University of Twente

Master Thesis

Information Retrieval by Semantically Grouping Search Query Data

Author
Wim Florijn - s1503251

Supervisors:
Dr. Djoerd Hiemstra
Dr. Mariët Theune
Msc. Knut Jagersberg

February 26, 2019

UNIVERSITY OF TWENTE.

Contents

Abstract

Query data analysis is a time-consuming task. Currently, a method exists where word (combinations) in queries are labelled by using an information collection consisting of regexes. Because the information collection does not contain regexes from never-before seen domains, the method heavily relies on manual work, resulting in decreased scalability. Therefore, a machine-learning based method is proposed in order to automate the annotation of word (combinations) in queries. This research searches for the optimal configuration of a pre-processing method, word embedding model, additional data set and classifier variant. All configurations have been examined on multiple data sets, and appropriate performance metrics have been calculated. The results show that the optimal configuration consists of omitting preprocessing, training a fastText model and enriching word features using additional data in combination with a recurrent classifier. We found that an approach using machine learning is able to obtain excellent performance on the task of labelling word (combinations) in search queries.

Keywords: Machine Learning; Word Embedding; Stemming; Lemmatization; NLU; Neural Network; Recurrent Neural Network; Dimension Reduction;

1 Introduction

With the rise of the Internet, the behaviour of consumers has changed. People are no longer restricted to the products offered by nearby stores: they now can order whatever they like online instantly. Resellers have noticed this shift in consumer behaviour and many have opened up an online store. When people want to know where to buy a product online, they execute a search query on a search engine. The search engine delivers a set of online stores featuring the product, from which a consumer can make a choice.

The search queries inserted in a search engine provide an indication of the interests of consumers. Search engines offer a service where search queries regarding different topics can be retrieved [1]. Producers can use frequently occurring search query themes to learn which products are in demand by consumers. They can make use of these insights by for example starting to offer these products, or optimize their findability. This way producers can adapt better to the ever changing needs of consumers, and optimize their sales. The need for a system which is able to retrieve information by

smart data grouping based on semantics will be explained in the Problem Statement.

This research topic has been provided by New-Media [2]. New-Media analyzes large data sets consisting of search queries, and provides search insights based on word patterns in and between these queries. These search insights are offered to producers in order to guide their company and establish their website in a demand-driven fashion.

1.1 Problem Statement

While it is possible to retrieve a large list of search queries together with their search volume from search engines, this unstructured data needs to be processed in order to lead to search insights. Search insights are developments in search volume of key words in search queries over a time period. To learn these search insights, useful key words need to be detected in the search queries, and labelled with categories. Different key words may be deemed as useful information in different domains. The retrieved useful key words need to be grouped based on semantics, in order to be able to provide a complete overview of each class of semantic coherent words. An example of the labelling of useful information in queries is provided in table 1. The underlining types clarify which word is labeled with a category.

Search Query	Categories
square plant pot from plastic	shape, pots, material
buy four green flowerpots	purchase intent, color, pots
bohemian crystal vase	location, material, vases

Table 1: Queries with labelled objects

The input of the categorization process is an unstructured set of queries, while the output is the set of queries each labeled with a (set of) categories. The problem can be summarized as a classification problem, where the input is a search query, and the output is an ordered sequence of labels for the query tokens. While a method exists which consists of a rule-based approach which retrieves information based on regular expressions (REs) as discussed in [3], the need has arisen to find a quicker, more scalable and smarter approach. The scalability of an application is its capability to handle a growing amount of work, or its potential to be enlarged to accommodate that growth [4]. The REs can be seen as a list of word (combinations)

making up a the categories. In the current system, query categories need to be defined manually, and every type of useful information needs to be added manually to the useful information collection in the form of a separate RE. When the data set is very large, many RE definitions must be added manually, resulting in decreased scalability.

Because the current RE-based system extracts information based on a default list of REs, its performance varies across different domains: the system uses one useful information collection in every domain. Therefore the need to add domain REs exists which implies that the system needs manual adjustments before it can be used in new domains. This limits the scalability in terms of the number of domains. A smart domain-independent system, which is able to learn (domain-specific) word definitions using Machine Learning (ML) approaches, eliminates the need for manually adapting the useful information definitions. This results in a higher profit for the organizations using such an approach.

In preliminary research, an analysis on the RE-based system has been performed in order to find the system's strengths and weaknesses. The research has shown that the manual work component in the RE-based system plays a huge role in achieving good performance, and that ML is a promising technique for the classification of word entities. The RE-based system without the manual work component obtains a recall score of only 0.55 when examining the labelling of unique words with 6 categories. The words accounting for the remaining percentage need to be added by performing manual work. The preliminary research has shown that the usage of independent word embeddings of individual words combined with a simple Neural Network improves the recall score to 0.85. In this follow-up research, a ML-based system is presented which is able to identify and classify words occurring in search queries automatically, eliminating the RE-based system's weaknesses. The following research question is proposed:

How can machine learning based on word embeddings be applied to retrieve information from search queries?

How can the content of queries be categorized? Query content categorization is useful because it enables selecting categories which have been deemed appropriate for delivering search insights by a domain expert. The set of queries can be compressed to sets of word (combinations) making up the categories.

When categorizing the content of search queries, the way in which the features are composed together with the classifier type influence the result. The different way in which query features can be composed and enriched, together with the different promising classifier types need to be examined in order to learn the optimal configuration.

In order to find an answer to the central research question, the question is decomposed into sub-questions which help with finding an answer. The following sub-questions are proposed:

1. Does removing word inflection increase the quality of a word embedding?
2. Which feature composition model is best for categorizing query words?
3. Are additional external data corpora helpful for feature composition?
4. Which query classification method performs best given optimal hyper-parameters?

1.2 Objective

The task at hand is to create a system which is able to classify the content of search queries. The classifiers are implemented by making use of the Keras [5] framework. In order to generate well-performing classifiers, correct features which optimally indicate categories have to be created from the queries. For the task at hand, we hope to create a well-performing classifier which can support and automate the annotation of queries. We will test the performance of the classifiers, and will give recommendations about which classifier configuration to use.

1.3 Outline

In chapter 2, an investigation of the research which has been performed in this field is presented, together with a motivation of why this research is useful. Due to the ever-changing nature of the state-of-the art in ML for text processing, the focus is on work published in recent years. The answers obtained from the literature study and the experiments will be used to answer the research questions. Then experiments must be performed in order to learn which classifiers and features perform best. The setup of these experiments is explicated in chapter 3. The results of the experiments are stated in chapter 4. Finally a conclusion is drawn, and recommendations for future work are proposed in chapter 5.

2 Background

In this chapter, the concepts needed to perform query processing are worked out. In the area of Natural Language Processing (NLP), researchers explore how computers can be used to understand and manipulate natural language text or speech [6]. Furthermore, the process of a computer learning the meaning of a text fragment is called Natural Language Understanding (NLU). A wide range of NLP and NLU techniques are discussed, in order to learn the state-of-the-science in semantic word classification. Furthermore research presenting different classification techniques is discussed, including feature generation methods. Finally ways to optimize the classifiers and visualize the features are presented.

2.1 Pre-processing Techniques

Pre-processing involves the processing of raw data, to prepare it for use in another processing procedure. The goal of pre-processing is to transform data into a format more applicable and effective for the purpose of this process. In this section, variants of a pre-processing component called data reduction are discussed. Data reduction techniques aim to enable the analysis of a reduced representation of the data set without compromising the integrity of the original data while producing quality knowledge. Data reduction techniques may improve the results of query classification, by establishing higher quality word features.

Stemming, lemmatization and morphological segmentations are data reduction methods. The goal of both stemming and lemmatization is to reduce inflectional and sometimes derivationally related forms of a word to a common base form. Morphological word segmentation is the process of dividing a word into its component words. Removing the inflection of words before generating word features may improve the quality of the word features. Stemming, lemmatizations and morphological word segmenation are used to map words to a denser matrix, in order to increase a system's recall by sacrificing precision. A systems precision is the fraction of relevant instances among its retrieved instances, while its recall is the fraction of relevant instances that have been retrieved over the total amount of relevant instances. Stemming reduces all inflected and derivational forms of a word to the same stem by removing and/or appending characters. The most well known stemming algorithm is the Snowball stemmer which has been created by Porter [7] in which rules can be expressed in a natural way. Furthermore Paice [8] has come up with the Lancaster stemming method which is more aggres-

sive and faster. The aggressiveness of a stemmer corresponds to the degree of inflection reduction. A stemmer which is not aggressive enough or too aggressive yields sub-optimal results where the inflection-reduced words are either too specific or not specific enough. Lemmatization on the other hand reduces all inflected forms of a word to the same lemma by performing a dictionary-lookup. Gaustad [9] has created a lemmatizer which makes use of the CELEX lexical databases for dictionary-lookup. The Frog [10] NLP package also features a Dutch lemmatizer using the CELEX database. The Frog [10] also features a morphological word segmenter which also makes use of the CELEX database for word-lookup.

2.1.1 Suitability

The lemmatizer and word segmenter featured in Frog are the most suitable dictionary-based inflection reducing methods for Dutch words, because they make use of the most extensive Dutch word database available. The Snowball stemmer is the most suitable stemmer, because it finds the correct balance between aggressiveness and speed. These are the most suitable techniques for removing the inflection of Dutch words.

2.2 Word Features

To be able to successfully classify words into their semantic category, appropriate word features have to be determined. Recently a lot of research has been performed in the area of word embeddings. Word embeddings can be seen as language modeling and feature learning techniques, which map words or phrases from a vocabulary to vectors of real numbers. Mikolov et al. [11] have started an innovation trend in word embeddings, by creating a local context window based continuous skip-gram model called word2vec. A skip-gram model tries to predict the source context words (surrounding words) given a target word (the center word) in a word n-gram. The model is able to learn high-quality distributed vector representations that capture a large number of precise syntactic and semantic word relationships. The word vectors are learned functions of the internal states of the model. Another method called GloVe [12] uses global word co-occurrence counts to extract and represent the contextual meaning of words in terms of a feature vector. Speer et al. [13] have combined data from different data corpora like ConceptNet with word2vec and GloVe word vectors by making use of retrofitting (extending an existing model with information not fitted during manufacturing), and the resulting model performs better than the individual

models. A drawback of the methods discussed above is that they are unable to predict the embedding of unseen words. Therefore Bojanowski et al. [14] have created fastText which is able to predict the embedding of unknown words, by training vectors on character n-grams according to the skip-gram method. This enables the generation of representative embeddings for unseen inflectional word forms. The word embeddings are composed from a bag of n-gram vectors.

The word embedding models discussed above only capture a single word representation, while ambiguous words have multiple representations dependent on the context. Li and Jurafsky [15] find that multi-context embeddings improve performance on part-of-speech tagging, semantic relation identification and semantic relatedness but not named entity recognition or various forms of sentiment analysis. Nguyen et al. [16] have proposed a method to learn multi-sense word embeddings, based on an underlying topic model. The topic of a word is determined based on its surrounding words. Huang et al. [17] propose a technique which makes use of a neural network to learn (multiple) semantic meanings of words. The neural network takes into account both the local and global context of a word while learning. It is important to take away from this research that words can have multiple meanings, and a ML system performs better when it is able to learn more than one word embedding. Finally, the current state-of-the-art in word representations are models called ELMo [18] and BERT [19] which model characteristics of word use like syntax and semantics, together with how these uses vary across linguistic contexts. Elmo uses a deep bidirectional language model to learn the word vectors, which is a model where a word feature is generated by using the words on both of its sides in a sentence. BERT uses attention transformers as opposed to a deep bidirectional model. These models take into account the surroundings of a word when generating its embedding. All language models have in common that they perform best on tasks when having been trained on large data sets originating from the target domain.

