Review Spam Detection Using Deep Learning

A Comparison of Linear Classifiers and Convolutional Neural Networks

A Thesis Submitted for the Degree of

Master of Science (M.Sc.)

In Information Systems

At the School of Business & Economics
Humboldt-Universität zu Berlin

Submitted by
Martin Müller
Matr. No.: 539010

Supervisor:
Prof. Dr. S. Lessmann
Institute Information Systems
School of Business & Economics
Humboldt-Universität zu Berlin

Berlin, 28 December 2016

Table of Content

List of Figures

List of Tables

1 Introduction

Since the early years of the internet, spam has always been a side-effect of this global and public platform. Most people probably associate the word spam with annoying e-mails, but other forms of spam do exist. Review spam is a phenomenon encountered at online platforms where participants can rate and express their opinion on a particular product or service. Spammers leave untruthful reviews in order to promote their own or denigrate competitors' offerings. This type of spam has not achieved much attention in the academic literature so far.

In this work, we take a closer look at the state of academic research on this topic with the objective of filling some of the gaps identified. There are several obstacles to overcome when doing research on review spam. Firstly, real-world data is mostly unlabelled, i.e. the use of either ex post labelling techniques or unsupervised learning is necessary. Our analysis is based on two real-world data sets from Amazon and Yelp, where we identify untruthful reviews using a labelling technique based on reviews' Jaccard similarity. As far as we know, with about 50.000 and 10.000 fake reviews detected in each data set, respectively, this is the largest set of review spam used in academic research so far.

Existing studies take into consideration different features, models or data, which limits the comparability of their results. We employ a wide range of features, pre-processing techniques and models in order to find promising techniques to successfully detect review spam. This includes sophisticated deep-learning models that recently produced state-of-the-art benchmarks on related text classification tasks. In addition, we make use of the large amount of unlabelled data and train continuous word representations from scratch using a *word2vec* algorithm. These highly problem-specific word vectors improve the performance of our deep-learning model, but eventually we find that a simple linear classifier based on 2-gram bag-of-word input features outperforms most other approaches, including our deep-learning models.

In the following chapter, we first summarize existing literature on the topic of review spam, in order to quantify the scope of this phenomenon and identify promising detection techniques. Subsequently, we introduce the data our analysis is based on and elaborate on the methodology to identify spam in the formerly unlabelled data, which is a necessary condition for using supervised machine learning approaches. In the fourth chapter we will discuss recent developments in the area of deep-learning, which are relevant to review spam detection, followed by the final section that describes our experiments.

2 Literature Review

Since the early years of the internet, spam has always been a side-effect of this global and public platform. Most people probably associate the word spam with annoying emails that advertise everything from gambling to medical substances. Other forms of spam include the posting of unrelated advertisements in forums or newsgroups and the publication of keyword-heavy articles in order to influence search engine rankings. Whereas these relatively crude forms of spam seem to have been contained relatively well with improved filtering techniques, more sophisticated and subtle forms of spam, such as untruthful online reviews of products and services, remain a major problem.

2.1 Economic Importance of Review Spam

Review spam detection has been somewhat less in the center of academic attention than other types of spam detection. This might be due to their relatively small importance during the early times of the internet compared to email or web spam, for example. But the number of reviews for products and services available on the web has increased steadily and some estimates see the share of faked opinions on the web as high as one third of the total.[1] Some businesses rely on feedback merely to help customers in the decision making process before an actual purchase, but other business models completely depend on reviews as their main asset, such as hotel or restaurant recommendation websites. Since reviews are now ubiquitous on the web and accessing this information comes at little to no cost, online reviews play an important part in consumers' purchasing process. Although sales of stronger brands' products are less influenced by online reviews, especially weaker brands can profit strongly from positive reviews, which indicates that consumers rely more on them when perceived knowledge about a brand is lower. A surge in positive reviews can thus result in a virtuous cycle of more sales and potentially more positive reviews.[2] Hence, there are significant incentives for businesses to promote their reputation using fake positive reviews, as well as to degrade the reputation of a direct competitor. It has been shown that businesses are indeed more likely to engage in faking when economic pressure increases. Restaurants are more likely to engage in faking if they recently received bad reviews, have little reputation in general or face competition from newly opened venues of the same type of cuisine (but not a different one). Again, small privately owned restaurants and hotels are more likely to fake than bigger chains, where reputation is mainly built through marketing and branding

[1] Streitfeld (2012)
[2] Ho-Dac et al. (2013), p.37ff

campaigns.[3] The share of untruthful reviews has steadily increased over recent years, at least at Yelp, one of the largest restaurant and hotel recommendation website.[4] Affected companies are responding: One of the world's biggest online retailers, Amazon, is continuing to sue individuals and companies, which offer to post untruthful reviews on behalf of sellers as a paid service.[5] The share of suspicious and thus filtered reviews at Yelp is about 16%.[6] Another study suggests that especially review communities where posting costs are lower, i.e. anyone can post reviews without having to purchase a product or service, have seen a rising share of deceptive reviews in recent years. The authors estimate the share of such reviews at one to six percent for different platforms.[7]

Even without knowing the exact magnitude of the deceit, it becomes clear that the economic importance of review spam is not at all insignificant. Consumers, review platforms and honest businesses are all on the losing side as faking businesses erode trust in the system for their own short-term gain. Hence the need to fight these malicious practices appropriately. Effective automated detection techniques are an important part of the solution, as they are cheap, scalable and can easily be integrated into existing software without raising the bar for honest reviewers to leave their feedback.

2.2 Review Spam Detection Literature

As mentioned above, review spam has not received as much attention as other fields of spam detection. The first studies that cover the problem and possible approaches to automated detection emerged in the late 2000s.

Researchers face several decisions when trying to classify spam reviews. Will data be scraped from websites or artificially created (e.g. by hired workers)? Raw data is always unlabelled, that means a decision on whether unsupervised algorithms or any kind of labelling strategy will be used has to be made. Furthermore, feature engineering plays an important role in the modelling process and numerous ways of encoding text into numerical features have been developed by NLP research. Finally, the modeller has several models to choose from and must pick a suitable performance metric.

[3] Ho-Dac et al. (2013), p.43ff
[4] Luca and Zervas (2015), p.6
[5] Kieler (2016)
[6] Luca and Zervas (2015), p.6
[7] Ott et al. (2012), p.207

2.2.1 Data Sources & Labelling Techniques

There are numerous websites that offer an enormous amount of reviews to analyse. Most popular among researchers are product reviews from Amazon, which hosts millions of reviews for products of all kinds.[8] Another common source is the Yelp recommendation platform. Here, consumers can leave feedback for restaurants, shops and other local services. The platform also discloses the filtered reviews for each venue, which researchers can use as a benchmark for their own classification results.[9] Additionally, well-known recommendation and hotel platforms such as Tripadvisor, epinions.com or booking.com are used to obtain authentic reviews.[10]

Regarding labelling, two approaches exist in the literature: In one strand, authors hire manual labour to create fake reviews for products or services, using online on-demand platforms such as Amazon Mechanical Turk. Truthful reviews are then scraped from real websites. Ott et al. (2011) use this technique to create a Gold-standard dataset of 400 fake reviews and an equal number of authentic ones scraped from a recommendation website. The data was subsequently used by several other authors.[11] This procedure comes with the benefits of complete knowledge about which reviews are counterfeit, but the number of obtained fake reviews is rather small, since manual labelling is costly. Furthermore, is has been shown that these types of reviews deviate from real-world fake reviews. Mukherjee et al. (2013a) compare the artificial Gold-standard data with supposedly similar fake reviews (on hotels in Chicago) obtained from Yelp's filtered reviews, and find that their model performs better on the real-world data by a large margin.[12] Other researchers try to extract labels from downloaded raw reviews. Based on specific similarity measures, they aim to find duplicate and near-duplicate reviews, which they mark as spam.[13] The underlying assumption is that (professional) spammers often reuse their reviews or at least alter them only minimally in order to save time. This method produces real-world untruthful reviews, but potentially misses cases that do not closely resemble other reviews (i.e. the more authentically looking untruthful ones). Models trained on this type of data might overfit on this particular type of fake review. It also has to be taken into account that some websites contain genuine duplicates by default, which have to be filtered beforehand. Amazon, for example, merges reviews from different versions of a product (e.g. ebook and hardcover). Morales et al. (2013) employ yet

[8] See Jindal & Liu (2008), Lau et al. (2011)
[9] Mukherjee et al. (2013b), p.3f
[10] Crawford et al. (2015), p.19
[11] See Banerjee & Chua (2014), Long et al. (2014)
[12] Mukherjee et al. (2013a), p.4f
[13] See Jindal & Liu (2008), Li et al. (2011)

another method. They develop an algorithm that newly produces fake reviews from a set of unique online reviews.[14] Some authors also use unsupervised and semi-supervised learning in order to alleviate the drawbacks of aforementioned labelling techniques.[15]

2.2.2 Feature Engineering

Features or predictors are the essential ingredient to basically every statistical model. In most supervised approaches the values of features for each corresponding observation are fed into the model and the learning algorithm optimizes its parameters (or weights) based on a specific metric. Thereafter, the model is able to produce a prediction for unseen data points using the observations' feature values and its internal parameters determined during the training process. The way how features are represented can have a significant influence on model performance and finding optimal representations is a vital part in the model building process. For example, the combination of two predictors, such as their ratio, can be more informative to the model than the two individually.[16] Especially in NLP problems, feature engineering is vital in order to train successful models, since data is mostly represented as raw text, unsuitable as input to machine learning algorithms.

Other authors dealing with review spam detection make use of a wide array of features. Whereas Jindal and Liu (2008) and Li et al. (2011) use only manually created features based on the review, reviewer and product characteristics,[17] Ott et al. (2011) and base their analysis solely on features extracted from the raw review text by employing bag-of-words, Part-of-Speech (POS) tagging and Linguistic Inquiry and Word Count (LIWC).[18] For example Mukherjee et al. (2013a), use a combination of both approaches.[19]

One traditional and still often used approach in text classification is called bag-of-words. Here, documents are represented as the set of words that occur in them. Each feature corresponds to a specific word and simply indicates whether it was found in the document or not. Hence, the feature set consists of all words that occur in the training corpus. Obviously, in this simple representation it cannot be accounted for any other relation between words than their co-occurrence in individual documents. The raw corpus can be altered using techniques such as stop word removal, stemming, filtering of infrequent words and removing of case and

[14] Morales et al. (2013), p.1088ff
[15] See Li et al. (2014), Lau et al. (2011)
[16] Kuhn & Johnson (2013), p.27
[17] Jindal & Liu (2008), p.223ff; Li et al. (2011), p.2490f
[18] Ott et al. (2011) p.313f,
[19] Mukherjee et al. (2013a), p.3f

punctuation. This was mainly used to reduce data size and dimensionality, for which the need has vanished due to the rapid increase in computing power and storage capacity.[20]

POS Tagging is another popular way to add more information to raw text. There is a fixed set of POS tags and each word in a text belongs to one category. These tags can then aid the modelling process, as models are able to distinguish between words being used as noun or verb, for example, in the respective document.[21]

In contrast, LIWC is rather a single tool than a general technique of feature engineering. It is based on a dictionary of several thousand words, each of them belonging to a set of hierarchically organized categories that describe it linguistically. Main categories include "summary language variables" (e.g. analytical, tonality), "Linguistic Dimensions" (e.g. pronouns, prepositions) or "Psychological Processes" (e.g. positive/negative emotion, cognitive processes). For each word the tool imports from a given text, the count for corresponding categories is incremented. This results in 89 output dimensions, which capture the various emotional, cognitive and structural components present in the text.[22]

A large number of yet other features can be manually created based on metadata. These features can convey a lot of predictive power, but have to be manually designed for each specific task and possibly require some domain knowledge. In review spam detection, metadata can be obtained from several entities. With regard to Amazon for example, it is possible to include characteristics of the review itself (word length, rating, helpfulness etc.), reviewer (No. of votes, average rating, rating deviation etc.) and product (price, average rating etc.). Available meta-characteristics differ by website, of course, but essentially these three categories persist. As mentioned before, it is up to the modeller to create new predictors from available ones which potentially improve model performance during the feature engineering process. Different authors come up with varying opinions on which features to include. Jindal and Liu (2008) include by far the largest number of hand-crafted features (which is necessary since they refrain from using BOW, POS etc.). They use 22 features overall, including characteristics of the review, e.g. a dummy whether the review was the first or the only review for the corresponding product, the ratio of helpful to overall votes and the rank of the review by time.

