires a scale
- Places map is useful
- Provide a search function for places map
- Fix tag links with spaces
- "Isn't every beer unhealthy?"
- Calories scale was shortly confusing
- Being able to compare ratings in radar chart is good
- Showing rating count of users is important
- Show similar beers
- Might be useful to show since when a user has been active
- Design looks nice

I2.T3: 09. July 2018 14:30; Amsterdam, The Netherlands

- It's intuitive to recognize the official description
- Keywords chart should have a scale, hard to understand otherwise
- Using "unhealthy" as maximum of calories scale is questionable
- Not directly clear what is meant by average user rating in radar chart
- Consider age of reviews, "Maybe the recipe changed or taste"
- Link to user profile page, it's interesting to know more about a reviewer
- Keywords chart is very interesting, quality could be improved

I2.T4: 10. July 2018 10:45; Amsterdam, The Netherlands

- Show list of ingredients, important for people with allergies
- Image of beer seems off in the layout, not neatly aligned
- User heatmap is interesting
- Using a radar chart for visualizing ratings is a good choice for comparisons
- Quality of keywords could be improved
- Show star rating scale for reviews too, parsing radar charts takes longer
- Relying on hover for showing detailed values in charts can be dangerous since it's easy to miss
- StarSpace seems to offer more interesting reviews and relevant keywords
- Keywords usually roughly aligned between the different algorithms
- Separate image box with different images, e.g., beer in a glass
- Bitterness scale is very useful, allows to avoid very bitter beers
- Information about bottle and keg availability could be included
- Showing similar beers would be a nice addition
- Colors in rating radar chart should be fixed, seemingly random change is confusing
- Calories scale, including minimum and maximum labels, is very good
- Official description could be more structured
- User heatmap could use a different label since it's based on the beers a user has written reviews about
- Info button next to rating radar chart to explain functionality if required
- Backlinking to RateBeer is a good idea
- Places map is interesting
- Future iterations could include buttons for editing or writing a review
- Ratings should be even more uniform
- User heatmap is somewhat useful but doesn't add anything for understanding the beer
- Specify calories per what

I2.T5: 10. July 2018 13:00; Amsterdam, The Netherlands

- Style score is confusing, could also be number of a beer style
- Keywords is a good idea but too small and a word cloud might be a better visualization
- Unclear what user heatmap is based on, label should be changed and a regular map with pins might suffice
- Not clear what's meant by average user rating in radar chart comparison
- Same label should be used for average rating chart and beer rating in comparison
- "Bottle" keyword renders rather useless results
- Reduce amount of keywords in order to reduce clutter
- Star rating should show inactive stars
- IBU scale is a good addition
- Use hover/click info for IBU instead of link
- Context for calories is very useful
- No immediate difference visible between embeddings
- Quick facts on the side are rather small and easy to miss
- Radar charts could be bigger in order to improve readability
- Future iterations could allow to show more/all reviews and different sorting modes
- GloVe seems to favor short, one-line reviews
- Highlighting keywords in reviews helps
- Quick facts should be moved to the main block
- Small Google Maps are usually not very useful, zoom on country level, help user to navigate

I2.T6: 13. July 2018 15:00; Amsterdam, The Netherlands

- Beerlytics has a better design compared to RateBeer
- Minimalistic design is nice and easy on the eyes
- Reviews should be more pronounced somehow, maybe a light-grey backdrop
- Should be clearer that reviews are separate from sidebar info on the right
- Dark-grey as font color is a good choice
- Possibly move image to right side of the official description text
- Consider showing less keywords, e.g., 5 or 6 instead of 10
- Reduce noise in keywords
- Keyword insights are interesting and useful, fun to play around with and explore
- Make it possible to change amount of visible keywords
- Collapse lower part of reviews in order to reduce amount of visible charts and maps
- Fix colors in rating radar chart
- Tags are rather uninteresting, could be moved below places map
- Beer style and alcohol by volume should be more visible and better aligned with heading
- Reviews seem to be pretty old
- "Unhealthy" label in calories scale should be changed to, e.g., "high-calories"