Furthermore external graph-based text corpora like Open Dutch Wordnet [20] or ConceptNet [21] can be used to learn semantic relationships between words. By calculating the degree of association between words and categories, the relationships become suitable to be used as a word feature.

2.2.1 Suitability

The most suitable methods to compose word features are word2vec and fastText, due to their maturity and wide adoption in the community. The models are widely used and pre-trained models are available. Word embeddings can be enriched by incorporating existing knowledge graphs like the Open Dutch Wordnet.

2.3 Word Classification

Now that the state-of-the-art in useful word features is clear, the state-of-the-art in word classification methods will be discussed. Traditionally, words features like n-gram frequencies and WordNet information have been used to classify words into e.g. named entities. Fleischmann and Hovy [22] have classified named entities using different feature types with Decision Trees (DT), Neural Networks (NN), Support Vector Machines (SVM), Naive Bayes (NB) and k-Nearest Neighbours (kNN) classifiers into fine-grained categories.

More recently however, simultaneously with the rise of word feature vectors, the classification methods have shifted to NN variants. Wei et al. [23] have done research in the area of Named Entity Recognition (NER). They have used features like words, part-of-speech tags, chunking information and word shape features such as dictionary and morphological features to train a conditional random field (CRF). A CRF [24] is a type of probabilistic model which is used to segment and label sequence data. Additionally, a Bidirectional Recurrent Neural Network (BRNN) was trained with word embeddings. A RNN is a variant of a neural network which takes into account the feature vectors preceding a feature vector during training, where its directed nature allows for temporal dynamic behavior for a time sequence. When the RNN is Bidirectional, also the subsequent feature vectors are taken into account, by training the RNN simultaneously in positive and negative time direction [25]. Finally, the output of both models is fed into a support vector machine (SVM) in order to obtain the combined results.

Lample et al. [26] have also done research in the area of NER. They have used a RNN variant called Long Short-term Memory (LSTM) which has been trained using a combination of both supervised and unsupervised learning. The word features in the classification are character-vector based word vectors, combined with distributed word2vec word vectors. Both the unsupervised and combined models obtain state-of-the-art performance in NER

in four languages including Dutch.

2.3.1 Suitability

Little research has been performed in the classification of modern word embeddings using conventional classification methods like DT, SVM, NB or kNN, which makes them less likely to be a good candidate. The most suitable methods for classifying word embeddings are the various NN types. The RNN and BRNN extensions help with understanding words, because their ability of temporal dynamic behavior for a time sequence allows taking into account context.

2.4 Sentence Features

There are different feature types being used for sentence/text classification. Traditionally, linear methods like bag-of-words or bag-of-n-grams were used to classify sentences [27]. Recently however, more researchers exploit word embeddings when performing sentence classification in combination with Convolutional Neural Networks (CNNs) [28]. A CNNs is a deep neural network type which opposed to other NN variants requires minimal preprocessing, because it is able to learn filters from the input data. Sentence features can be learned from individual word features by using a max pooling architecture [29], [30]. A sentence feature is constructed using a sliding window of n word features. The best performing sliding window feature is appointed the sentence feature.

Furthermore, Cer et al. [31] have come up with a sentence encoding model which is specialized in creating sentence embeddings which can be used for transfer learning to other NLP tasks. Transfer learning means that a pretrained model is fitted for a domain by further training an existing model on domain-specific data. The model uses a transformer encoder and a deep averaging network encoder to encode word vectors into sentence vectors. The sentence level embeddings surpass the performance of transfer learning using word level embeddings alone.

Finally Mikolov et al. [32] have created a unsupervised learning method which creates fixed-length paragraph vectors from variable-length sets of words. A Distributed Memory Model of Paragraph Vectors is used to represent paragraph semantics. They have achieved state-of-the-art results on several text classification and sentiment analysis tasks.

It is obvious that sentence features which have been trained on a data set

from the target domain achieve better performance than features which have been trained on a generic data set. Because a large domain-specific data set is not always available, one could use transfer learning which starts with an already trained model. Furthermore, Limsopatham and Collier [33] have come up with a novel approach which combines word embeddings created from both generic and target domain corpora using a CNN and they achieve state-of-the-art performance on several sentence classification tasks.

2.4.1 Suitability

The most suitable sentence feature composition method is a bag of word embeddings of the words occurring in a sentence. Which of the word embeddings are useful is then learned during training (using e.g. a sliding windows of size n with a max pooling neural network architecture). This approach is most appropriate for the sentence layout of queries, often with one or two word(s) indicating the query topic. The relative small training data set size does not allow for state-of-the art results when using the paragraph vector approach. Furthermore it is interesting to also investigate the traditional bag-of-words and bag-of-ngrams features in order to be able to make a comparison.

2.5 Sentence Classification

Now that the sentence features are clear, we will discuss the state of the art in sentence classification methods. Blei et al. [34] describe Latent Dirichlet Allocation (LDA), a generative probabilistic model for collections of discrete data such as text corpora. LDA is a three-level hierarchical Bayesian model, in which each item of a collection is modeled as a finite mixture over an underlying set of topics. Each topic is, in turn, modeled as an infinite mixture over an underlying set of topic probabilities. Moody [35] has expanded LDA by incorporating word embedding vectors, naming it LDA2vec. The model learns dense word vectors jointly with Dirichlet-distributed latent document-level mixtures of topic vectors. In the context of text modeling, the topic probabilities provide an explicit representation of a document.

Collobert and Weston [36] have developed a multi-task deep neural network semi-supervised learning approach which is able to perform NLP sub-tasks including part-of-speech tagging, chunking, named entity tagging, semantic role labeling and finding semantically similar words based on word vector features and sentence features. The results of sub-tasks are shared in this deep neural network, resulting in state-of-the-art performance. Kim [30]

has used CNNs to classify sentences, by means of using word vector features computed according to the word2vec skip-gram method. The neural network is trained on sentence features derived from the vectors of individual words in a sentence. This approach improves on the state-of-the art in multiple domains, including sentiment analysis and question classification. This research indicates that combining unsupervised methods to create word embeddings with supervised methods for classification yields great results when classifying sentences. Yin et al. [37] have compared the performance of CNNs with the LSTM and GRU RNN variants. The GRU has a more simple internal structure than the LSTM, which results in the LSTM having the ability to model more temporal related features. They execute the model types on different NLP tasks like sentiment classification, answer selection and part-of-speech tagging. They find that RNNs deliver robust performance and outperform CNNs in almost all NLP tasks.

Amrit and Hek [38] performed a literature study on clustering the results of brainstorm sessions, and have compared the performance of promising unsupervised word clustering techniques by evaluating them on one data set. They conclude that the AKC [39] and WUP [40] algorithms obtain the highest accuracy (73 and 72% respectively). AKC and WUP both make use of IS-A relations, which words share in the Dutch word knowledge graph Cornetto [20], in order to cluster words. These methods perform better than word2vec-based methods.

2.5.1 Suitability

The most suitable method for the classification of sentence embeddings is by using some NN variant, which is trained by sentence features composed on top of word embedding vectors. The LDA(2vec) is more suitable for the classification of documents instead of short sentences. The usage of existing knowledge graph data is less suitable in our scenario, because clustering using IS-A relations of words with all other words in a data set is a exponential time-complex process which does not scale well.

2.6 Hyper-parameter Optimization

Classification algorithms feature a lot of parameters which influence their performance. The process of fine tuning a model's parameters in order to optimally solve the machine learning problem is called hyper-parameter optimization. While grid-based and random-based search have been traditional way of searching for a model's optimum, more efficient methods have been

proposed. A method using Bayesian optimization [41] has been proposed, where a learning algorithms generalization performance is modeled as a sample from a Gaussian process (GP). Bayesian optimization performs a trade-off between exploration and exploitation in order to minimize the number of function queries. Furthermore, other techniques like gradient-based optimization [42] where hyper-parameters are optimized using gradient descent, and evolutionary optimization [43] where the space of hyper-parameters for a given model is explored by making use of evolutionary algorithms have been proposed.

2.6.1 Suitability

The most suitable method for optimizing the hyper-parameters of the classification algorithms is Bayesian optimization, because of its state-of-the-art performance in combination with its wide adoption in the field. Its characteristic to minimize the number of times a classifier has to be trained makes it well suited for this scenario where the optimization of classifiers is expensive to evaluate.

2.7 Dimension Reduction

Dimension reduction techniques are useful for creating a compact vector which can among other things be used for visualization and classification tasks. Principal Component Analysis [44] involves finding the solution of an eigenvalue-eigenvector problem for a positive-semidefinite symmetric matrix. The principal components are ordered in a way that the first few hold most of the variation found in the original variables. PCA seeks to preserve the distance structure within the data. Recently other techniques which favor the preservation of local distances over global distance have been proposed. T-distributed Stochastic Neighbor Embedding (t-SNE) [45] is a machine-learning based technique which widely used for data visualization. T-SNE first constructs a probability distribution over pairs of high-dimensional objects. Then t-SNE defines a similar probability distribution over the points in the low-dimensional map, and it minimizes the KullbackLeibler divergence between the two distributions with respect to the locations of the points in the map. UMAP [46] is a novel technique which has a strong mathematical foundation being based on Riemannian geometry and algebraic topology. Its clustering performance is similar to t-SNE, while it is claimed to be faster and more scalable.

2.7.1 Suitability

For the visualization of feature vectors in two-dimensional space for semantic clustering, the preservation of local distances is favored over the preservation of global distances. Therefore the most suitable method for reducing the dimension of feature vectors is UMAP. It is a recent but well adopted technique, which strong mathematical foundation combined with its state-of-the-art performance make it the best choice especially for large data sets.

2.8 Application

The pre-processing techniques discussed in chapter 2.1.1 will be used to inspire how sentences and individual words can be pre-processed in this research. Furthermore, the word features and classification techniques described in chapters 2.2.1 and 2.3.1 together with the sentence feature calculation techniques and the sentence classification techniques discussed in chapter 2.4.1 and 2.5.1 will be used for the classification of the word combinations occurring in search queries. These classification methods will be optimized using the hyper-parameter optimization technique described in 2.6.1. Finally word and sentence feature vector distributions are visualized using the dimension reduction technique discussed in chapter 2.7.1.

3 Methodology

To determine how ML can be used to create an adaptive query processing system, we design new systems which are inspired by the related research discussed in chapter 2. In order to measure the performance of the systems, a valid testing method is constructed. Finally, real world data sets are used to measure the performance of the method in different domains according to the testing method defined later in this chapter.