On reviewer level, they include variables such as the share of first votes on a product by the reviewer and whether he has posted only good reviews. They further include the product's price and sales rank for each review. Finally, for all reviewer and product, they calculate the

[20] Scott (1999), p. 2
[21] Voutilainen (2003), p.220
[22] Pennebaker et al. (2015), p.1ff

average and standard deviation of rating given or received. The same set of features is used by Lau et al. (2011). Li et al. (2011) and Mukherjee et al. (2013a) create further features such as the percentage of negative and positive words in the reviews text, time between first and last post of the reviewer and number of reviews for the product in order to complement the BOW approach. Table 8.1 in the Appendix provides an overview of which exact features where used by which authors.

2.2.3 Models

In the next step, the created features are used to train a classifier which aims to effectively separate truthful reviews from fraudulent ones. There are actually hundreds of classifiers available, but the performance of many of them is in many cases not superior to the simple and well-established ones.[23] Consequently, most authors in the review spam literature use simple linear out-of-the-box models with no meta-parameters to tune. Commonly used are logistic regression (LR), linear support-vector machines (SVM) and naïve Bayes (NB) classifier. Jindal and Liu (2008) for example test these three classifiers and find that LR performs best on their data. Ott et al. (2011) and Mukherjee et al. (2013a) on the other hand, find that SVM with a linear kernel yields the best performance. Morales et al. (2013) use an ensemble of both SVM and NB. However, some authors create models which are supposed to be more suited for the task at hand. For example, Li et al. (2011) use semi-supervised co-learning algorithm besides a traditional NB classifier to make use of the large pool of unlabelled data and find that this model outperforms NB by a relatively large margin (5 – 8%). Most notably Lau et al. (2011) employ an interesting approach where the similarity between reviews is used directly as input to the model instead of identifying near-duplicates from the rest. Instead, their semantic language model predicts the likelihood of any review being a semantically copied version of an existing one in order to detect review spam.[24] Based on a pre-defined text window, the model predicts the subsequent word or sequence of words. The authors' model aims at predicting the semantic similarity between reviews, i.e. it can be accounted for substituted words when looking for duplicates. Xie et al. (2012) use another unsupervised approach to review spam detection. They assume that the numbers of reviews as well as the ratings are relatively stable over time, and sudden bursts of very positive or negative reviews are likely to be initiated by spammers. The authors construct multi-dimensional time series and use a simple template matching algorithm on fitted curves to find

[23] Fernández-Delgado et al. (2014), p.3155ff
[24] Lau et al. (2011), p.254

simultaneous bursts in all dimensions.[25] Some authors use yet another approach representing the review eco-system of a platform as a network structure.[26] Akoglu et al. (2013) in particular represent it as a bipartite graph where products and reviewers represent nodes and reviews graph edges. They employ a two-step framework in order to identify spammers and spam reviews in an unsupervised manner. Lastly, also semi-supervised approaches are used for review spam detection. Li et al. (2014) compare a supervised SVM to positive-unlabelled (PU) learning based on a Chinese character review set. In contrast to SVM, the PU approach is able to better predict the review class and additionally identifies a large number of potential fake reviews in the unlabelled data part.[27]

2.2.4 Results

Most studies differ a lot in their experimental setting, i.e. they are based on different datasets, use different labelling methods, similarity measures and thresholds to filter for duplicates. Besides, extracted features and used classifiers often vary between studies. All of this makes a meaningful comparison of the results obtained by different authors difficult. Nonetheless, it is worth compiling them in order to provide an overview of what has been achieved with differing approaches and examine the current state-of-the-art benchmark.

For example, Lau et al. (2011) achieve a very high area under the curve (AUC) value in their experiment. However, they assemble their data by collecting reviews with high and low cosine similarity values and mark them as spam and non-spam, respectively. Later, they go on and classify reviews based on this measure, which is very likely to cause an upward bias in model accuracy.[28] Li et al. (2014) on the contrary, achieve a comparatively low F-score, but their analysis is based on Chinese character uni- and bigrams. As Chinese characters are usually not separated by whitespaces, doing so may reduce their linguistic information content.[29] Using only hand-crafted features, Jindal et al. (2008), who did one of the first studies on review spam, achieve an AUC of 0.78. They also find that review centric features are most helpful, and that using only text-based features actually produces the worst results. Their model is also able to effectively detect reviews that deviate a lot from the average rating of a product (outlier reviews). The authors see this as evidence for its ability to detect spam reviews which are no duplicates.[30] Ott et al. (2011) not only compare different features such

[25] Xie et al. (2012), p.825ff
[26] See Akoglu et al. (2013); Fayazbakhsh and Sinha (2012)
[27] Li et al. (2014), p.473
[28] Lau et al. (2011), p.24
[29] Li et al. (2014), p.471
[30] Jindal et al. (2008), p.225f

as POS, LIWC and n-grams (with a combination of LIWC and bigrams achieving the best performance and n-gram based features outperforming non-text features), but also show that simple classifiers are able to outperform humans by a large margin when it comes to review spam detection.[31] When it comes to classification based on Yelp reviews, Mukherjee et al. (2013a) also find that advanced features such as POS make only minor difference in performance compared to text-based features. They analyse hotel and restaurant reviews separately and achieve best results on both using POS-tagged bigrams as input features.[32] In their supervised setting with naïve Bayes classification, Li et al. (2011) achieve an F-Score of 0.583 using a wide range of features such as text-based, sentiment, reviewer centric and meta-data features.[33]

All in all, in most supervised training experiments, models are able to separate truthful and fraudulent reviews relatively well. The compiled studies will provide a benchmark for our future analysis. Table 2.1 summarizes selected spam review studies, showing their data sources, the number of spam reviews used for classification as well as the best results obtained.

Author	Data Source	#Spam Reviews	Classification Score
Jindal (2008)	Amazon	4488	0.78 (AUC)
Ott (2011)	AMT	400	0.898 (Accuracy)
Mukherjee (2013a)	Yelp	9170	0.681 (Accuracy)
Li (2011)	epinions	1398	0.583 (F-Score)
Lau (2011)	Amazon	2061	0.9986 (AUC)

Table 2.1 - Spam review detection studies, data sources and no. of spam reviews used for classification and best obtained classification score

The previous literate review yielded an overview of the state of research in the review spam field. At the same time, it emphasizes existing shortcomings in this research area. Firstly, the incomparability of various studies on the topic due to varying parameters such as data and labelling makes it hard to assess the global state-of-the-art and separate beneficial methods from the ones merely profiting from a set of experimental factors. Secondly, most authors focus on the stages of data preparation and feature engineering and rely on standard classification models to perform the actual prediction task. However, recent developments in the field of deep learning, have shown to be superior to these models especially for complex NLP and computer vision problems, at the same time reducing the potential need for costly

[31] Ott et al. (2011), p.315f
[32] Mukherjee et al. (2013a), p.3f
[33] Li et al. (2011), p.2492f

and problem specific manual feature engineering.[34] In subsequent chapters, we will therefore explain our ex post labelling strategy, give an introduction into deep learning techniques most relevant to NLP problems, and then compare various existing review spam detection approaches in a fixed experimental setting on different datasets. Finally, we will explore how a deep learning framework might be used to tackle this problem and whether it proofs superior to traditional models.

3 Data & Labelling

As outlined in the previous section, when it comes to review spam analysis and detection, most researchers rely on extracting real-world reviews from recommendation and e-commerce websites. Approaches to labelling untruthful reviews, in contrast, differ more across the literature. Our goal to deploy novel deep learning models to detect review spam entails the necessity for large training data. Data like the "gold-standard" set with 800 reviews in total as provided by Ott et al. (2011) would not suffice to train complex deep classifiers. The creation of thousands of artificial fake reviews by paid contractors is infeasible due to financial constraints. Thus, using real-world data and retrospectively detecting and labelling spam reviews is the only option to realise our experimental setup efficiently. The data used in our experimental setup was obtained from Amazon, and kindly provided by the Stanford University,[35] which enables us to analyse a large amount of data but deems time-consuming web scraping of reviews unnecessary. The total dataset consists of as much as 142 million reviews, starting from 1996 up to the year 2014. The university also offers smaller samples which contain reviews for individual categories. It has to be accounted for a practice of Amazon that actually results in many duplicate but genuine reviews: Some products are distributed in several forms under the same ASIN, e.g. a book can be sold as hardcover or e-book. Reviews belonging to the title are displayed for both versions, which results in the review for the same product by the same reviewer being downloaded twice. In the provided data, exact duplicates posted by the same reviewer are already removed. This solves the issue of merged reviews for similar products, but also removes duplicates posted by the same reviewer for completely different products. This is both, beneficial and harmful to our experiments, since legitimate duplicates are removed but also exact copies which are likely to be spam entries and hence relevant to our analysis. Unfortunately, it was impossible to repeat the de-duplication process given the raw data, since there is no way to tell from the data

[34] See Collobert et al. (2011); LeCun et al. (2015)
[35] See McAuley et al. (2015)

whether a product is a different version of the other or totally different, as all have a unique identifier and also differ in product title, price, sales rank etc. We chose to analyse the category with the second largest number of reviews, as we will most likely find a large number of spam reviews in it. We refrain from using the by far largest category in the Amazon data, which is books, as it contains more than 22 million reviews and would require too much memory and processing time to analyse. Instead, we turn to the category Electronics (7.8m reviews). They contain enough reviews so that we will be able to find a number of spam reviews sufficient to train all our models efficiently. Additionally, we assume that they mostly contain durable and relatively pricier products, i.e. customers have a higher incentive to incorporate reviews into their buying decision and the sellers' incentive for falsifying reviews also increases.

Obviously, online reviews are not labelled as spam or truthful reviews, which means some form of retroactive labelling is necessary. Besides creating untruthful reviews by paid freelancers[36], which is costly and may results in reviews deviating from review spam commonly found on websites like Amazon[37], another approach taken by several authors is to detect near-duplicate reviews and label them as spam. The underlying assumption is that reviews posted multiple times are most likely untruthful reviews, since genuine customers very rarely purchase similar products that would justify the posting of such near-duplicate reviews. It is refrained from looking for duplicates only, because spammers are likely to reuse a large part of counterfeit review texts and only change mentioned product names or attributes, for example. Those items would be missed during a filtering based on complete similarity. This turns the problem of labelling into a near-duplicate detection task. Documents cannot be compared string by string, since this method would only find real duplicates. A common approach to this problem is to turn documents into separate character strings of fixed length, so-called shingling. Each document is then represented by a set of strings occurring within it.[38] Several measures providing a notion of similarity between sets exist. Commonly used is the Jaccard similarity, which simply compares the size of the intersection of two sets A and B to their union.