Appendix C: MySQL Database

Field	*Type*	*Length*
id	INT	11
beer_id	INT	10
embedding	VARCHAR	10
query	VARCHAR	50
neighbor	VARCHAR	50
similarity	DECIMAL	10,10

Table 2: Schema for table beer_nn

Field	*Type*	*Length*
id	INT	11
beer_id	INT	10
place_id	INT	10

Table 3: Schema for table beer_places

Field	*Type*	*Length*
id	INT	11
name	VARCHAR	100
url	VARCHAR	200

Table 4: Schema for table beer_styles

Field	*Type*	*Length*
id	INT	11
beer_id	INT	10
tag_id	INT	10

Table 5: Schema for table beer_tags

Field	*Type*	*Length*
id	INT	11
url	VARCHAR	200

Table 6: Schema for table beer_urls_global

Field	*Type*	*Length*
id	INT	11
url	VARCHAR	200

Table 7: Schema for table beer_urls_nl

Field	*Type*	*Length*
id	INT	11
name	VARCHAR	100
brewery_id	INT	10
style_id	INT	10
url	VARCHAR	200
location	VARCHAR	50
image	VARCHAR	200
description	VARCHAR	1000
overall	INT	10
style	INT	10
weighted_verage	FLOAT	
total_ratings	INT	10
ibu	INT	10
calories	INT	10
abv	FLOAT	
scraped	TINYINT	1
latitude	DECIMAL	10,8
longitude	DECIMAL	11,8
processed	TINYINT	1

Table 8: Schema for table beers

Field	*Type*	*Length*
id	INT	11
name	VARCHAR	100
url	VARCHAR	200

Table 9: Schema for table breweries

Field	*Type*	*Length*
id	INT	11
slug	VARCHAR	100
name	VARCHAR	100
type	VARCHAR	15
street	VARCHAR	50
locality	VARCHAR	50
region	VARCHAR	50
country	VARCHAR	50
postal_code	VARCHAR	10
rating_count	INT	10
rating_value	FLOAT	
facebook_url	VARCHAR	200
twitter_url	VARCHAR	200
website_url	VARCHAR	200
telephone	VARCHAR	30
opening_times	VARCHAR	100
image_url	VARCHAR	200
scraped_beers	TINYINT	1
latitude	DECIMAL	10,8
longitude	DECIMAL	11,8

Table 10: Schema for table places

Field	*Type*	*Length*
id	INT	11
review_id	INT	10
embedding	VARCHAR	10
cluster	TINYINT	3
is_centroid	TINYINT	1

Table 11: Schema for table review_clusters

Field	*Type*	*Length*
id	INT	11
beer_id	INT	10
user_id	INT	10
date	DATE	
location	VARCHAR	150
text	VARCHAR	10100
score	FLOAT	
aroma	INT	10
appearance	INT	10
taste	INT	11
palate	INT	11
overall	INT	11
language	VARCHAR	10

Table 12: Schema for table reviews

Field	*Type*	*Length*
id	INT	11
name	VARCHAR	50

Table 13: Schema for table tags

Field	*Type*	*Length*
id	INT	11
original_token	VARCHAR	100
cleaned_token	VARCHAR	100

Table 14: Schema for table token_dictionary

Field	*Type*	*Length*
id	INT	11
name	VARCHAR	100
total_ratings	INT	10

Table 15: Schema for table users

Appendix D: Frontend Prototype

All screenshots are captured on the 16th of July 2018.