3.1 Data Sets

In this research, data sets which consist of user defined search queries (provided by New-Media) are used. These data sets originate from different domains, and contain various flavours of query data. Each query in the data sets can be seen as a simple sentence consisting of mainly key words. Two different label types are attached to the tokens in the queries:

1. Global category: a label attached to a token which indicates the global topic of the entire query.

2. Detail categories: a label attached to a token which indicates details about the query topic (Global category).

The global category label in a query is an entry from the list of domain-specific global categories. Every domain has a defined set of global categories. The detail categories are words which provide details about the global category. They indicate what information consumers search for with respect to the global category. Detail categories are not domain-specific and contrary to global categories occur in all different domains. Tokens belonging to the same global category, and tokens belonging to the same detail category are semantically related. This allows for unsupervised semantic clustering and supervised classification of the tokens in the queries. Multiple data sets are used to evaluate the ML-based system. Examples of queries where the tokens are labelled with both label types are provided in table 2. The different underlining types clarify which word (combination) is labeled with a category.

Query	Glob. Category	Det. Categories
square **plant pot** from plastic	**pots**	shape, material
buy four green **flowerpots**	**pots**	purchase intent, color
bohemian crystal **vase**	**vases**	location, material

Table 2: Queries with labelled global and detail categories

The data sets used to evaluate the ML-based system are described in the following sections.

3.1.1 Barbecue product data set

This product data set contains Dutch queries from the domain of barbecues. The data set consists of 14423 relevant queries of users looking for concepts or products related to barbecues. Every query contains one global category label. The data set is imbalanced over the global categories. The distribution of the global categories over the relevant queries is visualized in figure 1.

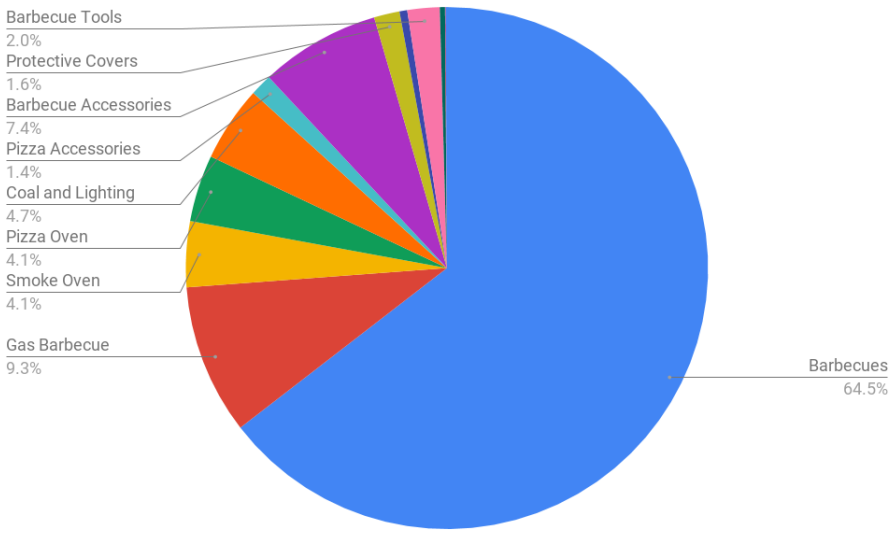

Figure 1: The distribution of the global categories over the relevant queries

The data set contains 9773 detail category labels. The data set is more balanced over the detail categories than over the global categories: no single

detail category dominates the distribution. The distribution of the detail categories across the relevant queries is visualized in figure 2.

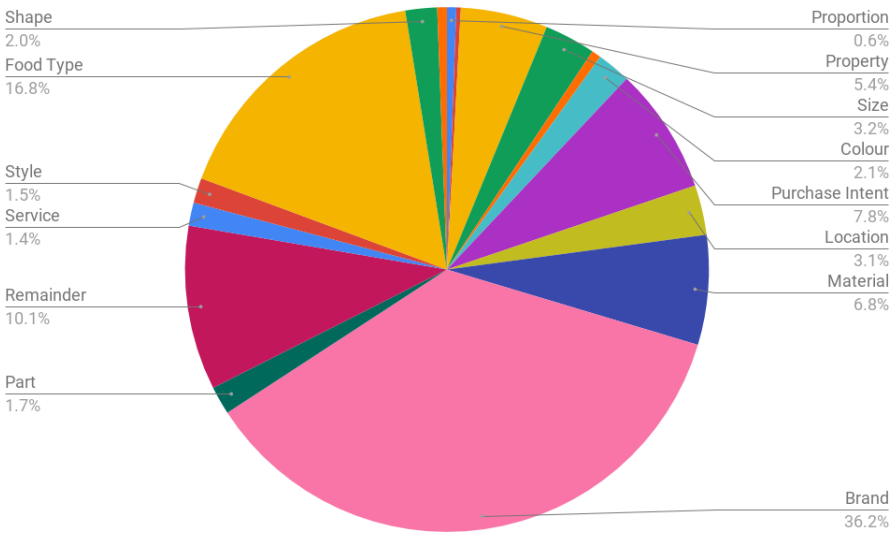

Figure 2: The distribution of the detail categories over the relevant queries

Examples of relevant queries of the barbecue data set where the tokens are labelled with different global categories and detail categories are presented in appendix A.

3.1.2 Shower product data set

This product data set contains Dutch queries from the domain of showers and baths. The data set consists of 36143 relevant queries which are search terms of users looking for concepts or products related to showers or bathing products. The global categories are again imbalanced in this data set. The distribution of the global categories over the relevant queries is visualized in figure 3.

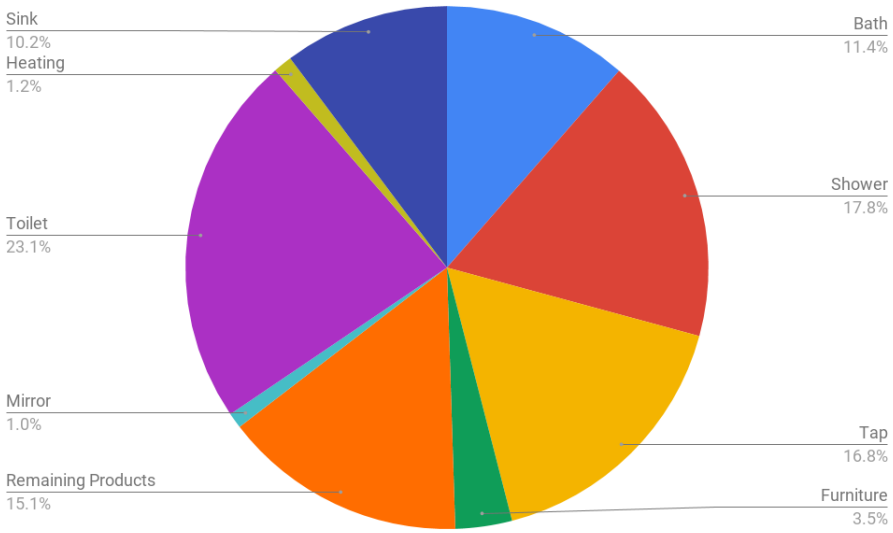

Figure 3: Distribution of the global categories over the relevant queries

The data set is more balanced over the detail categories than over the global categories. The distribution of the 51751 detail categories across the relevant queries is visualized in figure 4.

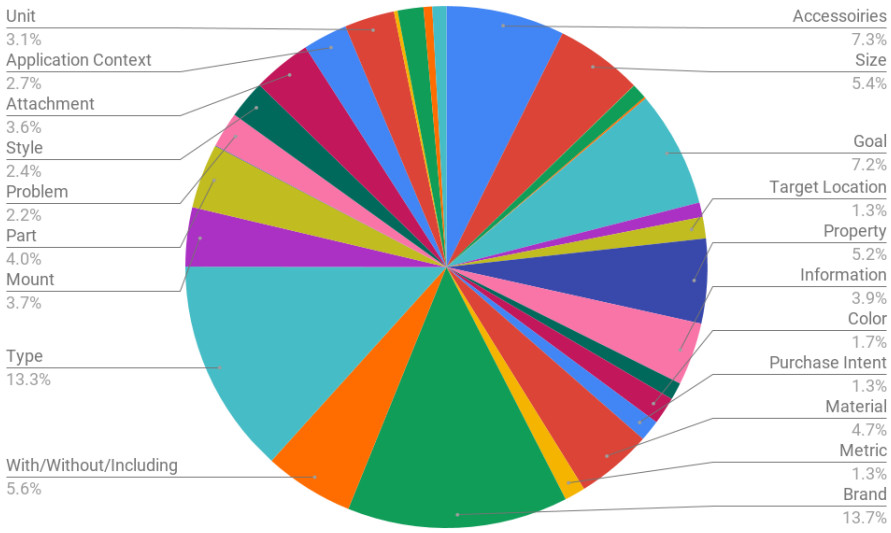

Figure 4: Distribution of the detail categories over the relevant queries

Examples of relevant queries of the shower data set where the tokens are labelled with different global categories and detail categories are presented in appendix B.

3.1.3 Goal

The goal of the ML-based system is to predict the global and detail category labels of the tokens occurring in queries. In order to reach this goal, a ML-based system which is able to detect and match both global and detail categories is created. How the classification of tokens into global categories and detail categories takes place is discussed next.

3.1.4 Pre-processing

Multiple pre-processing methods are examined in order to learn if and which type of word inflection removal helps optimizing the task of classifying queries. First the queries are tokenized. The inflection reduction techniques which are examined are the following:

1. Stemming
2. Lemmatization

3. Morphological word segmentation

4. Omitting pre-processing

The tokens are stemmed using the Porter Snowball stemmer [7]. Furthermore the tokens are lemmatized using the Frog [10] NLP package. Examples of stemming and lemmatization are provided in table 3.

Token	Stem	Lemma
bins	bin	bin
flowerpots	flowerpot	flowerpot
university	univers	university

Table 3: Stemming and lemmatization

Also, the tokens in the queries are morphologically segmented using the Frog [10] NLP package. This means that single word tokens are split up into multi word tokens. Only the free morphemes are used, while the bound morphemes (inflection) are omitted. These multiple token segments can later be matched with the global categories. Examples of morphological word segmentation are provided in table 4.

Token	Free Morph. 1	Free Morph. 2	Bound Morph. 1
bins	bin		s
flowerpots	flower	pot	s
universities	university		s

Table 4: Morphological segmentation

3.2 Classification

Queries contain one or more tokens indicating a global category, and zero or more tokens indicating detail categories. Tokens within a category are semantically related. Therefore a classification technique based on the word embeddings of the words occurring in a query is proposed.

3.2.1 Feature Composition Methods

As word features, word embedding vectors are used. As described in section 2.2.1, the word2vec and fastText models are suitable feature composition

methods. These unsupervised models are implemented using the gensim [47] Python library. The models are trained on the query data set described in section 3.1. The resulting word embeddings are 100-dimensional feature vectors. This relatively small size has been chosen because of the limited train set size. Like discussed in section 2.4.1, a bag of word embeddings is suitable for use as a sentence feature.

3.2.2 Additional data corpora

Word embeddings are of better quality when being trained on a large data-set [48] originating from the target domain. Because large domain-dependent data sets are not always available, other models pre-trained on generic domains can be used to improve word embeddings [33]. Wikipedia-based pre-trained word2vec [49] and fastText [14] models featuring 300-dimensional word embedding vectors are combined with the domain-specific models using vector concatenation in order to generate richer domain-fitted word embeddings. The performance of the enriched features is compared with the performance of the standard features.