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|}$$

[36] Ott et al. (2011), p.311f
[37] Mukherjee et al. (2013a), p.4ff
[38] Manber (1994), p.4f

The Jaccard distance can be inferred from this measure by simply subtracting its value from one. As stated in the previous section, the Amazon review dataset is of considerable size, which makes a procedure based on pairwise comparison infeasible. Theoretically, it would result in $n * (n - 1)$ comparisons to calculate the distance of each document (represented by its set of shingles) to every other document. Furthermore, if n is very large, the data may not fit into main memory. Thus, less computing and memory intense methods have to be considered. One way is to reduce n-gram sets representing documents into more compact signatures, which preserve, at least approximately, the similarity between documents and enable comparison based on a specific similarity measure. *Minhashing* is a procedure that provides all these features. As the name indicates, this method makes use of hash functions to remain computationally feasible. Logically, all documents and their corresponding n-gram sets can be represented as a sparse matrix, where each column represents one document and the rows represent the union of all existing n-grams. If an n-gram is included in a document, this is indicated by a one. For each document, n hash functions are applied on the respective n-gram set, each returning the index of one n-gram that is treated as the minimum of the set for this particular hash function (hence the name *minhashing*). This results in a matrix of dimensions D x n, where D is, again, the number of documents in the corpus and n is the number of hash functions. Each document is therefore represented by a column in this matrix. Since this matrix consists of mere integers instead of a (potentially large) number of words, it is much more memory efficient.[39]

Documents with a large share of n-grams in common will also have a lot of similar entries in the signature matrix. A hash function *i* will produce the same value for two documents if an n-gram is selected that is contained in both sets. It will produce different values in case the n-gram for each document is not an element of the other. Let's assume A is the set of n-grams contained in both documents, and B the set of n-grams that occur only in one of the documents. For the hash function i and two documents D_1 and D_2, $h(D_1) = h(D_2)$ if an n-gram from A will be selected. On the other hand, $h(D_1) \neq h(D_2)$ if an element from B is selected. Hence, the ratio of equal to unequal entries in the signature matrix will be $\frac{A}{A+B}$. As for the Jaccard similarity, A is exactly the size of the intersection of the two documents, whereas A and B is the union of them. According to the law of large numbers, the similarity of the signatures will approach that of the documents with an increasing number of hash functions.[40]

[39] Leskovec et al. (2014), p. 80ff
[40] Leskovec et al. (2014), p. 82f

Now, having reduced n-gram sets to smaller similarity preserving signatures might have reduced the necessary memory requirements by several orders of magnitude, but the problem of pairwise comparison remains. In the case of near-duplicate detection, it is unnecessary to know all distances for all pairs. Rather, it can be focussed on the most similar ones. Locality Sensitive Hashing (LSH) is a procedure that aims to extract candidate pairs with a similarity above a given threshold, which can subsequently be further analysed by pairwise comparison. Firstly, the aforementioned signatures are "cut" into a predetermined number of bands *b*, each of the same length *r* (with signature length n, $r = \frac{n}{b}$). The signature matrix is virtually split into *b* matrices with dimensions D x r. Thereafter, each column of each band is hashed to a number of buckets. In case two columns are identical in one band, they are hashed to the same bucket and considered a candidate pair.[41] Again, this is a heuristic procedure and produces false negatives (similar pairs that are not identified as candidates), as well as false positives (pairs with low similarity identified as candidates). The threshold similarity when a pair is likely to be identified depends on the parameters b and r (and therefore also n): The more rows a band contains, the unlikelier it is that two signatures are identical within one band and vice versa. Additionally, the more bands are created the higher the chance that two signatures will be identical in any of these. The formula describing the probability that two reviews with true similarity *s* will be identified as candidate pairs is then $1 - (1 - s^r)^b$, where s is the similarity of the reviews, and *r* and *b* the number of rows and bands, respectively.[42]

Using *minhasing* and LSH, the quadratic cost of comparing the distance between documents on a pairwise basis is reduced to linear complexity (for each new document, the hash functions have to be used only once). The price paid is to allow for false positives and negatives, which might be inacceptable when it comes to detection of plagiarism or similar cases, but our goal is to extract untruthful reviews in order to create a labelled data set as the basis for supervised machine learning. Hence, not identifying every near-duplicate review in the data is not a problem.

After pairwise comparison of candidate pairs identified by LSH, the distribution of Jaccard-similarities looks bimodal, with the majority of pairs having a similarity of less than 0.3, only about 5 – 10% being very similar (> 0.8) and very few reviews between those values. Still, it remains the question above what similarity threshold reviews can be considered as spam. Different authors come to different conclusions: Jindal and Liu (2008) consider pairs with a

[41] Leskovec et al. (2014), p. 87ff
[42] Andoni & Indyk (2006), p. 119f

Jaccard similarity of 0.9 or more as near-duplicates[43], Lau et al. (2011) choose a cosine score of 0.83.[44] Both authors emphasize that their chosen values may not be optimal and see it as the potential subject of future research. Setting the threshold is basically a trade-off between data size and risk of including genuine reviews. We want to gather as many untruthful reviews as possible, which would propose a rather low value, but also make sure the boundary is high enough to exclude genuine reviews that just happen to be similar due to the usage of common phrases, for example. Table 3.1 contains some cherry-picked example review texts and the corresponding maximum similarity value.

In contrast to the fixed cut-offs used by other authors, we propose special handling of short reviews (in terms of number of words) in order to increase the number of extracted spam reviews while simultaneously reducing the number of accidentally included genuine items. As expected beforehand, manual inspection confirms that for longer reviews, sentences, phrases or just single words have been replaced in near-duplicate reviews. Even texts with lower scores of around 0.6 can be considered fake reviews after manual inspection due to large overlapping sections (see Table 3.1). Reviews of shorter length are another issue, however. For lower similarity levels, it is often ambiguous whether they can be considered untruthfully copied reviews or just similar due to common phrases. In the second row of Table 3.1, for example, there are five unique shingles in the two reviews' union (*"excellent easy"*, *"easy to"*, *"to use"* etc.), three of them occur in both reviews. Hence a similarity value of 3/5 or 0.6. But it is not at all clear whether this is a case of slightly augmented duplicates or just randomness due to the usage of common words. Therefore, reviews below a fixed word length of seven and similarity values of lower than 0.8 are not labelled as spam. This procedure reduces the total number of reviews available for model training, but ensures not too many genuine reviews are included either. The problems caused by short reviews do not end here. Exact duplicates can also not always be undoubtedly marked as spam, as review texts such as *"I love it"* or *"Exactly what I wanted"* illustrate. Their share of total near-duplicates detected is not trivial; hence they would have significant influence on the model. In order to solve this issue, the summary text is included in the analysis: When both summary and text are identical, the reviews are considered untruthful duplicates. If the combination of review text and summary is unique, the corresponding review is considered to be genuine and not labelled as spam. The rationale behind this is that the probability of a review being fake decreases when the summary is unique, since an untruthful poster would probably not copy

[43] Jindal & Liu (2008) p.222
[44] Lau et al. (2011) p.17

the review text but then spend time in creating a new summary, especially not if the review text is only several words long.

The decisions made during the selection process are still based on non-exhaustive manual inspection and subjective assumptions. Nonetheless, it can be assumed that this improves data quality by an additional margin compared to a single cut-off similarity value.

We detect 54.377 supposedly untruthful reviews with this procedure, or about 0.7% of the total. Even though our similarity threshold is lower, this is roughly half the number of what other authors discovered in data from the Amazon website.[45] This deviation could have many causes: The heuristic *minhashing* & LSH procedure employed, as well as the fact that all duplicates posted by the same reviewer were already deleted. Besides, the samples downloaded by the other authors are small in size compared to our exhaustive approach, and thus may be biased, too.

Review Text		**Similarity Score**
Awesome, just what I needed. Thank you.	Just what I needed.	0.5
Excellent, easy to use	Excellent, easy to use and clean.	0.6
Bought this for my daughter in law for Christmas. She loves it and says it works great	I got this for my mom for Christmas, she loves it and says it works great.	0.55
[…]Great quality, especially for the price. No way would I spend $40-50 on a towel. This is very plush and absorbent and has held up well, even when washed in hot water.	[…] Great quality, especially for the price. This is very soft, plush and absorbent and has held up well, even when washed in hot water.	0.65
…I shopped around considerably before investing in Colorwave, but now have eight place settings in green and eight in red. They are wearing well after a year.	…I shopped around considerably before investing in Colorwave, but now I have eight place settings in green and eight in red. This particular bowl has a very pleasing shape.	0.7
Bought as a gift for daughter. She loves them.	Bought as a gift for my daughter, she loves them.	0.7
Excellent, easy to use and clean.	Easy to use and clean	0.8
…My order exceeded my expectations. My order will go great with my collection. I think this is a good picture of Rick Springfield in concert.	…My order exceeded my expectations. My order will go great with my collection. I think this is a good picture of Cheryl Ladd.	0.8
I'm really pleased with these bags. […] I buy garlic in bulk, and so far they're holding up well.	I'm really pleased with these bags. […] I buy onions in bulk, and so far they're holding up well.	0.9
I recommend these flannel pillow cases […] This set looks exactly as it appears in the images, and holds up pretty well	I recommend these flannel sheets […] This set looks exactly as it appears in the images	0.9

Table 3.1 - Selected review excerpts and their corresponding Jaccard similarity values

[45] Jindal & Liu (2008), p.223

Besides the extensive Amazon review data the structure of which was already explained in previous chapters, we searched the web for other sources of online reviews and found two additional datasets. The first one contains roughly 200.000 reviews of hotels in selected cities posted on the Tripadvisor (TA) webpage, the second is made publicly available by Yelp as part of the annual Yelp Dataset Challenge. It contains 2.2m reviews posted by more than half a million about 77.000 businesses and even comes with pictures and information of social connections between users.[46] We run the same procedure of labelling on them as we did for the Amazon data. Using the 1% near-duplicates found there as baseline value, we would expect to find about 2.000 spam reviews in the TA data, but only discover a mere 106, which is just too few to build robust and meaningful models based on them. For the Yelp data however, the procedure yields more than 12.000 spam reviews for a similarity threshold of 0.8, i.e. more than half a percentage point, which is relatively close to the Amazon baseline.

4 Deep Learning

Besides classical machine learning approaches discussed in section 2, ever more sophisticated models became highly popular during the last few years. Due to their recent success in different contests and implementation in real-world applications, they have also received much public attention. One case was the victory of a model in the ancient Asian game of Go against Lee Seedol, who is considered one of the best human player in the world.[47] Similar techniques are successfully applied in technologies like self-driving cars, face recognition and language translation, to name just a few. These architectures are commonly referred to as deep learning models. In contrast to classical machine learning, their architecture usually involves multiple so-called modules, stacked on top of each other. The most prevalent models among deep learners are deep neural networks (DNN). They primarily consist of input, output and several hidden layers, which are connected by weights that are numerically optimized during the training process. The main difference is that deep architectures are usually composed of many layers which are not necessarily fully-connected to each other.[48] Although the main concepts used to develop DNNs were known for decades, their breakthrough only came in the mid-2000s, advanced by ever increasing processing power and size of the data to learn from.[49] These models have shown to obtain state-of-the-art performance in several benchmark tasks and perform particularly well on computer vision problems, such as image

[46] Yelp (2016)
[47] Borowiec (2016)
[48] LeCun et al. (2014), p.436ff
[49] Schmidhuber (2015), p. 85ff

classification and object recognition, and in natural language processing tasks. In recent competitions, the best models were even able to outperform humans, e.g. in recognizing street signs or handwritten digits.[50] Another key advantage of deep learning architectures is the automatic detection of intricate structures in the training data. Before the advent of deep learning models, machine learning tasks relied heavily on manually crafted feature representations, which describe the relevant aspects well, but are also invariant to irrelevant changes in the training data (e.g. the same object of interest in two images presented in different pose, light etc.). This usually requires a considerable amount of time and domain knowledge. Deep learning models come with a built-in general-purpose learning procedure that can make expensive manual feature engineering obsolete.[51]

In this chapter, one of the most popular models from the deep learning family, namely convolutional deep neural networks, is presented and its suitability for text classification tasks is discussed.