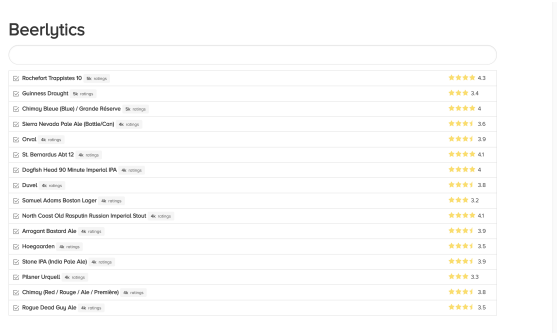


Figure 5: Beerlytics index page

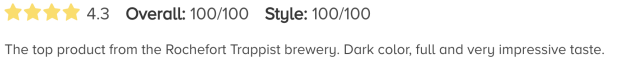


Figure 6: Heading of a beer on Beerlytics

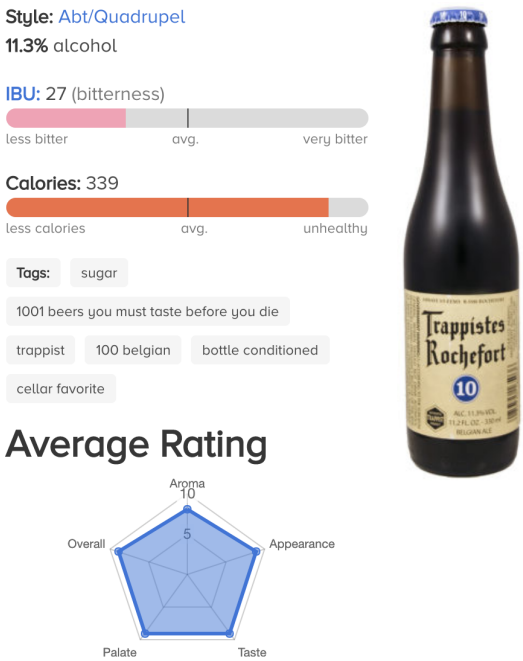


Figure 7: Sidebar information of a beer on Beerlytics

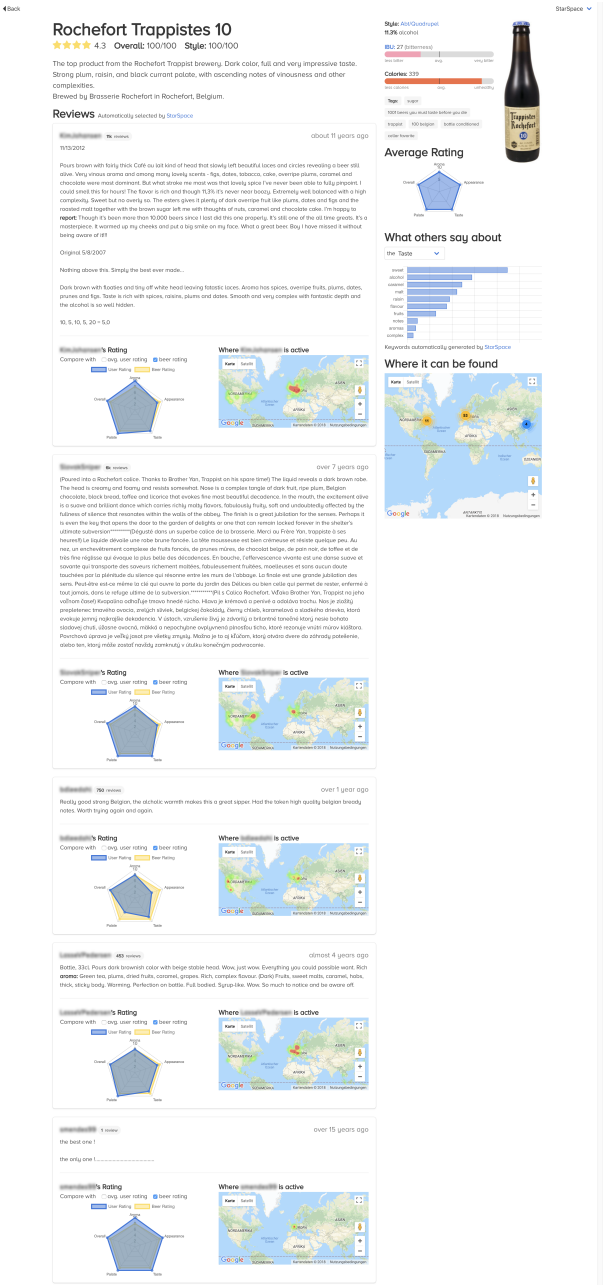


Figure 8: Full Beerlytics beer page