3.2.3 Classification Methods

According to section 2.3.1 and 2.5.1, neural network variants including CNNs and RNNs deliver state-of-the art performance on the task of text classification. Therefore the following classifier types will be evaluated on the task of query classification:

1. Artificial neural network (NN)
2. Long-short term memory neural network (LSTM)
3. Gated recurrent unit neural network (GRU)
4. Convolutional neural network (CNN)

These classifiers need a layout which performs well on a text classification task. Such an architecture commonly includes max-pooling and dropout layers [30], [37]. The layouts are as follows. All classifiers have an input layer. The input layer receives an array of word embeddings for each query. The arrays are padded to the size of the longest query.

For regularization, the models employ dropout on the input layer. A dropout layer which randomly omits a subset of the hidden unit in the network

reduces the overfitting of a classifier [50] especially on small training data sets. The models employ a dropout layer [51]. The dropout layer is located behind the input layer with word embeddings.

Furthermore, the classifiers contain a specific architectural unit. The NN model features a variable amount of hidden layers each consisting of a variable amount of hidden units. The LSTM model contains a LSTM component consisting of a variable amount of LSTM cells. The GRU model contains a GRU component consisting of a variable amount of GRU cells. The CNN model features a variable amount of convolutional layers each with a variable amount of filters.

The final layer is the output layer. The output layer delivers a prediction of the category labels of the query. The size of the output layer is the number of words in the input queries. Each output label obtained from the output layer corresponds to a query word. The global layout of the classifiers is visualized in figure 5.

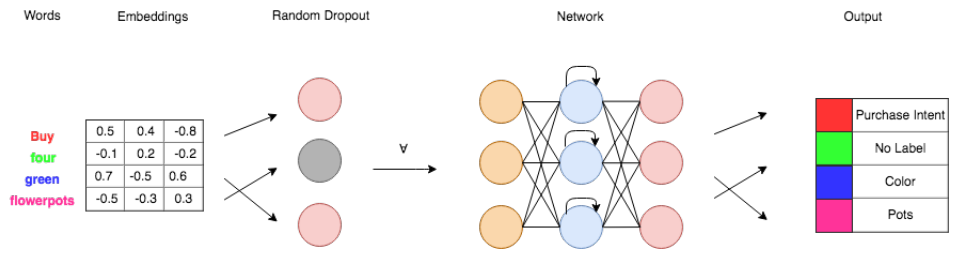

Figure 5: The global layout of the classifiers

The classifiers are implemented in Python using Keras [5] (with a tensorflow [52] backend).

3.2.4 Test Setup

The quality of the features described in section 3.2.1 influences the quality of the downstream query classification task. All combinations of the 4 pre-processing techniques, 2 word embedding models and 2 additional data options need to be evaluated with each of the 4 classifiers mentioned in section 3.2.3. Therefore a total of 4 * 2 * 2 * 4 = 64 combinations need to be evaluated on both the data sets mentioned in section 3.1. The data sets are split in a train/test set and a validation set, using a split ratio of 80:20.

For every combination of the feature creation methods and the classifiers, a different set of optimal classifier hyper-parameters may exist. Therefore the hyper-parameters of the classifiers need to be optimized before every experiment following the Bayesian optimization method discussed in section 2.6.1, on the validation set. The optimizable classifier components are the following: in all classifiers, the dropout rate of the dropout layer is optimized. Furthermore, in the NN the number of hidden layers, and their sizes are optimized. Additionally, in the LSTM the number of LSTM cells is optimized. The number of GRU cells is optimized in the GRU-based classifier. Finally, in the CNN the number of convolutional layers, together with their kernel sizes are optimized.

The feature composition methods in combination with the classifiers will be evaluated by performing 5-fold cross validation on the train/test set. First, the data set is randomly partitioned into 5 equal size splits. Of the 5 splits, one split is retained as the test set, and the remaining 4 splits are used as train set. A valid testing setup has been compiled in order to ensure that the train and test sets are kept strictly separate. Therefore the feature composition models are trained strictly on the train set. The words from the test set are looked up in this model in order to generate test set word embeddings. The classifier is trained on the train set, and evaluated on the test set. This process is repeated 5 times, with each of the 5 splits used exactly once as the test set. Finally the mean of the 5 results is calculated to produce a single estimation. This ensures that all the entries in the data set are eventually used for both training and testing. The way the data set is split in order to facilitate hyper-parameter optimization and model evaluation is visualized in figure 6. The performance of the classifiers on the tasks of classifying global and detail categories is evaluated for each task individually.

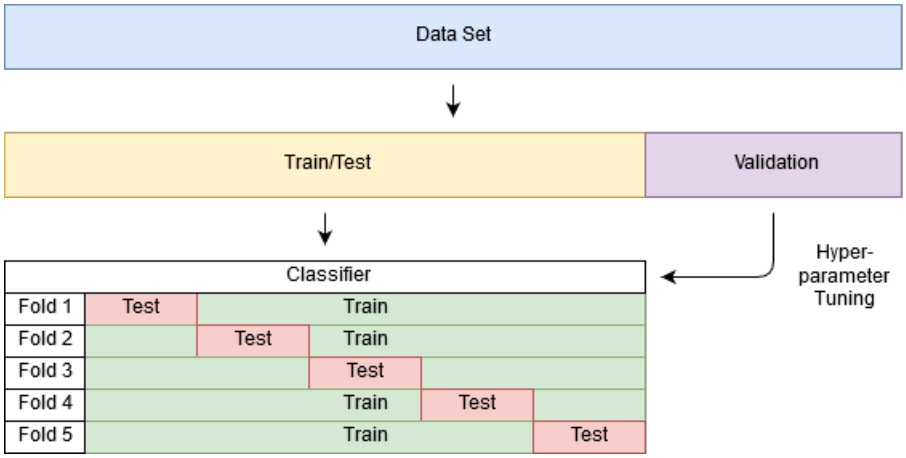

Figure 6: The data set splits

3.2.5 Performance Measures

In this section, appropriate performance measures are determined for the labelling of query word (components) with both global and detail category labels. Because the isolated results of each of these two classifiers are interesting, the performance of these two classifiers will be evaluated separately. Each of these two tasks can be viewed as executing a multi-label classifier on the words occurring in the queries. For the sake of evaluation, the used multi-class classifiers can be viewed as a set of binary classifiers, each predicting one class.

The main difference between the global categories and the detail categories, is that the distribution of the global categories is imbalanced, while the distribution of the detail categories is balanced. First a couple of concepts will be discussed, in order to motivate why a certain evaluation metric has been chosen for a task. True positives (TP) are cases in which a binary classifier correctly predicted a value as having the class. True negatives (TN) are cases where a value has correctly been predicted as not having the class. In the case of false positives (FP) the classifier incorrectly predicted a value as having the class. Finally, in the case of false negatives (FN) the classifier incorrectly classified a value as not having the class. The true positive rate (TPR) is the percentage of the values actually belonging to the class which are correctly classified. The false positive rate (FPR) is the percentage of values not actually belonging to the class but who have been

incorrectly classified as doing so. Thus the TPR and TNR get higher when the number of TPs and TNs get higher. The positive predicted value (PPV) is the percentage of values actually belonging to a class and which have been correctly classified as doing so.

Because New-Media prioritizes obtaining qualitative good results over obtaining quantitative good results, the goal of the classifiers is to encounter a minimal amount of results which are incorrectly classified by one binary classifier as having some class (FP). Therefore, resulting from a literature review, this research will use the ROC-AUC (area under curve) [53][54] performance metric which is used to measure the actual performance of classifiers on imbalanced data sets [55][56].

The receiver operating characteristic (ROC) graphs show the classifier performance plotting the TPR against the FPR. The area under this ROC curve (which is known as the AUROC) shows the ability of the classifier to correctly separate the data set being tested into the different categories. The AUROC can be defined as the expectation that a uniformly drawn random positive is ranked before a uniformly drawn random negative [57]. The AUROC score gets higher with an increasing TPR, or a decreasing FPR.

In order to calculate this score in a multi-class setting, the TPR and FPR are calculated over the combined result of all binary classifiers, resulting in the micro-averaged AUROC score. The micro-averaged AUROC score aggregates the distinctive abilities of all classes while incorporating their size when computing the averaged metric, which is more useful when classes are imbalanced than simply averaging all individual class scores. The direct connection of the AUROC score with the number of FPs, and suitability for imbalanced data sets, make it a suitable metric for measuring the performance of the classification of global categories.

When evaluating the performance of the system on labelling the detail categories, two evaluation metrics should be taken into account. Again, the multi-label classifier is viewed as a set of binary classifiers for the sake of evaluation. First of all, there should be examined how well the classifier is able to correctly classify the words labeled with the class. This ability can be measured by calculating the TPR of the classifier. Furthermore there should be examined how many of the words have correctly been classified as having the class. This is measured by calculating the PPV score. The F1 metric combines these two scores into a single score by taking the harmonic mean. The scores of all binary classifiers are aggregated by taking the micro-average. Furthermore the F1 metric has been widely used in the

area of NLP to measure the performance of comparable research like named entity recognition [58][59]. These properties make the F1 metric suitable for the task of measuring how well the system is able to classify the tokens with a detail category label in the queries.

4 Results

This chapter displays the results of this research. It presents and discusses the performance of the ML-based system in matching global and detailed category labels to query tokens. Furthermore, the differences in performance when using the pre-processing methods, model types and the optional usage of additional data corpora in combination with the classifiers is discussed. This leads to insights in which setup performs optimal in the scenario of this research.

4.1 Global categories

To get an idea of the distribution of the global categories across the queries in the data sets, the global category labels are visualized. For each data set, a fastText model is trained. Then the queries are tokenized, and the feature vectors of the n words occurring in the query are looked up. This results in a 2-dimensional matrix containing n rows with word vectors. For multi-word tokens, the mean of every column is calculated, in order to generate a single token feature vector. Finally the token feature vectors are dimension reduced using the UMAP technique, which is motivated in section 2.7.1.

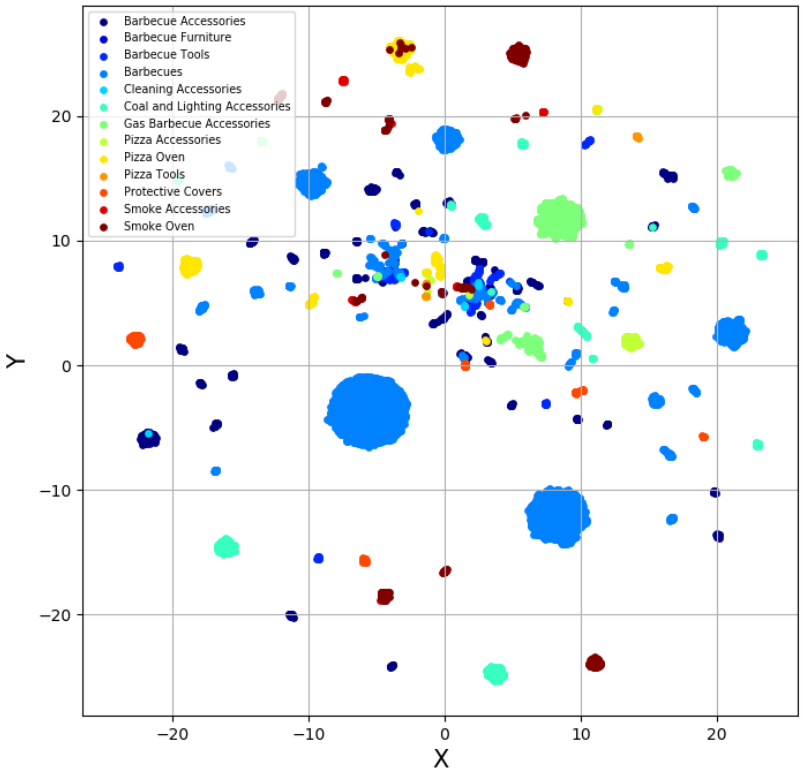

Figure 7: Dimension reduction of the features in the barbecue data set

The visualization of the features of the tokens which have a global category label, occurring in the 14423 queries of the barbecue data set, is presented in figure 7. There can be concluded that most of the tokens within the global categories are clustered well with other tokens belonging to the same category. The clusters of a single global category label are divided across the visualization space, because of the local distance preservation characteristic of UMAP which focuses on preserving the local neighbour relations. Some outlier tokens which are separated from the main category clusters can be observed. There can be concluded that global category clusters are well defined when the token features vectors are reduced to two dimensions.