4.1 Convolutional Neural Networks

One specific type of deep neural network architectures is the so-called convolutional neural network (CNN). CNN were invented for and have been proven to be especially successful in computer vision applications.[52] They have produced several state-of-the-art benchmarks in this field, such as object classification or digit recognition, to name just a few.[53] The name stems from a layer in its architecture, where a filter with fixed window size "convolves" over the input feature space in order to extract local features. This approach was introduced as early as 1980,[54] but had its breakthrough in the late 1990s, when it was successfully trained on raw pixel images (i.e. without costly hand-crafted features) and produced long lasting state-of-the-art results on the MNIST data, a popular machine learning benchmark data set.[55] CNN gained further popularity in 2012, when a CNN implementation won the renowned annual ImageNet Large Scale Visual Recognition Challenge by a large margin and also delivered outstanding results in subsequent challenges.[56]

[50] Schmidhuber (2015), p. 97
[51] LeCun et al. (2015), p. 436
[52] Kim (2014), p.1
[53] Bengio (2009), p.24
[54] Schmidhuber (2015), p.90
[55] LeCun (1998), p.1
[56] Krizhevsky et al. (2012), p.1f

4.1.1 Main Components

As mentioned above, the crucial ingredient of a CNN is the convolutional layer. The input layer is represented by a feature matrix. In case of an image, this could be the greyscale or RGB values for each pixel in the image. For a sentence or sequence of words, it would be some vector representation of the words (e.g. continuous word vectors) concatenated to a matrix. Several filters of fixed dimensions convolve over the input layer, meaning they change their position by a fixed distance (*stride*) until every part of the input layer was seen. For each *window* or *receptive field*, a dot product in the form of $W^t \cdot x + b$ is calculated, and one unit or neuron of the subsequent layer is produced.[57] For example, if the input is an image of size 28 by 28 pixels, the filter has dimensions 3 by 3 and the stride is exactly one pixel, the resulting feature map would be of dimensions 26 x 26 (the filter can be shifted 28 - 3 times plus the initial position). The input can also be *padded* with zeros at the edges in order to a get subsequent layer of equal size, which is useful when a CNN has many layers in total to prevent decreasing layer size. Note that in the aforementioned formula, the input vector x does not consist of all observations, as in fully-connected neural networks, but rather the local subset seen in the corresponding *window*. Additionally, the weight vector W^t is shared across all neurons produced by this specific filter.[58] This principle of convolution demonstrates two main advantages of CNN, the locality of receptive fields and the shared weights. The former leads to some degree of invariance or equivalence to certain input changes.[59] This has proved highly advantageous in computer vision applications, where some features naturally matter a lot more than others, e.g. the shape of an object is usually more important than its position in the picture, at least in object recognition. The shared weights are a further advantage when it comes to the number of parameters of the model. Assuming quadratic greyscale images of k *x* k, a standard feed-forward neural network based on matrix multiplication and with *n* neurons in the subsequent layer requires $k^2 * n$ parameters, since every input is connected to every output neuron. In convolutional layers, the parameter set is multiplied with different input nodes, i.e. used multiple times.[60] Ultimately, the layer outputs are transformed by some activation function, such as a rectified linear or hyberbolic tangent.

In addition, convolutional layers are usually followed by pooling layers in most architectures. Pooling further "shrinks" the output dimensions, as several units are summarised by a pre-specified function into a single unit. This operation further helps to increase invariance to

[57] LeCun (1998), p.6
[58] Goodfellow et al. (n.d.), p.331ff
[59] LeCun (1998), p.6
[60] Goodfellow et al. (n.d.), p.335

small input translations. It also decreases the input size to subsequent layers, thus reducing the overall number of parameters in the model. In some cases, pooling can also be used to shrink input of variable dimensions to the desired size where needed, e.g. in the fully-connected output layer.[61] Text input of variable length can be reduced to one neuron when using max-pooling for each sentence or document, for example. Figure 4.1 illustrates the concepts of convolutional and pooling layers on an image classification task. As in standard neural networks, the output layer is usually fully-connected to the previous one and fro example produces class probabilities via an arbitrary transformation function.

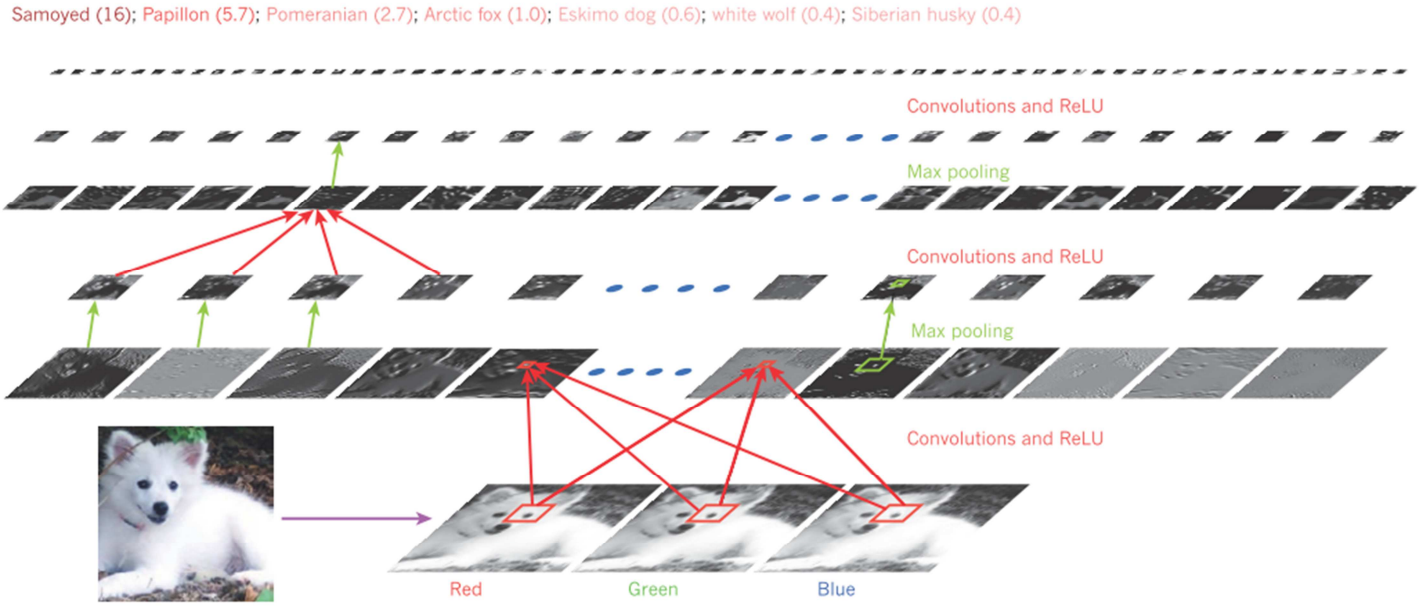

Figure 4.1 - The inner workings of a CNN. Different filters slide over the input and produce feature maps which are further transformed by a pooling operation. The steps are repeated for subsequent layers. Reprinted from LeCun et al. (2014), p.438

Another point is that in contrast to fully-connected deep neural networks, which could only be successfully trained using unsupervised pre-training of layers, CNN delivered outstanding results even when initialized randomly.[62]

Note that this architecture requires a potentially massive amount of hyper-parameters to be optimized. There is the number of filters, their size, the size of the stride, the number of layers and their order and so on. After the early implementation of LeCun et al. (1998), architectures primarily got more sophisticated in terms of number of deployed filters and layers.[63] However, quite recently it was shown that the number of filters combined with the number of

[61] Goodfellow et al. (n.d.), p.340f
[62] Bengio (2009), p.24
[63] Krizhevsky et al. (2012), p.4f

total convolutional and pooling layers mattered more than other decisions on hyper-parameters. Simonyan and Zisserman (2014) won another important object classification challenge in 2014 with a CNN architecture that simply relied on the same settings in every layer, but used 16 layers in total and doubling the number of filters in almost every layer.[64]

4.1.2 CNN in Sentence Classification

More recently, researchers started using CNN on NLP tasks such as topic and sentiment classification, embedding extraction from raw text and on token level applications such as POS-tagging.[65] Collobert et al. (2011) made the first major contribution to this area. Their goal was to achieve reasonable performance for common NLP tasks such as POS-tagging, NER, chunking, etc., but without the manual feature engineering which was required to obtain a good performance at that time. They built a convolutional architecture with a look-up table in the first layer, where each word is substituted by its corresponding 50-dimensional vector representation, followed by max-pooling and a fully-connected layer, which predicts the corresponding label for each word. When the input word embeddings were initialized randomly, the model did not yield great performance. They went on to train word embeddings using a simple neural network structure on corpora of several hundred millions of words. With these pre-trained word vectors however, performances close to state-of-the-art models were obtained across all tasks.[66] This laid the foundation for later algorithms aimed at training unsupervised word vectors, such as *word2vec*. It will be further elaborated on these models in the following subsection.

Kalchbrenner et al. (2014) use CNN for sentiment analysis and question classification. The novelty of their approach is to use so-called dynamic k-max-pooling, where the k highest values are extracted from feature maps instead of a single global maximum. Additionally, the k value is a function of sentence length and the number of layers in the network. The authors argue that this architecture helps to identify and draw together important features, such as words which are several positions apart and higher-order features which potentially span the whole sentence. Indeed, they were able to beat current benchmarks in sentiment prediction and reached a performance close to state-of-the-art in question type classification, where the best models still relied on extensive feature engineering.[67]

Johnson and Zhang (2015) use CNN for sentiment and topic classification. They are able to obtain state-of-the-art performance on both tasks in a purely supervised fashion, i.e. without

[64] Simonyan & Zisserman (2014), p.1ff
[65] See Johnson & Zhang (2015), Collobert et al. (2011)
[66] Collobert et al. (2011), p.2514f
[67] Kalchbrenner et al. (2014), p6ff

using pre-trained word embeddings. Instead, they use one-hot encodings to represent words in the input layer. Additionally, they elaborate on task depending hyper-parameter settings and note that for sentiment classification, smaller filter sizes with max-pooling over the whole document are more suitable, whereas for topic classification, longer filter sizes and multiple average-pooling operations yield better results. This makes sense, since short text fragments can have a high predictive power when it comes to the text's sentiment usually regardless of where they occur (e.g. "this is great"). To successfully identify the topic however, longer snippets (accounted for by greater filter sizes), their position in the text (multiple pooling operations) and also the complete content (average-pooling) matter a lot more.[68] Regarding the success of CNN in beating traditional approaches such as SVM on text classification tasks, the authors argue that the former are better in preserving word order and hence are able to extract more information from the input. A bag-of-words approach does not take into account word order. For example, the phrase "not bad, definitely recommended" typically conveys a very positive attitude, but represented as unigrams, a model cannot make use of this information and has to rely purely on word occurrences (e.g. the phrase "bad, definitely not recommended" is identical when represented as BOW). CNN however, keep word order within their filter windows and are therefore able to capture text segments with high predictive power.[69] Another advantage of CNN is that through the use of word embeddings they can take into account words which did not occur in the training data when making predictions on new data. Johnson and Zhang (2015) showed that even when networks are initialized without pre-trained embeddings, due to internally produced embeddings they are still able to capture the predictive power of a text snippet as long as some of their constituent words did occur.[70]

Finally, Kim (2014) used a quite simplistic CNN architecture to produce outstanding results when measured against various other models (NN-based as well as not NN-based) on multiple benchmark data sets for sentence classification.[71] The model we use to classify product reviews as spam or non-spam is based on this architecture and will be further elaborated on in the next section on experimental design.

[68] Johnson & Zhang (2015), p.6f
[69] Johnson & Zhang (2015), p.8f
[70] Johnson & Zhang (2015), p.9
[71] Kim (2014), p1.ff

4.2 Continuous Word Representations

The classic NLP approach to tasks such as Part-of-Speech tagging, Named-Entity-Recognition or sentence classification was heavily based on manual feature engineering, which required domain expertise and time, due to its reliance on trial and error. The more complex the problem, the more demanding the feature engineering process usually is.[72] In basically every NLP tasks, words cannot be fed into models as raw strings, instead, they need to be represented in some numerical form. These inputs play an important role in the model building process. The costly feature engineering step can be avoided if it is possible to create representations that contain most of the relevant information. The simplest way to encode words into numerical representations is via so-called one-hot-encoding. The words in the sample are transformed into a sparse matrix with dimensions of number of observations (documents, sentences etc.) in the data and the size of the total vocabulary (i.e. all words in the corpus). A simple model can then be trained on this data in order to solve a variety of tasks, such as document classification. A common approach is to use n-grams instead of single words for encoding. An n-gram consists of n words appearing next to each other in the data. Typical n-gram sizes are two to five words. These forms of discrete encoding offer benefits, such as simplicity and the possibility to train relatively simple models on them, but the downside of this type of encoding is that it is impossible to capture semantic or syntactic relationships between different words. Even though "great" is very similar in its meaning to "awesome", it does not have more or less in common with it than with any other unrelated word in the discrete vector space, since the dot product of the two vectors is zero. Additionally, the whole representation matrix scales with the number of words in the vocabulary, which can easily reach tens of thousands.[73]

As an alternative methodology, continuous representations or words embeddings have emerged. They rely upon the idea that words can be represented by the context in which they appear; an old but powerful approach. There exist basically two different approaches to map discrete word representations into lower-dimensional continuous ones. One branch, known as matrix factorization methods, uses a sparse matrix as input and derives continuous word vectors through singular value decomposition (SVD). As input matrix, a document-term matrix can be used, which captures word occurrences across documents in a corpus, as well as

[72] Collobert et al. (2011), p. 2498
[73] Mikolov et al. (2013a), p.7

a term-term matrix, which displays counts of words frequently occurring together in a window of fixed word length.[74]

Employing SVD on the input matrix subsequently results in another matrix of the same size. It is then up to the practitioner to choose an appropriate submatrix, depending on the desired variance captured, with dimensions |V| * k, where k will be the dimensions of each word vector.[75] In recent research, vector length is usually between 50 and several hundred dimensions.[76] This method has its drawbacks: Firstly, it is not very flexible, i.e. when new words are added to the corpus, the dimensions change. Furthermore, the matrix is very sparse since most words just never occur in the same context with most other words and is also high-dimensional. The usage of SVD leads to exponential training costs, and it also has to be accounted for the extreme imbalance of words.[77]

Another way to derive continuous word representations from a corpus are so-called *window-based methods*. These models try to predict a word based on its direct neighbouring context instead of deriving a representation based on global co-occurrence counts. Neighbouring context in this case means a pre-defined window of words, usually containing either a fixed number of words to the left or a symmetric window to the left and right of the center word. As in common neural network implementations, the prediction error is then backpropagated through the network and appropriate representation vectors learned for each word in the vocabulary. Early implementations of these models relied on a complete neural network architecture, where the word embeddings are a subset of the parameters of the network determined through the training process.[78] However, a recent publication by several researchers employed by Google introduces models that skip the costly computation of neural networks' non-linear hidden layer(s) and make predictions directly based on the word vectors.[79] These models are now commonly referred to as *word2vec*. The projection matrix is of size $|V| * D$, where D is the length of the embedding vector, and thus represents all word vectors concatenated to one matrix. Multiplying the discrete one-hot input vectors with the projection matrix simply results in a vector with D dimensions. Basically, this means one-hot vectors are still the raw input here, but they merely determine which row of the projection matrix is selected (i.e. they merely "look-up" the corresponding word vector). This vector is then multiplied with the $D * |V|$ dimensional weight matrix, which results in a vector of

[74] Pennington et al. (2014), p.2
[75] Furnas et al. (1988), p.469f
[76] Mikolov et al.(2013a), p.7
[77] Pennington et al. (2014), p.2
[78] Bengio (2003), p.1141f
[79] Mikolov et al. (2013a), p.4

scores for each word in the vocabulary, consequently of $|V|$ dimensions. Finally, the scores are turned into probabilities via a softmax function. As in standard neural-network architectures, the prediction error is then backpropagated and the projection layer updated accordingly. [80] Mikolov et al. (2013a) introduced two variations of this architecture: In the so-called continuous bag-of-words model (CBOW), a symmetric window of fixed length around the center word is used to predict this center word. In order to end up with a single vector as projection layer, word vectors representing each word in the window are averaged.[81] Hence the name bag-of-words, indicating that word order is not taken into account. The second model uses the opposite approach, as it predicts the surrounding words based on the single center word. The projection layer is therefore a vector by definition, and it is accounted for the multiple output layers in the objective function that is optimized. Note that the normalization in the denominator of the softmax-function is a very costly calculation, since it takes the whole vocabulary into account. Alternative approaches to circumvent this and hence make the whole computation more efficient have been proposed, such as hierarchical softmax and negative sampling. These methods resulted in rapid improvements in computation time of general word vectors based on very large corpuses.[82]

One advantage of word embeddings is that they can be learned from unlabelled corpora. This is hugely beneficial, since especially for NLP problems, most of the existing data is unlabelled and manual labelling costly. Additionally, word vectors can be stored independently and used for initializing the projection matrix in arbitrary subsequent NLP tasks, which makes a repetition of the training process unnecessary. Many researchers have not only published their empirical results but also the obtained word embeddings for public use.[83] Early versions of neural network based continuous word representations were able to outperform classic n-gram based approaches.[84] It has later been shown that models based on pre-trained word vectors additionally improve performance for most NLP tasks, compared to randomly initialized word vectors which are then trained via backpropagation.[85] As several studies have shown, their quality increases with training data size as well as dimensionality, but in turn leads to increased computation time.[86]

[80] Rong (2016), p.1ff
[81] Rong (2014), p.6f
[82] Mikolov et al. (2013c), p.8
[83] See Mikolov et al. (2013a); Pennington et al. (2014)
[84] Bengio (2003), p.1148
[85] Kim (2014), p.3f
[86] Mikolov et al. (2013a), p.7

The list of advantages of continuous word representations does not end here. Before 2013, the quality of word embeddings was measured primarily by examining word vectors closest to each other in the feature space and then manually comparing their meaning. For example, word vectors close to *France* should be syntactically and semantically related, such as *Germany*, *Belgium*, *Europe* etc., instead of *persuade*, *blackstock* or *collation* (real example from Collobert et al. [2011]).[87] The aforementioned Google researchers made another remarkable finding in 2013, as they were able to show that word vectors are capable of encoding not only syntactically and semantically similarity well, but also relationships in multiple other dimensions, in the form of vector offsets. To proof their point, the authors formulate analogy questions of the structure "a is to b, as c is to x", where x is unknown. In order to examine syntactic regularities inherent in the word vectors, the parameters can be replaced by base/comparative/superlative forms of adjectives or singular/plural and possessive non-possessive forms of nouns, to name just a few possibilities. For semantic regularities, analogy questions can be expressed like "clothing is to shirt as dish is to bowl". The authors continue to show that these questions can be effectively answered using a simple vector offset method based on cosine similarity. For example, subtracting the word vector of *man* from *king* and adding *woman* results in a vector that is closest to the learned representation of *queen*. This implies that commonalities between pairs of words in particular relations are represented as constant vector offsets. Employing this method on standardized test sets using word vectors trained via matrix factorization (LSA) yields significantly worse results than using NN based vectors.[88]

5 Experiments

As outlined in section 2, most studies are based on different data, classifiers or loss functions among other things. Therefore, it is hard to make valid statements about the effectiveness of individual methods for identifying review spam. This paper's aim is to examine multiple approaches with varying parameters in order to provide a broader picture on this issue. In our experimental setup, two main approaches are employed. Firstly, we use a simple classifier to assess the effect of varying features on the quality of spam detection, as captured by several performance metrics. We additionally examine the effect of (severe) class imbalance on classifier performance, as review spam only accounts for a relatively small minority of total submitted reviews. Secondly, we use a novel deep learning approach to tackle this problem,

[87] Collobert et al. (2011), p.2510
[88] Mikolov et al. (2013b), p.746ff

training a convolutional neural network which successfully produced benchmark results for sentiment classification tasks.[89] Instead of manually-engineered features, we rely on word vectors kindly provided by Google as inputs to this model, and additionally create our own embeddings trained on 7.8 million reviews from the Amazon Electronics section.

5.1 Linear Classification

In the following subsections, we describe the choice of features and the architecture and tools we use to train our models. Subsequently, we present our results and finally discuss their implications.

5.1.1 Feature Engineering

As stated earlier, we use manually developed features alongside a bag-of-words approach and POS tagging, as well as different combinations of them to examine their effect on classifier performance.

Regarding manually created features, we try to emulate the feature set introduced by Jindal and Liu (2008) and also used by Lau et al. (2011) to achieve comparability of results. The only feature we are not able to replicate is the percentage of positive and negative words in a review text, since the authors did not disclose the list of words they used. Most predictors as listed in Table 8.1 are self-explaining. On review level, dummies are included indicating whether the review was the first one for the corresponding product and whether it is still the only one. Another dummy variable shows whether the review was posted as the second one with a rating opposite to the first one. Additionally, other review characteristics such as word length of summary and review text, as well as helpfulness ratings of other customers and the overall rating are considered. Regarding reviewer characteristics, the average rating and standard deviation are considered, alongside the percentages of first and standalone reviews the reviewer posted. Several dummies indicating whether the reviewer only posted reviews of similar rating tendency are also included. Finally, on product level, we include price, the sales rank for the product and its average rating and standard deviation.

BOW is automatically implemented in our machine-learning tool, but a few decisions on pre-processing have to be made. The level of pre-processing can usually range from almost no intervention at all to the removal of punctuation, stop words and stemming of words. In BOW, the order of words is irrelevant, hence the argument for removing stop words and punctuation, which occur in almost every text and carry little information. Dimensionality reduction can also play a role here, as many machine learning algorithms cannot handle a lot

[89] Kim (2014), p.4

of features, and stemming and stop word removal might significantly reduce the size if the feature space. Because the Vowpal Wabbit (VW) tool we use can handle billions of features, we refrain from intense manipulations such as stop word removal and stemming. Instead, we create a "clean" version of text reviews with punctuation and capitalization removed, and another version where almost no changes are made except removal of consecutive empty spaces and symbols VW is unable to handle. For POS tagging the review text, we use open-source software called openNLP. The software that creates LIWC features from raw text is unfortunately proprietary. We contacted the support team on whether creating features for data as large as ours is computationally feasible to make sure buying a license makes sense, but received no response. We therefore forego this step. Presumably, the generic features would not add much predictive value on top of the other approaches anyway. In Table 5.1, a brief overview of the different datasets is given.

5.1.2 Model Training

We measure model performance using three distinct metrics that are also commonly employed in the review spam literature, namely the area under the ROC curve (AUC), accuracy and F-score. All three come with different properties which have to be considered in order to interpret results correctly. Accuracy is one of the most basic methods to assess a classifier's performance. It is simply the ratio of correctly classified observations to all observations in the (evaluation) data. The concept makes no distinction about the type of error made. In some scenarios, for example medical trials, false negatives (e.g. infected patients faultily classified as healthy ones) can incur much larger costs than false positives or vice versa.[90] To a lesser extent, this is also true for review spam detection. To label and filter a truthful review as spam might be not as harmful to the platform as missing a deceptive one. Secondly, in the presence of severe class imbalance, even poor models could obtain a high accuracy value by simply classifying all observations as belonging to the majority class, which is definitely an issue when it comes to review spam. In the amazon data, discovered minority class instances (i.e. spam) amounts to less than one percent of the total. The effects of class imbalance will be further examined later in this chapter. To address the weaknesses of the accuracy measure, we also deploy AUC and F-score metrics. The AUC criterion is based on the class probabilities assigned to each observation by the model. A threshold determines at what value a data point is assigned to one class or the other, and altering this threshold results in more or less correct and false predictions, as well as different false positive and false

[90] Kuhn & Johnson (2013) p.254f

negative rates. Hence, there is a sensitivity (true positive rate) and specificity (true negative rate) trade-off. To obtain the receiver operating characteristic (ROC), these two values are plotted against each other for a continuum of thresholds.[91] The ROC curve not only helps to determine an optimal threshold given a particular prediction problem, but the area under the curve can also be used to assess classifier performance. In case it exists a threshold where all observations are correctly classified, AUC would be one. The F-score is again based on the confusion matrix, as it is simply the harmonic mean of sensitivity and another metric called precision, which is the ratio of true positives divided by all samples predicted as belonging to the positive class.

When it comes to software, besides the standard R software and several packages suitable for respective steps undertaken within the scope of our experiments, we use two additional pieces of open-source software. To train our models we use an online learning system called Vowpal Wabbit (VW), which was initiated by Yahoo! and is now sponsored by Microsoft. The main advantages for us are basically its speed and ability to deal even with terafeature datasets.[92] Since our dataset is about two orders of magnitude larger than data in previous research, we end up with more than 10^8 features for a bag-of-words approach and 2-grams. Other systems simply lack the ability to train models in such a large feature space, let alone in a convenient enough time span to test differing hyper-parameter settings. Furthermore, the tool can train SVM as well as LR models, the two classifiers primarily used in review spam research.

5.1.3 Results & Discussion

There are several datasets which we create in order to examine how different data pre-processing and selection methods affect performance. Table 5.1 provides an overview. Dataset 1 (DS1) is created according to the procedure outlined in chapter 3.1, where reviews with a Jaccard similarity above 0.6 are marked as spam and very short reviews as well as commonly used phrases are filtered. As for the second dataset (DS2), we aim to closely resemble the data used by Jindal and Liu (2008), who use a threshold of 0.9 with no subsequent filtering to create their training data. To obtain negative or non-spam samples, we draw randomly from the subset of remaining reviews without any restrictions in order to avoid the fallacy of biased sampling. Lau et al. (2011) achieve an almost perfect AUC value of 0.998 by selecting samples below a certain similarity threshold and subsequently treating cosine similarity as predictor in their model. The third dataset is sampled from the Yelp data using a threshold of 0.8 Jaccard similarity. From manual inspection, it seemed like users were

[91] Kuhn & Johnson (2013) p.424f
[92] Langford (2016)

able to add additional text to their already published reviews, which then occur twice in the data. This means those can have high similarity values without being untruthful reviews, so we had to set the similarity threshold higher than for the Amazon reviews (at 0.8) in order to filter most of them.