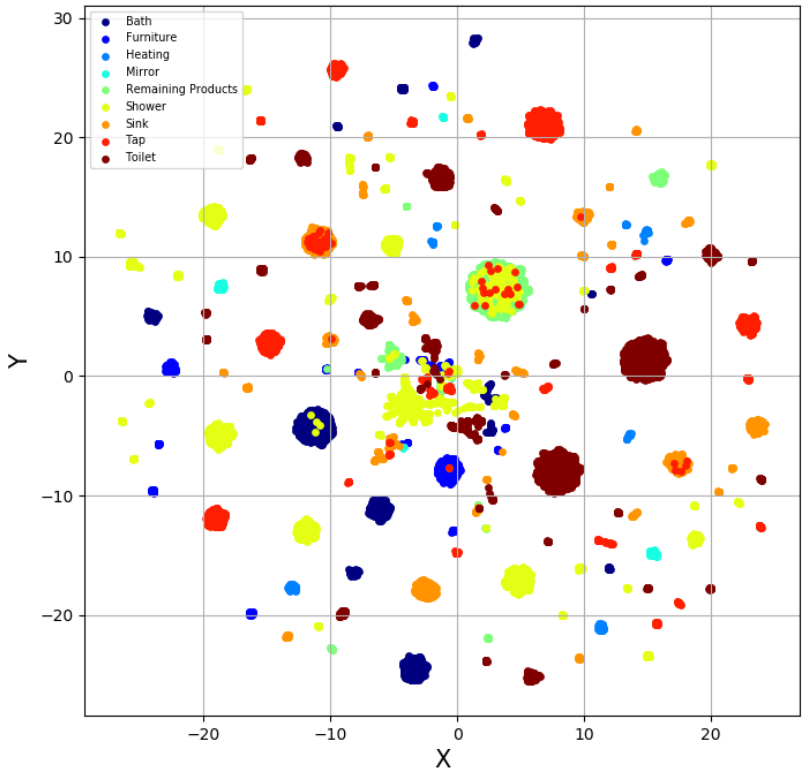

Figure 8: Dimension reduction of the features in the shower data set

The visualization of the features of the tokens which have a detail category label, occurring in the 36143 queries of the shower data set, is presented in figure 8. There can be concluded that the shower data set contains more clusters than the barbecue data set. Most of the tokens are in or close to a cluster of their global category although there are more outlier tokens than in the barbecue data set. There can be concluded that the categories are well defined when the feature vectors are reduced for visualization in two-dimensional space, however they are not as clearly distinct as the category clusters in the barbecue data set.

The research continues with examining which pre-processing method per-

form best when classifying tokens into global categories. To answer this question, the pre-processing methods used for token inflection reduction during all experiments are examined. 16 results are obtained for each of the 4 pre-processing methods, on both the data sets. The mean and maximum values of the AUROC scores in these groups are calculated. The results of the different pre-processing methods on the data sets are visualized in figure 9. For clarification, the scores are listed above the respective groups.

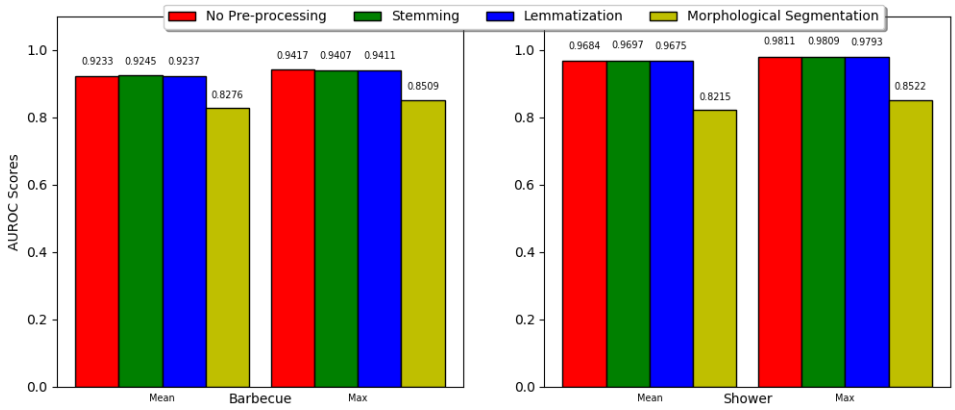

Figure 9: The average and best classifier performance using the different pre-processing methods

From the results can be concluded that the morphological segmentation technique obtains significantly lower mean and maximum scores than the other pre-processing methods. The relative performance of the pre-processing methods on the two data sets is nearly identical. From the results can be concluded that the performance difference between omitting pre-processing, and performing stemming or lemmatization is negligible. Morphological segmentation is an outlier because it obtains far lower scores than the other pre-processing methods. There can be concluded that omitting pre-processing delivers the best results: performing pre-processing does not substantially improve the classifier results.

Now the different model types used to create token features in the different experiments are evaluated. To find out which of the two model types performs best, the mean and maximum scores of each of the two groups containing the scores of 32 experiments on both data sets are calculated. These scores are visualized from left to right in figure 10.

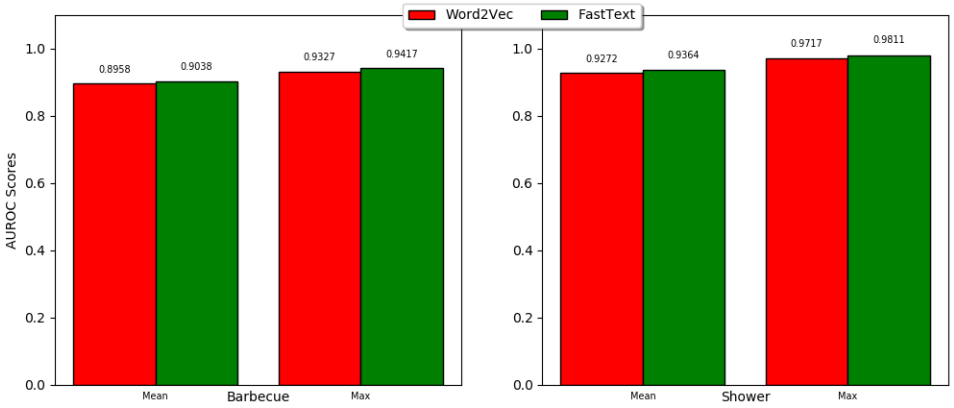

Figure 10: The average and best classifier performance using the different model types

There can be concluded that while the performance of both model types is good, the fastText model performs overall better on both data sets. Both the average and maximum scores are higher in the experiments when a fastText model is used to create word embeddings. Using both of the model types, the results are better on the shower data set, than on the barbecue data set, however the relative differences between the model types stay intact. There can be concluded that fastText outperforms word2vec substantially.

Next step is to examine the performance of the classifiers when the features have been enriched using a large domain-independent data corpus consisting of all Wikipedia articles. The AUROC scores of all 64 experiments per data set are grouped on the case whether additional data has been used to compose the word features. The mean and maximum AUROC scores of these groups are calculated, and presented in figure 11.

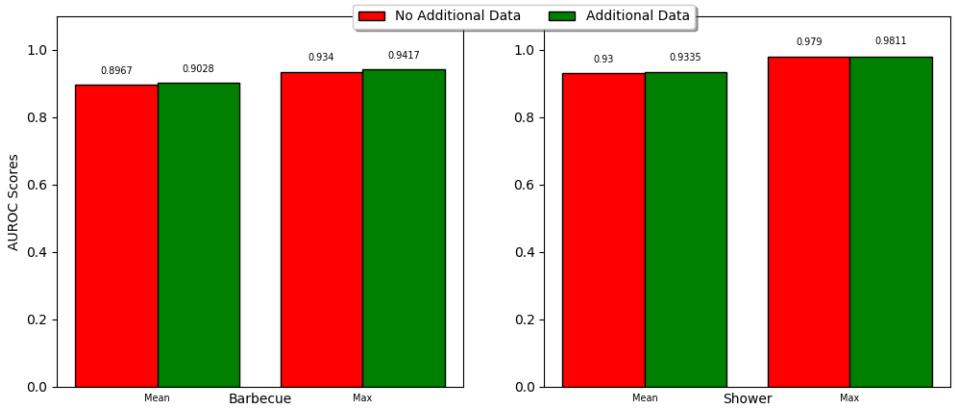

Figure 11: The average and best classifier performance when using additional data

From the results can be concluded that the performance of the classifiers increases slightly when using enriched features. Both the average, and the best scores on both data sets are approximately 1 percent higher when the features have been enriched using Wikipedia data. From the results can be concluded that incorporating additional data sources improves the classifier performance.

Finally, the performance of the classifier variants on labelling words with global category labels is evaluated. The resulting AUROC scores of all experiments are grouped on the usage of the classifier types on both data sets. The mean and maximum performance of these groups is calculated, and the results are presented in figure 12.

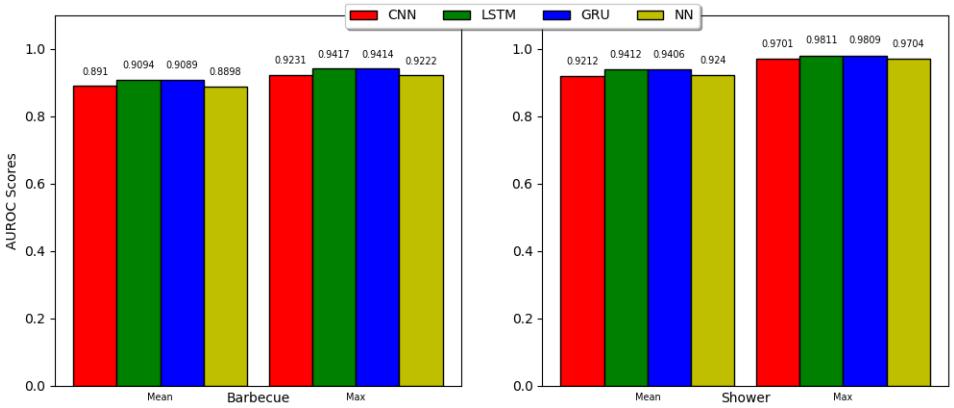

Figure 12: The average and best performance of the different classifier variants

From the results can be concluded that the average and best performance of the LSTM and the GRU is better than that of the NN and the CNN, on each of the two data sets. The performance of the LSTM is slightly better than that of the GRU, however the difference is negligible. There can be concluded that the recurrent models obtain substantially better results than the NN and CNN on the task of labelling query tokens with global category labels.

The best performing configuration of the feature creation methods with a classifier is identical two data sets. The best performing configuration is derived from the maximum scores displayed in figures 9, 10, 11 and 12. On the barbecue data set, the configuration of omitting pre-processing, training a fastText model, and incorporating an additional data set, in combination with a LSTM achieves the highest AUCROC score of 0.9417. On the shower data set, the same combination achieves the highest AUCROC score of 0.9811.

4.2 Detail categories

The results of the detail category labelling of tokens occurring in both data sets are described next. In order to get an idea of the distribution of the detail category labels across the different data sets, the features of the detail category word (combinations) are visualized. For each data set, a fastText model is trained, and the feature vectors of the detail category tokens are retrieved. If the detail category consists of multiple words, the mean of the

corresponding feature vector matrix is taken over the rows of the matrix. The feature vectors of the tokens are dimension reduced using the UMAP method, which is motivated in section 2.7.1.

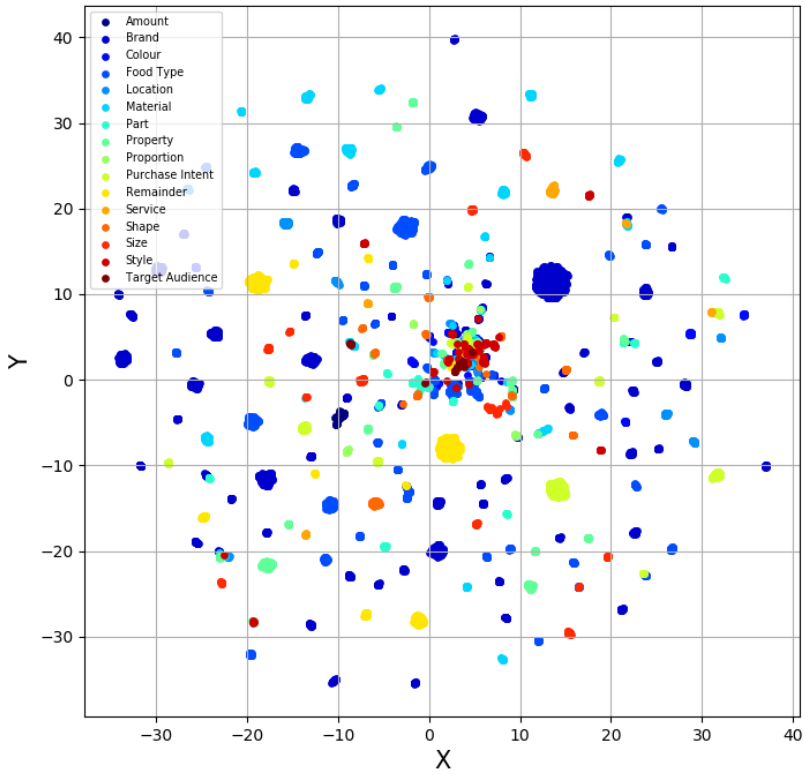

Figure 13: Barbecue data set visualization

The visualization of the features of the the 9773 tokens which have a detail category label, occurring in the queries of the barbecue data set, is presented in figure 13. There can be observed that the most of the detail categories consist of a set of distinct clusters. The clusters observed in the visualization consist of hundreds of word (combinations). These clusters consist of semantically related words like synonyms and antonyms. The clusters belonging to a detail category are scattered over the visualization.

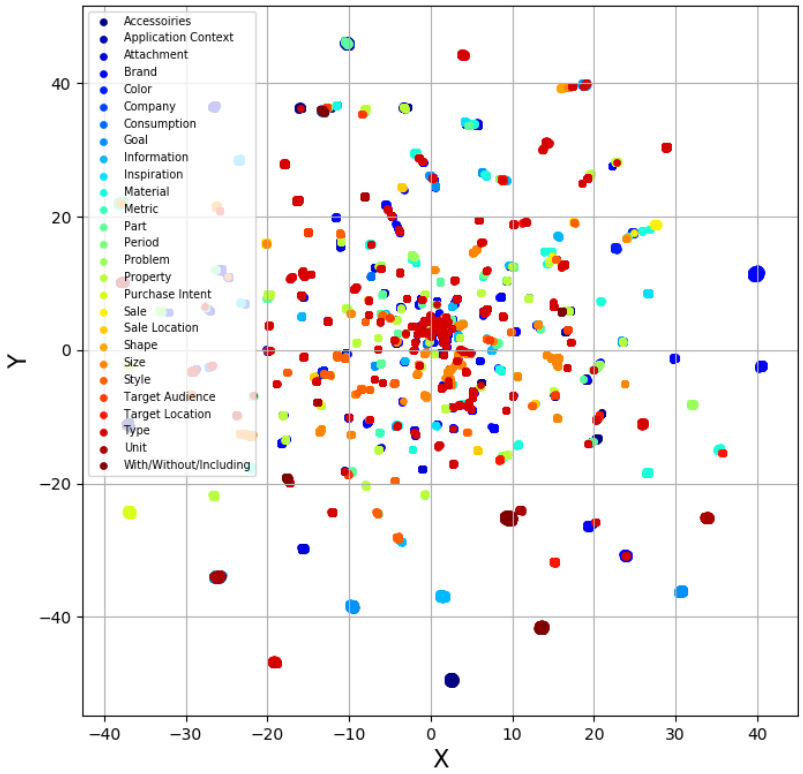

Figure 14: Shower data set visualization

The visualization of the features of the 51751 tokens which have a detail category label, occurring in the queries of the shower data set, is presented in figure 14. Again, the visualization consists of smaller clusters with closely semantic related words. The clusters making up a category are scattered over the visualization. The clusters are more segmented than the clustering of the detail categories of the barbecue data set.

The research continues with examining which word pre-processing method helps delivering the best results on the task of labelling query words with detail category labels. In order to answer this question, the pre-processing methods used for word inflection reduction during the experiments must be

examined. The resulting F1 scores of the 64 experiments on both data sets are grouped on the used pre-processing method. From these groups, the mean and maximum scores are calculated, and visualized in figure 15. The mean scores tell us which method performs generally better, while the maximum score help contribute to constructing an optimal feature configuration. For clarification, the scores are noted above the respective groups.

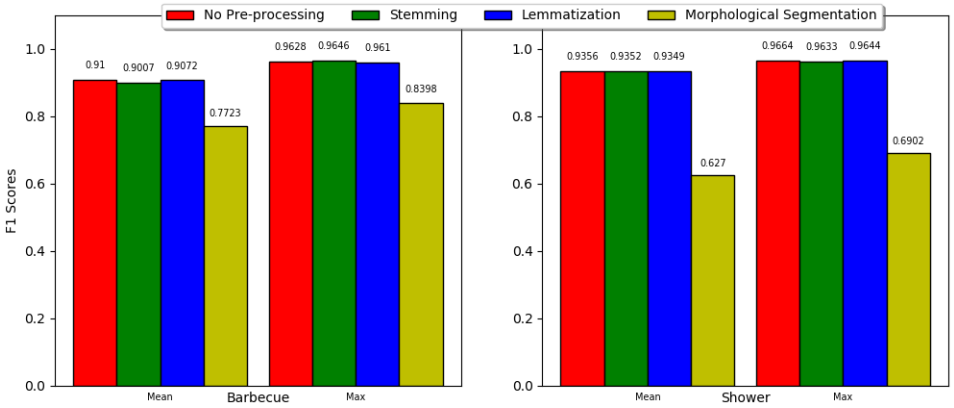

Figure 15: The average and best classifier performance using the different pre-processing methods

From the results can be concluded that stemming contributes to the highest score on the barbecue data set, while omitting pre-processing contributes to the highest score on the shower data set. They are closely followed by the other pre-processing methods, except morphological segmentation. Performing morphological segmentation yields far worse results than the other pre-processing methods, especially on the shower data set. There can be concluded that the additional effort of performing pre-processing does not result in substantially better results. Therefore, omitting pre-processing is the best choice.

Now the different model types used to create token features are evaluated. The results of the 64 experiments on each of the 2 data sets are grouped on the used model type. The resulting mean and maximum F1 scores of each group are visualized in figure 16 for both data sets.

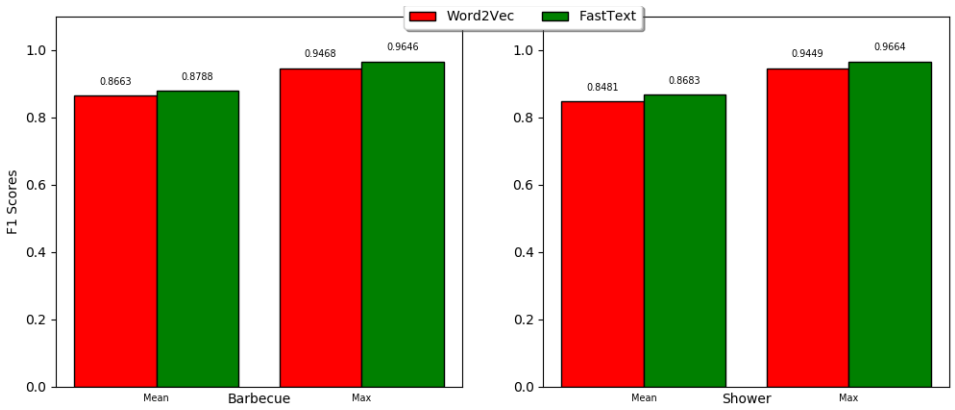

Figure 16: The average and best performance of the classifiers using the different model types

From the results can be concluded that the fastText model performs better than the word2vec model. Both the average, and the best scores of classifiers using fastText models are higher than classifiers using the word2vec counterpart. The difference between the best performing word2vec-using model and the best performing fastText model is as high as 2 percent on the barbecue data set. There can be concluded that fastText performs substantially better than wordvec.

Next step is to examine the performance of the classifiers when the features have been enriched by using a large domain-independent data corpus which consists of all Wikipedia articles. The F1 scores over all experiments are grouped on the case whether additional data has been used to create the feature. The mean and maximum scores of the groups are calculated for both data sets, and presented in figure 17.

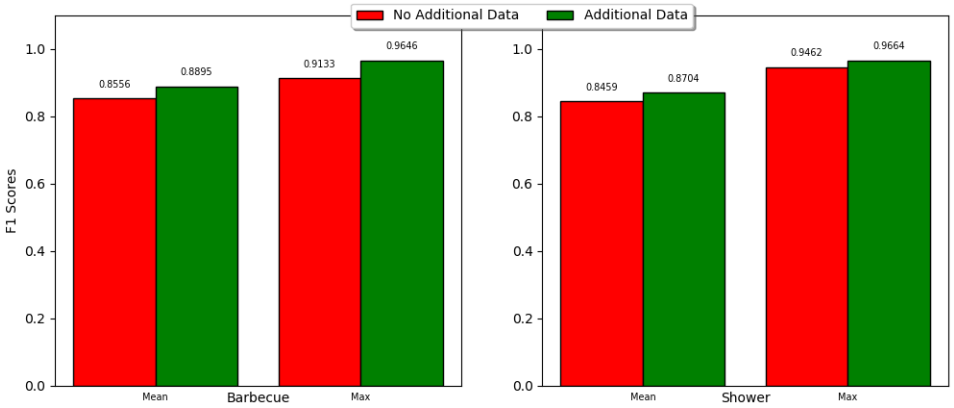

Figure 17: The classifier performance when using additional data

From the results can be concluded that both the average and the best performance of the classifiers are higher when a fastText model is used instead of a word2vec model. The increase in maximum performance is as high as 5 percent on the barbecue data set. There can be concluded that the fastText model yields substantially better results than the word2vec model.