Name	Source	Similarity Threshold	Observations	% minority class
DS1	Amazon	0.6	108.754	50
DS2	Amazon	0.9	57.976	50
DS3	Yelp	0.8	24.504	50
DS4	Amazon	0.6	67.971	20
DS5	Amazon	0.6	60.418	10
DS6	Amazon	0.6	57.238	5
DS7	Amazon	0.6	55.486	2
DS8	Amazon	0.6	54.926	1

Table 5.1 - Datasets used for classifier training and testing

In the first round of model training using VW, we use DS1 to test the effects of varying experimental factors and meta-parameters. We examine the performance of SVM as well as LR and alter training options such as the n-gram value or learning rate. Additionally and more importantly, we experiment with different sets of input features, ranging from a BOW approach to POS annotated words to hand-crafted features and combinations of these three. When using BOW, we also test a setup where the text is "cleaned", i.e. special characters are removed and all characters are lowercase. An overview of the hand-crafted features is provided in Table 8.1 in the appendix.

Regarding the BOW approach, we test multiple values and come to the conclusion that 2-grams yield the slightly best performance and so we keep this value constant during further analysis. Using raw text as input in combination with BOW produces the best results. Not eliminating punctuation and stop words improves AUC by about 1%. When it comes to model selection, LR consistently outperforms SVM by a small margin and is therefore used for all further experiments. Surprisingly, hand-crafted features as in Jindal et al. (2011) do much poorer than the simple BOW model. We also obtain a much worse AUC of just about 0.7 compared to the authors' 0.78. However, this difference shrinks significantly when DS2 is used, where spam reviews are sampled according to Jindal et al. (2011), which underlines the impact of sample selection on classifier performance. Still, the effect of using DS2 instead of DS1 on AUC is not more than 1% when BOW instead of manually created features is used. Other variations of feature sets do not influence performance by that much. Although POS tagged text should contain more information, using the BOW model on the former performs

about equally well as on plain uncleaned text. It has to be noted that small differences in performance measures could always be due to the architecture of VW, which, as an online learner, is sensitive to the order of training samples and will not always reproduce identical results for the same data. Combining textual and hand-crafted features did also not improve the best results but merely averaged results obtained by each of the feature sets alone. Table 5.2 provides the final results given a balanced dataset. Due to space limitations, we do not present performance values for all tested combinations of meta-parameters but the best ones.

Dataset	Features	AUC	Accuracy	F-score
DS1	BOW (cleaned Text)	0.8714	0.799	0.804
DS1	BOW	0.8819	0.811	0.816
DS1	BOW (SVM)	0.8684	0.786	0.800
DS1	BOW (POS-tagged)	0.8772	0.806	0.812
DS1	hand-crafted	0.7009	0.643	0.604
DS1	BOW, hand-crafted	0.7957	0.723	0.719
DS2	BOW (cleaned Text)	0.8869	0.821	0.828
DS2	BOW	0.8933	0.828	0.834
DS2	hand-crafted	0.7360	0.668	0.636
DS3	BOW	0.8736	0.788	0.793

Table 5.2 - Classifier performance on balanced data

There are two main observations we make when evaluating resulting models based on performance. **Firstly, regarding input features, bag-of-words outperforms all other combinations of features**, even the identical text with POS annotations. Hand-crafted features for the Amazon data perform a lot worse than raw text. This comes quite as a surprise since other studies found that information contained in meta-data about review and reviewer significantly improves predictive models. It is not trivial to come up with a plausible explanation for this phenomenon. Other studies use yet different hand-crafted features, but our obtained results are so far off the best performance that we decided not to spend more time on manual feature engineering. **Secondly, our models perform relatively well compared to other studies in this field**. We do not obtain the performance of Jindal et al. (2008), whose experimental setup closely resembles ours with regard to feature set and data. With an AUC value of 0.701 we do not come close to their obtained performance of 0.78, even though we manually replicate features based on their study. But this is due to the poor performance we get when using hand-crafted features, as our BOW based models beat the score easily. Besides, performance improves, although just slightly, when the high-similarity data set (DS2) is used, which underlines the importance of data selection. With an AUC of

0.8819, we come close to the value Ott et al. (2011) achieves, who also use BOW but with POS-tags. As for the yelp dataset, we obtain a performance reasonably close to Mukherjee et al. (2013a) using only BOW with 2-grams. Since we build our training data from duplicate reviews, in contrast to filtered reviews published by yelp, scores are hard to compare.

However, it has to be taken into account that results are hard to compare since data sources and sampling, as well as pre-processing and feature engineering still widely differ. We did not replicate all feature sets for all compiled studies and most of the other studies are based on a significantly smaller subset of data.

5.1.4 Class Imbalance

In review spam detection, spam reviews do not make up a large share of the total number of reviews posted. The percentage of fake reviews we found in the Amazon and Yelp data using the near-duplicates detection technique is less than 1%. Other researchers who used the same method report equally low values of about 1% - 2%.[93] Even Yelp's review filter, which is said to being optimized on a high true-positive rate or sensitivity, only filters about 16% of all submitted reviews. These numbers emphasize the necessity of appropriate measures to deal with such a strong imbalance in order to produce effective spam detection models.

Different approaches tackle the problem at different stages of the model building process. One way to mitigate the effect of imbalanced classes is to provide it with resampled data as inputs, thus taking the problem away from the model. The basic way is to either up- or downsample observations from the minority or majority class, respectively, in order to end up with equal numbers for each class in the training data. Downsampling simply means to randomly draw from the majority class until both classes are balanced. Stratified sampling can be used if necessary to make sure some predictor values are contained in the data. Conversely, during upsampling, samples from the minority class are drawn with replacement, which can obviously lead to observations being represented multiple times in the training data.[94] Some models also allow for data points being weighted differently, which is closely related to upsampling but without its induced randomness. Furthermore, class imbalance can be considered directly during the model building process by tuning meta-parameters in a way that particularly improves sensitivity.[95] When the trained model is able to provide class probabilities, altering the cut-off probability at which a sample is considered to belong to one or the other class might be a sensible solution. This can highly influence sensitivity or

[93] See Jindal et al. (2008), Lau et al. (2011)
[94] Japkowicz (2002), p.440ff
[95] Kuhn & Johnson (2013), p.440

specificity of the model, but it has to be noted that it is merely a trade-off between those two measures and does not result in more or less accurate predictions.[96] Hence, it depends on the use case whether and how strongly altering the probability cut-off will actually improve classification results.

In order to account for the natural class imbalance present in labelled review spam data, we use two of the approaches briefly outlined, namely sampling techniques and alternating the probability cut-off. We refrain from using custom loss functions, as it is not possible to implement them in the VW tool. The dataset this analysis is based on is basically DS1 with the minority class of spam reviews being down sampled to 20%, 10%, 5%, 2% and 1%, respectively. Table 5.3 shows the change in performance metrics when the data gets more and more imbalanced. In addition to AUC, accuracy and F1-score we also provide sensitivity, specificity and precision in order to better spot the impact of changing class proportions.

Accuracy actually increases with the share of the majority class, since it does not penalize for simply classifying all instances as belonging to this class. AUC in contrast decreases with it, i.e. it becomes harder for the models to assign the right class probabilities to the samples in the test data. Looking at the other measures, it becomes clear that sensitivity is the main issue here, which means less and less observations of the positive class (spam) are labelled as such (false negatives). Unsurprisingly, specificity increases, as most models tend to overemphasize the majority class during the training process and classify the test data accordingly. For example, on data with a 1:99 class imbalance, even a naïve model which labels all instances as belonging to the majority class would achieve a specificity of 0.99. The same is true for our model: Specificity is one or close to one, meaning (almost) all samples from the negative class or correctly classified. The precision measure, which is also increasing alongside the share of the majority class, shows that the models are correct most of the time (or always, in case of DS7 and DS8) when classifying reviews as spam. The problem is basically that the number of missed spam reviews (false negatives) increases the smaller the share of the spam class in the data, as indicated by the sensitivity measure. It depends on the context whether this is beneficial or not. For example, as mentioned in the second chapter, Yelp wants to achieve the exact opposite trying to maximise the number of untruthful reviews that are filtered, accepting a potentially higher false positive rate. But in cases where it is very harmful to delete legitimate reviews by honest customers, a model tuned to maximise specificity and precision is imaginable. One possible explanation for these results is that some spam reviews (the ones correctly classified) come with certain characteristics which make it easy to

[96] Kuhn & Johnson (2013), p.440f

distinguish between them and the rest, resulting in high precision scores. The other part of spam reviews, however, possibly more closely resembles the truthful reviews. Due to the low share of spam reviews in the imbalanced data, the models are potentially unable to learn more subtle differences in the features.

Dataset	AUC	Accuracy	F-score	Sensitivity	Specificity	Precision
DS4	0.8439	0.896	0.669	0.531	0.986	0.905
DS5	0.8329	0.941	0.589	0.421	0.999	0.985
DS6	0.8128	0.968	0.529	0.3611	0.999	0.99
DS7	0.7799	0.987	0.472	0.309	1	1
DS8	0.8014	0.993	0.462	0.3	1	1

Table 5.3 - Classifier performance on imbalanced data

For tuning of the probability threshold, we need to further split our test set in order to find an appropriate threshold using the first part and measuring its impact with the other part. This is to avoid a biased estimate of the optimal threshold when tuning and testing it on the same data points. Figure 5.1 shows the plotted ROC curve for a model trained on data with the natural distribution of classes (1% positive samples). One can see that the curve is very steep in the bottom left part, indicating that for many thresholds a high sensitivity is reached by simply classifying all samples as non-spam (and thus obtaining no false-negatives and perfect specificity). On the other hand, the curve falls steeply on the top right, which means specificity strongly decreases as more and more observations are wrongly classified as spam. When the target is to maximise sum of sensitivity and specificity, selecting the threshold at the outer-most point of the ROC curve is reasonable, and results in a value of 0.029 in our case. The optimal value depends on the use case and the desired ratio of false positives and false negatives.

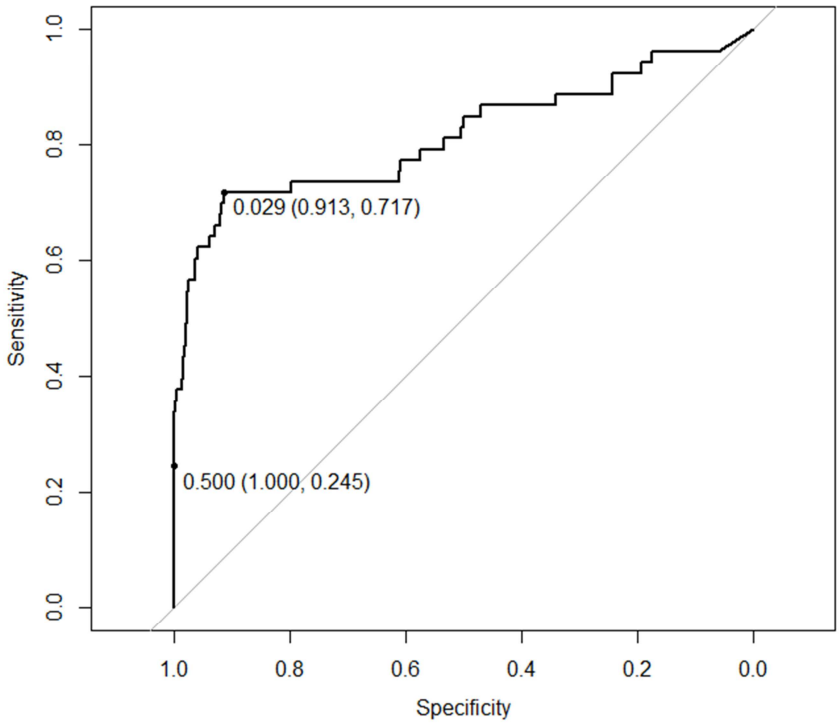

Figure 5.1 - ROC curve for test data set with natural class imbalance

5.2 Deep Learning

We try to benefit from the recent advent of deep learning models to improve review spam detection. The architecture we deploy is based on a CNN that was developed by Kim (2014) for classifying the sentiment (positive or negative) of online movie reviews, which has produced state-of-the-art benchmarks on several gold-standard datasets. Since our task is quite related (predicting binary labels for short snippets of user-generated text), we expect the model to perform well on this problem, too.

5.2.1 Model Training

Compared to other popular CNN architectures, especially those achieving top performance in image recognition tasks, the implementation by Kim (2014) is rather shallow.[97] The input layer is followed by a convolutional layer composed of several filters of different widths, which is followed by a max-pooling layer that is fully connected to the output layer. Dropout is used as regularization technique in the fully connected layer and the last layer produces a softmax output. Input features are created by looking up the corresponding word vector for

[97] Krizhevsky et al. (2012), p.4f

each word in a review and zero-padding to account for different review lengths.[98] This way, pre-trained word vectors as well as randomly initialized ones can be used as model input. The author resorts to word vectors trained and published by Google on a very large corpus of 100 billion words obtained from Google News. Words in the reviews which are not contained in the vocabulary of pre-trained word embeddings are randomly initialized based on the distribution of the Google News vectors. As for the movie reviews, roughly 88% of the vocabulary is contained in them. This serves as a useful baseline for our data. Furthermore, Kim's (2014) architecture allows for adjusting the word vectors used as input to the CNN by treating them as parameters of the model. In this *non-static* setup, the vectors are slightly altered or "fine-tuned" to the specific task at hand during the training process.[99]

In order to produce a model that delivers good performance, we need to adjust the hyper-parameters of the architecture according to the task at hand. In the following subsections, we further explain key components and discuss how hyper-parameter adjustments possibly affect performance.

The movie review data is relatively small, as it contains only about ten thousand sentences with the longest sentence containing 56 words. In contrast, our balanced dataset consists of more than 100.000 product reviews. Review length differs from one up to several thousand words. To include all samples, it would be necessary to zero-pad all shorter reviews up to the length of the longest one in order to produce valid input in matrix form. Due to the global max-pooling in the subsequent layer, this would not affect model performance. But it comes with increased computation costs, because every filter has to "slide" over each review mostly observing zeros towards the end. Hence, we cap the length of the longest review in the data to be able to run experiments in faster successions.

Kim's (2014) experiments show that deep learning models are able to perform well even on smaller data, at least when pre-trained inputs are used. This contrasts more recent research, which showed that deep-learning based models begin to outperform more traditional approaches only when the available data contains hundreds of thousands of samples or even millions.[100] Hence, we are interested in how classification performance is affected by data size and use subsets of the training set in order to test this. Table 5.5 presents the different data sets created and their characteristics. To examine whether the restrained review word length might affects the linear classifier's performance, we train and test a linear model on the

[98] Kim (2014) p.2f
[99] Kim (2014) p.5
[100] Zhang et al. (2015), p.5f

respective data that is used for deep learning model training using the setting which delivered the best performance in the previous chapter (BOW as input and logistic regression).

Kim (2014) further reports that performance usually improves slightly in the *non-static* setup. He goes on to compare selected words with the ones closest to them in cosine space and finds that fine-tuned word vectors slightly better incorporate semantics related to expressing sentiment. For example, Google's pre-trained word vectors consider "bad" most similar to "good", but for fine-tuned vectors synonyms like "terrible", "horrible" or "lousy" are closer to it. Additionally, "good" is closer to words expressing mildly positive attitudes like "nice", "decent" and "solid" in contrast to "great".

We take this notion of "fine-tuning" word embeddings a step further by creating our own, highly problem specific word vectors using the *Word2Vec* model. Taking advantage of the large body of unlabelled data that is available in the form of online reviews, we train a model based on the roughly 7.8 million reviews from Amazon's Electronics category. The goal is to obtain word vectors that reflect semantic similarities and other structures present in online reviews. Google's word vectors were trained based on a very large corpus of millions of articles from Google News. Although it is hard to beat this approach when it comes to data size, carefully written news articles do not closely resemble the style of user-generated product reviews, which possibly contain colloquial words and phrases, for example. Mikolov et al. (2013a) use a data set of question-answer pairs in order to evaluate how well the model is able to grasp different syntactic and semantic concepts. The model is supposed to find missing words for combinations such as capitals and countries, currencies or family relations, as well as syntactic combinations of adjectives and adverbs among others. Unsurprisingly, our word vectors perform worse when it comes to most semantic questions, since capitals or currencies are clearly underrepresented in online product reviews. Still, the model is able to correctly determine the majority of words for most syntactic questions despite the messy training data, which demonstrates the power of *word2vec* models. Following Kim (2014), we additionally explore common words' vectors and their closest neighbours in cosine space. Table 5.4 provides an overview. From this data sample, it seems like the resulting vectors indeed better incorporate structures which could be useful to solving the task at hand. As in Kim (2014), word vectors close to each other do not reflect syntactic similarities (e.g. the opposites good and bad), but semantic ones (good can often be used in the same context as decent, great etc.). Also, for nouns our model treats words as similar which one would expect to be used in similar context in opinion expression, which results in more suitable

representations compared to Google's word vectors (e.g. "device"). Additionally, the model is able to account for misspellings.

Word	Nearest five neighbours	
	Google News	Electronics Reviews
bad	good, terrible, horrible, Bad, lousy	terrible, horrible, good, poor, lousy
good	great, bad, terrific, decent, nice	decent, great, nice, fantastic, impressive
great	terrific, fantastic, tremendous, wonderful, good	fantastic, wonderful, terrific, GREAT, fabulous
poor	poorer, abysmal, poorest, lousy, woeful	Lousy, terrible, subpar, mediocre, inconsistent
reasonable	unreasonable, reasonably, resonable, justifiable, Reasonable	Competitive, good, decent, affordable, modest
quality	Highquality, Quality, ratios nonaccruing loans, SDVOSB firms, quailty	Quality, quailty, quaility, qaulity, quility
product	products, Product, onlySweet, product introductions, Rynaxypyr	item, case, unit, seller, company
device	devices, handheld device, Device, gizmo, gadget	Devise, unit, tablet, hub, machine

Table 5.4 - Closest five word vectors to specific example words vectors in cosine space by training corpus

As described in section 4, filters are located in the convolutional layer of CNNs. Their size, the number of filters used in each layer and the corresponding stride are meta-parameters to be set in advance by the modeller. Kim (2014) uses filters of length three, four and five words and 100 filters per filter size. Such small filters are often used in sentiment classification, but, as Johnson and Zhang (2015) point out, this might not be the optimal setting for other problems. The authors report that whereas smaller filters are successful in capturing the sentiment of a given text, they obtain better results for document topic classification when larger filters (length ten or above) are used.[101] A short phrase such as "this is great" conveys highly positive sentiment, which the model is perfectly able to capture using a filter of length three. The filter will produce a highly active neuron, and this signal is highly likely to be carried through subsequent network layers and ultimately help classify the sentence's sentiment. When it comes to topic classification for example, longer phrases usually contain more useful information than short snippets. This has to be reflected in the model architecture. We vary the number of filters used as well as their length in order to test which setting produces the best results in our spam classification task.

[101] Johnson & Zhang (2015), p.6f

Pooling usually happens after convolutional layers in CNNs. The data is simply condensed to a more compact format in order to further increase invariance to shifts in the input data and additionally, reduce model size. There are not so many options when it comes to choosing this meta-parameter. In fact, max-pooling and average-pooling are basically the two variants that are most commonly used. The size of the window on which pooling happens is of more concern in our case. Pooling over the whole review is a convenient way to overcome issues associated with different input lengths. For each review, global pooling returns a single value per filter. Since we already dealt with the variable review length problem by zero-padding, this is not a concern here. Still, we have to consider how many values we want to obtain after pooling. Is it plausible to assume that a single neuron can represent the necessary information to correctly classify the review? This is what happens in global pooling. In case pooling happens over an area smaller than the input length, multiple values are returned by the pooling layer. It is hard to come up with a plausible assumption beforehand. Therefore, we will simply test multiple settings empirically.

Name	Observations	Max. Review Length	LR Performance (AUC)
DL-DS1	62869	100	0.8764
DL-DS2	25000	100	0.7694
DL-DS3	12000	100	0.7142
DL-DS4	80139	200	0.8529

Table 5.5 - Datasets used for CNN model training and evaluation with AUC values obtained from logistic regression

As our data is several times the size of the movie review set, we run the experiments on an Amazon EC2 GPU instance, as GPU deployment can speed up the training process by one or two orders of magnitude compared to training on CPUs. Still, depending on data size and hyper-parameter setting, model training takes between roughly 30 minutes to several hours.

The data is split into training and testing set, while 20% of the training data is used as validation set during the model training phase to avoid overfitting. We use Theano, a mathematical Python framework aimed specifically at facilitating deep learning research, as the backend software for our experiments. It was introduced in 2010 and has since been improved and extended by its core developers and the community around it.[102] It offers support for parallelism on GPUs and in combination with the Python Keras library, it provides functionalities in order to quickly and conveniently experiment with deep learning models. Kim (2014) has published the Theano code accompanying his paper on github. It was later

[102] See Bergsta et al. (2010), Bastien et al. (2012)

modified by other users who achieved even better performance with slightly altered hyper-parameter settings.[103] We use this publicly available code and adapt it to our setup.

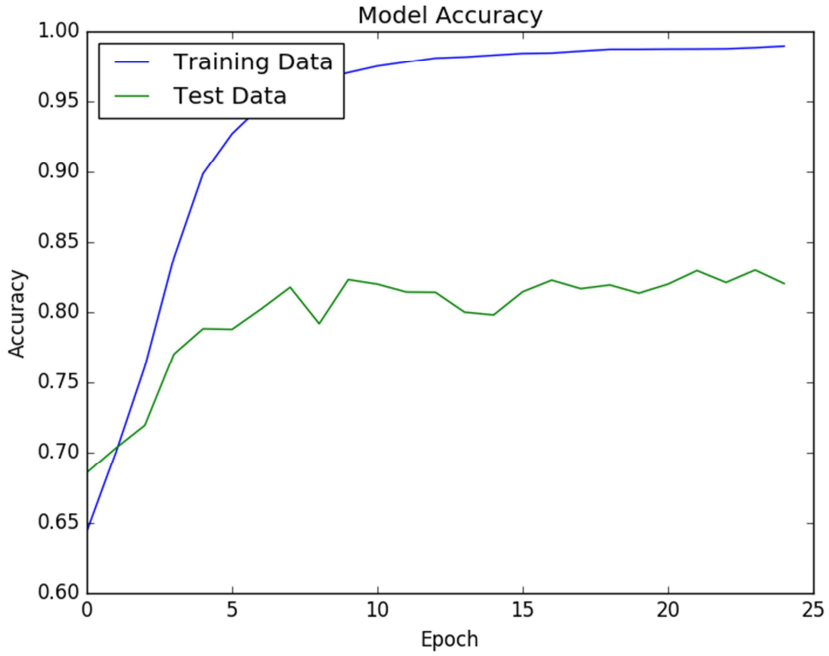

Figure 5.2 - Accuracy per epoch by training and test data for hyper-parameter setting "wide filters"

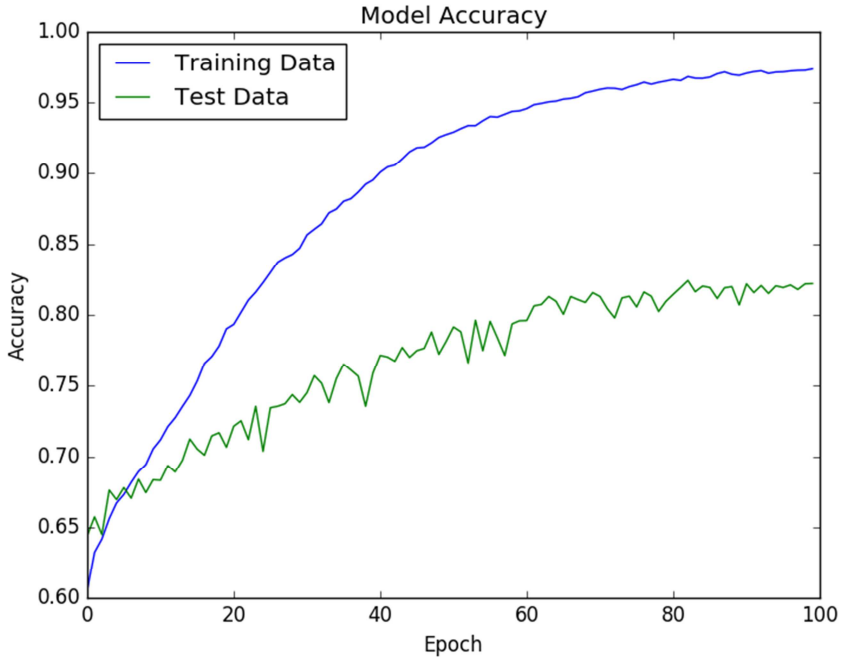

Figure 5.3 - Accuracy per epoch by training and test data for hyper-parameter setting "custom embeddings"

[103] Rakhlin (2016)

5.2.2 Results & Discussion

Table 5.6 summarizes the results obtained from our deep learning experiments for the corresponding hyper-parameter settings. The last column also states the difference in performance in terms of AUC compared to the linear model. For most parameter settings, the CNN performs worse than a logistic regression trained on the same data. When task-specific pre-trained word embeddings are used is the performance roughly equal. The exceptions are the data sets DL-DS4 and DL-DS3 which include sentences up to a length of 200 words (DL-DS4) and only 12.000 reviews in total, respectively. For both, the CNN performs about 1.7% better. The linear model's performance actually decreases when longer sentences are included, and it is also quite surprising that it performs worse than our deep learning model on the smallest data set.