Finally, the performance of the different classifier variants on the labelling tokens with detail category labels is investigated. First, the results of the 64 experiments on the two data sets are grouped on used classifier variant. Then the mean and maximum scores are calculated for each group, and the results are presented in figure 18.

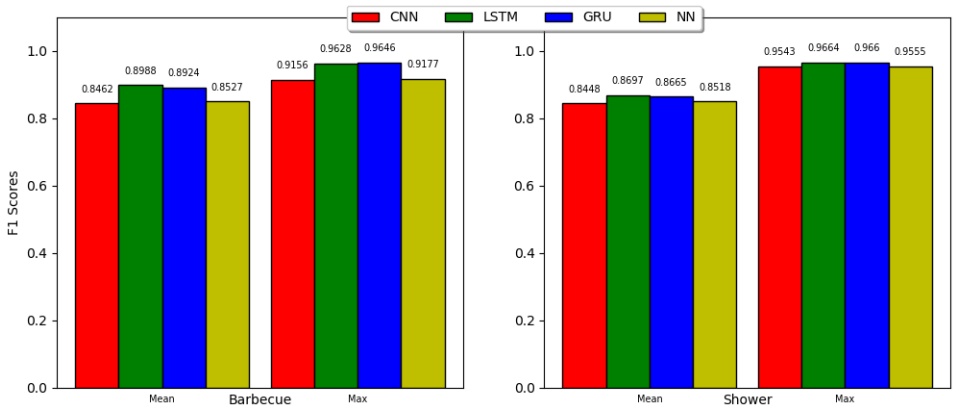

Figure 18: Performance when using different classifiers

From the results can be concluded that again the recurrent LSTM and GRU classifiers perform substantially better than the NN and the CNN. Which of the classifiers perform best differs slightly over the two data sets: the GRU performs slightly better than the LSTM on the barbecue data set, while the LSTM performs slightly better than the GRU on the shower data set. There can be concluded that the recurrent models perform substantially better than the NN and the CNN on the task of labelling words in queries with detail categories.

The optimal configuration of which feature creation methods to use in combination with which classifier differs slightly over the two data sets. For the barbecue data set, the optimal configuration consists of performing stemming, training a fastText model and enriching features using additional data in combination with a GRU resulting in a F1 score of 0.9646. The performance of stemming is not substantially higher than the performance when pre-processing is omitted, which results in a optimal F1 score of 0.9629. For the shower data set, the optimal configuration consists of omitting preprocessing, training a fastText model and enriching features using additional data in combination with a LSTM resulting in a F1 score of 0.9664.

5 Conclusion

In this chapter, conclusions are drawn based on the obtained results. Answers to the research sub-questions are found, and used to answer the research question. Finally, the research is discussed, and recommendations for future research are given.

Based on the obtained results, there can be concluded that the removal of word inflection does not substantially increase the quality of a word feature when labelling words with both global and detail category labels. There can be concluded that the word features of inflectional word forms are distinctive enough for the classifiers to obtain great results.

From the results can be concluded that the fastText model outperforms the word2vec model in labelling words with both global and detail category labels. The differences in performance are substantial, with an observed performance improvement of 1 percent when labelling words with global category labels, and 2 percent when labelling words with detail category labels. Most likely the fact that fastText is able to derive word embeddings of unseen words by taking into account n-grams of word characters, leads to this improvement.

The usage of an additional data corpus in order to enrich the word features has also lead to an improvement in the results by a slight margin when labelling words with the label variants. Especially when labelling query words with detail category labels, an improvement in the F1 score of the best performing classifier as high as 5 percent is observed on the barbecue data set. The improvements on labelling words with detail categories can be explained by the infrequent occurrence of certain words in the data set, resulting in their word embeddings not being of an optimal quality.

From the four examined classification methods, the recurrent neural networks perform significantly better than the default and convolutional neural networks. The fact that the recurrent models outperform their counterparts, can be explained by the fact that the recurrent models take into account surrounding words when assigning labels to words. Subtle differences are observed between the GRU and the LSTM on the two data sets. The difference in performance within the group of recurrent models, and within the group of the NN and the CNN is marginal. Because the difference is so minimal, no recurrent model can be appointed to be the best performing one.

Finally, for the labelling of tokens occurring in queries with both global and detail category labels, a optimal configuration has been found. The optimal configuration consists of combining token features from a fastText model trained on the data set with a model trained on an additional corpus, while omitting word pre-processing. This feature creation configuration is combined with a recurrent model like the LSTM for optimal sequence labelling performance.

5.1 Discussion and Limitations

While the results of the performed experiments are outspoken, some comments need to be made in terms of the abilities of the systems. First of all, the used embedding models are able to capture only one word meaning. When the word embedding of a homonym is looked up in word2vec or fastText, the word embedding will not always match the contextual meaning of the corresponding token.

Furthermore, the recurrent models only see words which occur before them in the query, not the tokens which occur after them. When important components of a multi-word label occur in a query, only for the last component the classifier takes into account all components, for label assignment.

5.2 Future Work

Although the results of this research are promising, there is still room for improvement. Therefore recommendations for future research are provided in this section. First of all, optimizations are possible in the process of generating feature vectors. The strength of an enriched feature vector can be optimized by utilizing transfer learning to fit the model trained on an additional data set to the domain-specific data. Furthermore, while the word2vec and fastText word embedding models deliver great results, recently more advanced models have been introduced. It is interesting to see whether state-of-the-art models like ELMo [18] and BERT [19] help further improving the labelling scores.

Secondly, now that it is clear that enriching word embeddings with additional Wikipedia data improves the classification, the addition of different domain-specific and domain-independent data sources can be examined in order to examine whether they help increase the performance even further.

Finally, optimizations are possible in the classifier used to classify the features. Now that is known that recurrent models perform best on the task of

labelling query tokens, it is interesting to check whether additions like making the RNN bi-directional [25] improve the performance baseline created in this research. Furthermore combining this research with other classification methods like CRFs [24] might help further improving the results.

Appendices

A Barbecue Product Data Set

Query	Global Category	Detail Category
weber **barbecue**	**Barbecues**	Brand
bbq tools sale	**Barbecue Tools**	Purchase Intent
best **bbq book**	**Barbecue Accessories**	
buy **barbecue furniture**	**Barbecue Furniture**	Purchase Intent
easyblue **gas bottle** price	**Gas Barbecue Accessories**	Brand, Purchase Intent
nice **pizza cutter**	**Pizza Tools**	Style
grandhall **pizza stone**	**Pizza Accessories**	Brand
pizzarette **stone oven**	**Pizza Oven**	
smoke oven sale	**Smoke Oven**	Purchase Intent
wood shavings intratuin	**Smoke Accessories**	Brand
white **charcoal**	**Coal and Lighting Accessories**	Color
action **bbq cover**	**Protective Covers**	Brand
weber **barbecue grid cleaner**	**Cleaning Accessories**	Brand

Table 5: Queries labelled with global and detail categories originating from the barbecue data set

B Shower Product Data Set

Query	Global Category	Detail Category
drain plug **wash bowl**	**Sink**	Attachment
design **bathroom radiators**	**Heating**	Style
door lock **toilet**	**Toilet**	Attachment
antique **bathroom mirror**	**Mirror**	Style
bruynzeel **hot tub**	**Bath**	Brand
non-slip **shower tray**	**Shower**	Type
aquavive **shower faucet**	**Tap**	Brand
bathroom furniture standing	**Furniture**	Mount
bamboo **bathroom** accessoiries	**Remaining Products**	Material, Accessoiries

Table 6: Queries labelled with global and detail categories originating from the shower data set

C Data Explorer Application

As part of this research, an application has been developed for New-Media. The application receives a data set with search queries as input, and outputs the data set where the query words are labelled with global and detail categories. In short, the application performs tokenization, trains a word embedding model, and is able to train a classifier based on an annotated data set, and use this classifier for label prediction. The words in the search queries for who a label is predicted, are dimension reduced and visualized for examination by a domain expert. In figure 19, a visualization is presented of the word (combinations) in the barbecue for which the classifier predicts a detail category label. The most basic functionality of the application is discussed next.

The visualization shows a plot with a set of clusters containing points which correspond to word (combinations). Each cluster color corresponds to one detail category label. The size of the point corresponds to the number of occurrences of this word (combination) in the queries. New clusters can be created, and existing clusters can be expanded. When a point is clicked, its cluster becomes selected, and its detail category label is presented in the text box to the right of the graph. The 'Text' and 'Label' check boxes let the domain expert declare whether the point word (combinations), and/or point label suggestions are visualized. Finally, the point label suggestions of selected clusters can be updated by inserting a new label in the text area in the top of the screen, and clicking the 'Label' button.

The sliders on the bottom of the interface are used to control the number of points which are visualized. The 'H' slider indicates that only the points are visualized for which the classifier is h percent sure that their corresponding word combination has a certain label, where h is the position of the 'H' slider. Therefore, when the slider is dragged to the left, more points are visualized: the classifier is less sure of the label prediction of the new word (combinations). The 'N' slider controls the total number of visualized points. When the 'N' slider is dragged, the number of visualized points increases or decreases, in order to present the domain expert with a more complete, or simpler overview.

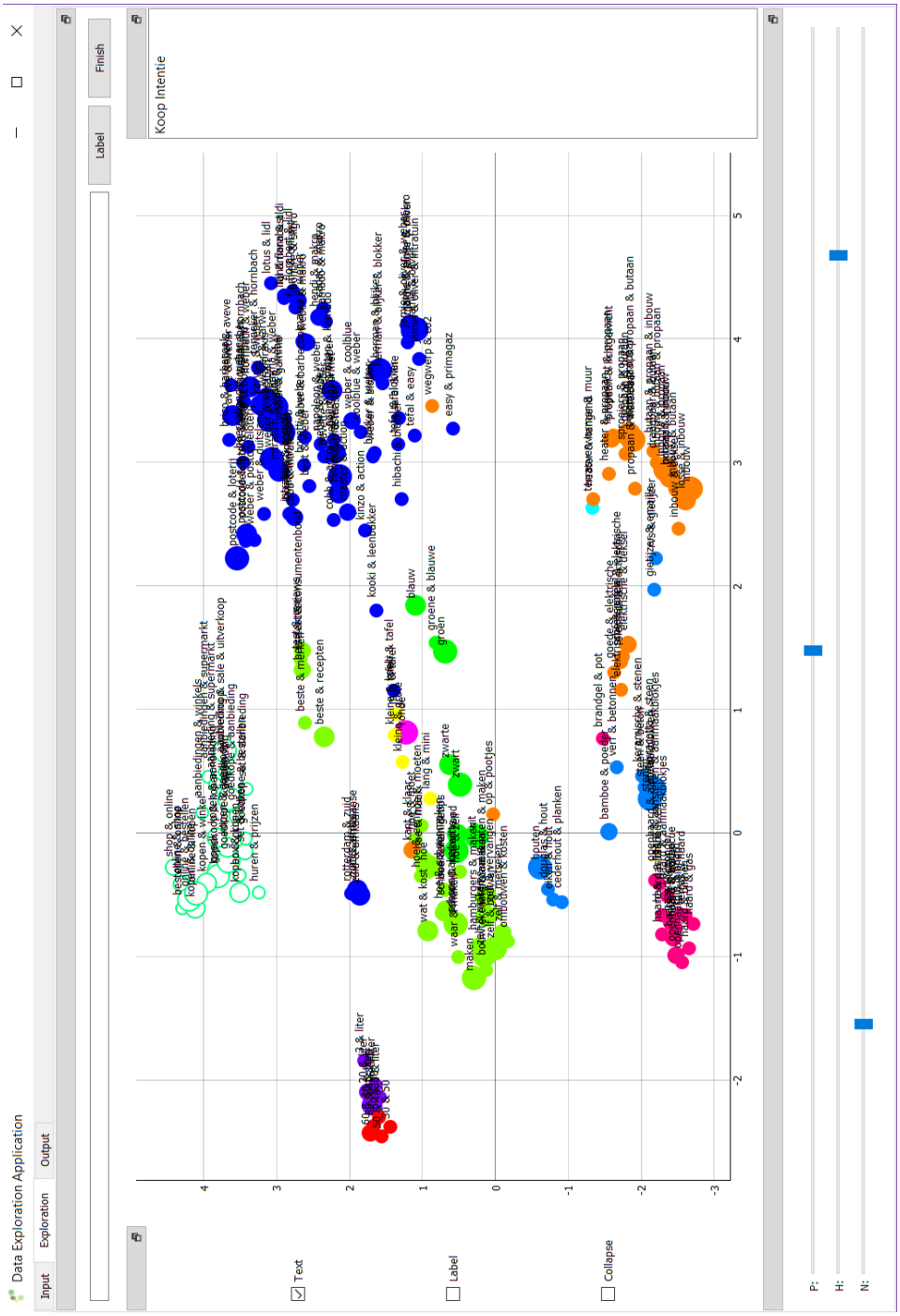

Figure 19: The data explorer application clustering query words from the barbecue data set based on detail category labels

References

[1] Google search analytics. `https://developers.google.com/webmaster-tools/search-console-api-original/v3/searchanalytics`. Accessed: 2018-05-16.

[2] New-media. `http://www.new-media.nl/`. Accessed: 2018-05-04.

[3] Kasper Welbers, Wouter Van Atteveldt, and Kenneth Benoit. Text analysis in R. *Communication Methods and Measures*, 11(4):245–265, 2017.

[4] André B. Bondi. Characteristics of scalability and their impact on performance. In *Proceedings of the 2Nd International Workshop on Software and Performance*, WOSP '00, pages 195–203, New York, NY, USA, 2000. ACM.

[5] François Chollet et al. Keras. `https://keras.io`, 2019.

[6] Chowdhury Gobinda G. Natural language processing. *Annual Review of Information Science and Technology*, 37(1):51–89.

[7] Martin F Porter. Snowball: A language for stemming algorithms. `http://snowball.tartarus.org/texts/introduction.html`, 2001. [Online; last accessed 26-01-2019].

[8] Chris D. Paice. Another stemmer. *SIGIR Forum*, 24(3):56–61, November 1990.

[9] Tanja Gaustad. *Linguistic Knowledge and Word Sense Disambiguation*. PhD thesis, University of Groningen, 2004.

[10] Antal van den Bosch, Bertjan Busser, Sander Canisius, and Walter Daelemans. An efficient memory-based morphosyntactic tagger and parser for dutch. *LOT Occasional Series*, 7:191–206, 2007.

[11] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 3111–3119. Curran Associates, Inc., 2013.

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

[13] Robert Speer, Joshua Chin, and Catherine Havasi. Conceptnet 5.5: An open multilingual graph of general knowledge. *CoRR*, abs/1612.03975, 2016.

[14] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146, 2017.

[15] Jiwei Li and Dan Jurafsky. Do multi-sense embeddings improve natural language understanding? In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1722–1732. Association for Computational Linguistics, 2015.

[16] Dai Quoc Nguyen, Dat Quoc Nguyen, Ashutosh Modi, Stefan Thater, and Manfred Pinkal. A mixture model for learning multi-sense word embeddings. *CoRR*, abs/1706.05111, 2017.

[17] Eric H. Huang, Richard Socher, Christopher D. Manning, and Andrew Y. Ng. Improving word representations via global context and multiple word prototypes. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers - Volume 1*, ACL '12, pages 873–882, Stroudsburg, PA, USA, 2012. Association for Computational Linguistics.

[18] Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. In *Proc. of NAACL*, 2018.

[19] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[20] Marten Postma, Emiel van Miltenburg, Roxane Segers, Anneleen Schoen, and Piek Vossen. Open Dutch WordNet. In *Proceedings of the Eight Global Wordnet Conference*, Bucharest, Romania, 2016.

[21] H. Liu and P. Singh. Conceptnet — a practical commonsense reasoning tool-kit. *BT Technology Journal*, 22(4):211–226, October 2004.

[22] Michael Fleischman and Eduard Hovy. Fine grained classification of named entities. In *Proceedings of the 19th International Conference on Computational Linguistics - Volume 1*, COLING '02, pages 1–7, Stroudsburg, PA, USA, 2002. Association for Computational Linguistics.

[23] Qikang Wei, Tao Chen, Ruifeng Xu, Yulan He, and Lin Gui. Disease named entity recognition by combining conditional random fields and bidirectional recurrent neural networks. *Database*, 2016, 2016.

[24] John Lafferty, Andrew McCallum, and Fernando CN Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. 2001.

[25] Mike Schuster and Kuldip K Paliwal. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681, 1997.

[26] Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. Neural architectures for named entity recognition. *CoRR*, abs/1603.01360, 2016.

[27] Sida Wang and Christopher D Manning. Baselines and bigrams: Simple, good sentiment and topic classification. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Short Papers-Volume 2*, pages 90–94. Association for Computational Linguistics, 2012.

[28] Ye Zhang and Byron C. Wallace. A sensitivity analysis of (and practitioners' guide to) convolutional neural networks for sentence classification. *CoRR*, abs/1510.03820, 2015.

[29] Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12(Aug):2493–2537, 2011.

[30] Yoon Kim. Convolutional neural networks for sentence classification. *CoRR*, abs/1408.5882, 2014.

[31] Daniel Cer, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St. John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, Yun-Hsuan Sung, Brian Strope, and Ray Kurzweil. Universal sentence encoder. *CoRR*, abs/1803.11175, 2018.

[32] Quoc Le and Tomas Mikolov. Distributed representations of sentences and documents. In *International Conference on Machine Learning*, pages 1188–1196, 2014.

[33] Nut Limsopatham and Nigel Collier. Modelling the combination of generic and target domain embeddings in a convolutional neural network for sentence classification. In *Proceedings of the 15th Workshop on Biomedical Natural Language Processing*, pages 136–140. Association for Computational Linguistics, 2016.

[34] David M Blei, Andrew Y Ng, and Michael I Jordan. Latent dirichlet allocation. *Journal of machine Learning research*, 3(Jan):993–1022, 2003.

[35] Christopher E. Moody. Mixing dirichlet topic models and word embeddings to make lda2vec. *CoRR*, abs/1605.02019, 2016.

[36] Ronan Collobert and Jason Weston. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th International Conference on Machine Learning*, ICML '08, pages 160–167, New York, NY, USA, 2008. ACM.

[37] Wenpeng Yin, Katharina Kann, Mo Yu, and Hinrich Schütze. Comparative study of CNN and RNN for natural language processing. *CoRR*, abs/1702.01923, 2017.

[38] C. Amrit and J. Hek. Clustering the results of brainstorm sessions: Applying word similarity techniques to cluster dutch nouns. In *2016 49th Hawaii International Conference on System Sciences (HICSS)*, pages 4232–4241, Jan 2016.

[39] Ergin Altintas, M. Elif Karsligil, and Vedat Coskun. A new semantic similarity measure evaluated in word sense disambiguation. In *NODALIDA*, 2005.

[40] Zhibiao Wu and Martha Palmer. Verbs semantics and lexical selection. In *Proceedings of the 32Nd Annual Meeting on Association for Computational Linguistics*, ACL '94, pages 133–138, Stroudsburg, PA, USA, 1994. Association for Computational Linguistics.

[41] Jasper Snoek, Hugo Larochelle, and Ryan P Adams. Practical bayesian optimization of machine learning algorithms. In *Advances in neural information processing systems*, pages 2951–2959, 2012.

[42] James Bergstra and Yoshua Bengio. Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, 13(Feb):281–305, 2012.

[43] James S Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. Algorithms for hyper-parameter optimization. In *Advances in neural information processing systems*, pages 2546–2554, 2011.

[44] Ian Jolliffe. Principal component analysis. In *International encyclopedia of statistical science*, pages 1094–1096. Springer, 2011.

[45] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605, 2008.

[46] Leland McInnes and John Healy. Umap: Uniform manifold approximation and projection for dimension reduction. *arXiv preprint arXiv:1802.03426*, 2018.

[47] Radim Řehůřek and Petr Sojka. Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 45–50, Valletta, Malta, May 2010. ELRA. `http://is.muni.cz/publication/884893/en`.

[48] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781, 2013.

[49] Stephan Tulkens, Chris Emmery, and Walter Daelemans. Evaluating unsupervised Dutch word embeddings as a linguistic resource. In Nicoletta Calzolari (Conference Chair), Khalid Choukri, Thierry Declerck, Marko Grobelnik, Bente Maegaard, Joseph Mariani, Asuncion Moreno, Jan Odijk, and Stelios Piperidis, editors, *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*, Paris, France, may 2016. European Language Resources Association (ELRA).

[50] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[51] Sungheon Park and Nojun Kwak. Analysis on the dropout effect in convolutional neural networks. In *Asian Conference on Computer Vision*, pages 189–204. Springer, 2016.

[52] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[53] Tom Fawcett. Roc graphs: Notes and practical considerations for researchers. *Machine Learning*, 31:1–38, 01 2004.

[54] Victoria López, Alberto Fernández, Salvador García, Vasile Palade, and Francisco Herrera. An insight into classification with imbalanced data: Empirical results and current trends on using data intrinsic characteristics. *Information sciences*, 250:113–141, 2013.

[55] László A Jeni, Jeffrey F Cohn, and Fernando De La Torre. Facing imbalanced data–recommendations for the use of performance metrics. In *Affective Computing and Intelligent Interaction (ACII), 2013 Humaine Association Conference on*, pages 245–251. IEEE, 2013.

[56] Yuchun Tang, Yan-Qing Zhang, Nitesh V Chawla, and Sven Krasser. Svms modeling for highly imbalanced classification. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 39(1):281–288, 2009.

[57] Tom Fawcett. An introduction to ROC analysis. *Pattern Recognition Letters*, 27(8):861 – 874, 2006. ROC Analysis in Pattern Recognition.

[58] Nathan Greenberg, Trapit Bansal, Patrick Verga, and Andrew McCallum. Marginal likelihood training of bilstm-crf for biomedical named entity recognition from disjoint label sets. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2824–2829, 2018.

[59] Hanieh Poostchi, Ehsan Zare Borzeshi, and Massimo Piccardi. Bilstm-crf for persian named-entity recognition armanpersonercorpus: the first

entity-annotated persian dataset. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC-2018)*, 2018.