We obtain the best performance when using short filters, global max-pooling, all the data we have available and above all, task specific pre-trained word vectors. As can be seen in Figure 5.2 and Figure 5.3, the model takes a lot more epochs to fully converge but finally delivers a slightly better performance (1%) than the models based on the Google News word embeddings. This strengthens the assumption that highly task-specific word embeddings indeed help classification. Word vectors trained on the complete set of all 142 million Amazon reviews possibly would have resulted in additional improvements, but we have to forgo further experiments in this area due to time constraints. Another point is the percentage of the vocabulary (i.e. the total set of words occurring in the corpus) available as a pre-trained continuous word vector. When Google News word embeddings are used, for about 60,7% of the unique words present in the data an associated word vector can be found. Compared to the 88% of the movie review contained in them, this value is quite low, which could be another reason for the relatively poor performance of our model. The reason is probably that whereas news articles are usually well-written and contain very few errors, user-generated online reviews instead often come along with a lot of misspellings and contain colloquial words and phrases which are not used in more sophisticated language. Consequently, the word embeddings trained on reviews from the Electronics category contain 88,1% the vocabulary in the training and testing data, which is not surprising since they were trained on the corpus from which the reviews were ultimately drawn. The difference to 100% results from omitting words with a very low frequency of one during the embedding training.

When it comes to data size, we observe a steep decline in performance (by about 10 percentage points AUC) when data size is halved. The decline is not as big but still significant when we drop another 50% of the data (using DS-DL3). This underlines how sensitive deep

learning architectures are to varying training data sizes, at least on this level. On the other hand, the linear model's performance also declines sharply with data size. When using DS-DL4 for model training, we obtain the best performance overall (just slightly better than with the smaller DS-DL1 and the same parameters). In the *smallest data, reduced complexity* setup, we test whether reduced architecture complexity helps to mitigate the effect of smaller data and reduce the number of filters significantly, with the result of an even poorer performance.

Regarding the different hyper-parameter settings, we find that longer filter sizes are surprisingly not superior to shorter ones in this context. In our experiments, a set of filters of lengths 3, 4 and 5 delivers marginally better (about 0.25%) accuracy than the same number of filters of lengths 5, 10 and 15. Johnson & Zhang (2015) find that longer filter work better for topic classification (longer phrases contain more words which hint to a specific topic) whereas shorter filters deliver better performance in sentiment prediction (sentiment is often expressed in short phrases). In the short filter setting, we also examine the effect of reducing the total number of filters (while keeping filter sizes constant) and, as expected, observe a clear drop in performance.

Also, all else constant, increasing model complexity by increasing the number of neurons in the hidden layer by a factor of ten does not aid performance. We have also tested average-pooling as an alternative to the global max-pooling in an earlier experimental setup without a separate test set, but found that it leads to such significant performance that we did not test it again in the final Keras based setup. The same is true for not using global max-pooling. When looking at the class probabilities produced for each review, we found that the output came close to a uniform distribution. This was possibly due to the trailing zeros, which can make up a large part of a single review and bias the results accordingly when average-pooling on the whole review is used. With global max-pooling, zeros are automatically omitted. Finally, we also test the static setup (word embeddings are kept constant during the training process) and random initialization of word embeddings, but, as does Kim (2014), we also find that these setups deliver worse performance than the non-static one.

The fact that our deep-learning based models are not significantly superior in this setup is surprising, as recent research has shown that these kind of models work very well on text classification problems.[104] However, the considered tasks were mainly sentiment classification (binary or multi-class), as well as document topic and question classification. To our knowledge, spam classification using CNNs has not been considered in the literature yet.

[104] See Kalchbrenner (2014); Johnson & Zhang (2015); Kim (2014)

It differs from the other tasks in the way that spam reviews aim to closely resemble real ones, in order to purposely mislead readers. Hence, they are difficult to detect even for human observers.[105] It is hard to say what features really identify an untruthful review. Hence, it is also difficult to assess what factors influence model performance in spam classification. Our most successful model uses global pooling over the whole review and relatively short filter lengths. This means a review is basically classified based on the single most predictive word sequence. In contrast, the BOW approach as used in our linear model discards word order completely and only rates each review on the words occurring in it. Based on these architectural differences, we could argue that the plain occurrence of specific words in a review alone has more predictive power than the occurrence of specific phrases. It could be that spammers use some words significantly more often than truthful reviewers, as they spend less time and effort per review and do not want to express their opinion in the most accurate way possible. In this case, a single phrase of fixed length as captured by the CNN's filters may not be as informative as the occurrence of specific words in the whole review.

Additionally, we have to consider that our labelling technique may provide a better base for the linear, BOW-based model. Since reviews were marked as spam based on word similarity, the reviews in the spam class probably resemble each other more closely than those in the randomly sampled non-spam class. These similarities are potentially easier to pick-up when considering global word occurrences instead of a single phrase.

In order to test the comparability of VW and CNN results, we evaluate the former on the movie review data where Kim (2014) achieves state-of-the-art performance. With an accuracy value of 0.788 (the only metric reported) it ranks behind the CNN, but roughly in the middle of the other classifiers. This proves that, at least for specific data, the CNN is indeed able to outperform a linear model that itself is able to reach a competitive score. The fact that this happens on a relatively small data set with about ten thousand observations is important here, as recent research has shown that traditional approaches, such as BOW combined with n-grams, tend to outperform deep learning based models on data with less than several hundreds of thousands samples and deep architectures begin to shine only on data sets larger than half a million observations.[106] Zhang et al. (2015) obtain better performances for BOW based models when data size is roughly half a million samples or below. Here, the best non-deep model outperforms the best deep one by 56% to 4.5%. For larger datasets, deep models win with a margin between 0.5% and 38%. Our data is well below this size, i.e. insufficient data

[105] Ott et al. (2013), p.312f
[106] Zhang et al. (2015), p.5f

size could also be a reason for the superior performance of the linear approach. The marginally better performance we obtain with the DL-DS4, which contains reviews of up to 200 words and about 80.000 reviews, and the much worse performance of the models trained on about 50% and 25% of the data are an indicator that data size may indeed plays a significant role here.

There are numerous other settings imaginable here in order to try to close the gap between the linear and deep learning model. The space of possible hyper-parameter combinations to explore is so large we have barely scratched the surface. Besides, we did not test other deep-learning architectures, which might have performed slightly better and thus closed the gap further.

Description		*Hyper-Parameters*			*Performance*			
Data set	Setting Name	Filter sizes	No. of Filters	Hidden Neurons	AUC	Accuracy	F-score	LR Difference
DL-DS1	Wide filters	5, 10, 15	50	50	0.8677	0.812	0.7948	-1%
DL-DS2	Smaller data	5, 10, 15	50	50	0.768	0.7066	0.7062	-0.2%
DL-DS3	Smallest data	5, 10, 15	50	50	0.7266	0.6717	0.6488	1.7%
DL-DS3	Smallest data, reduced complexity	3, 4	5	10	0.6859	0.6412	0.6521	-4%
DL-DS4	Long sentences	5, 10, 15	50	50	0.8682	0.8124	0.8099	1.7%
DL-DS1	Short filters	3, 4, 5	50	50	0.8677	0.812	0.7948	-1%
DL-DS1	Few filters	3, 4, 5	5	50	0.8395	0.7382	0.769	-4%
DL-DS1	Many Neurons	3, 4, 5	50	500	0.8516	0.8005	0.78	-2.8%
DL-DS1	Custom Embeddings	3, 4, 5	100	100	0.8765	0.8161	0.8027	0%

Table 5.6 - Deep Learning based classification results by data, features and hyper-parameter setting

6 Conclusion

In this work, we summarized most of the existing literature on review spam and took a holistic view on the topic in order to find promising techniques to fight spammers. We aimed to provide a broad overview on different topics and use a more realistic setting than most research has so far.

With the goal of being able to successfully train deep-learning models, we collected the largest data set by far in this area containing more than 50.000 spam reviews using authentic real world user reviews posted on Amazon's website. Our labelling technique was based on review similarity under the assumption that highly similar reviews are copied by spammers for convenience. We trained a linear model using a large set of experimental setups, ranging from differently sampled data to various feature extraction techniques to class imbalance. Finally, we used a deep-learning architecture which showed outstanding performance in related tasks for review spam classification. The results obtained in our experiments did often match the performance reported by other authors for similar settings.

Our main finding in this regard is that a powerful linear classification tool, such as Vowpal Wabbit, and a simple bag-of-words using (small) n-grams is enough to obtain a performance that is mostly superior to models based on more sophisticated features, and hence makes manual feature engineering obsolete.

Class imbalance issues definitely play a role in review spam detection. With our labelling approach we only found about 1% spam reviews in the Amazon data and about 0.5% in the Yelp data set. As expected, performance drops with increasing imbalance, and measures to overcome this effect, such as altering the probability cut-off, depend primarily on the individual use case.

Regarding deep-learning-based approaches, we use a Convolutional Neural Network architecture which has recently achieved highly competitive scores in related tasks, namely sentiment classification. Similar architectures have also been used for document topic and question classification, mostly beating conventional techniques. The architecture consists of only one convolutional layer, followed by a pooling layer and ultimately, a fully-connected layer. Compared to architectures used in other research areas such as computer vision, it is rather shallow. We test various hyper-parameter combinations but find that most of our models are not competitive compared to the linear models. The one exception is when pre-trained word embeddings are used which were created in an unsupervised fashion with *wod2vec* algorithms and based on more than 7 million Amazon reviews. The problem specific

word vectors account for colloquial language and misspellings present in user-generated reviews, among other things. Although the difference in performance is not massive, this emphasizes the power of *word2vec* based models which can take advantage of a large amount of (potentially unlabelled) data. Data size in general can be regarded as a decisive factor, since we find that model performance decreases drastically when data size is reduced. The role of other hyper-parameters is not as substantial in our experiments.

Despite a broad range of experiments carried out, there is still a massive amount of possible settings to be examined, ranging from deeper architectures to completely different deep-learning models or word embeddings trained on more data. This work can be considered a foundation for further research in this area, providing useful performance baselines and possible future experimental setups.

7 References

Akoglu, L., Chandy, R., & Faloutsos, C. (2013). Opinion Fraud Detection in Online Reviews by Network Effects. *ICWSM*, *13*, 2-11.

Amazon Sues Five Sites Promising Reviews For Cash. (2016). Retrieved June 02, 2016, from *https://consumerist.com/2016/04/26/amazon-sues-five-sites-promising-reviews-for-cash/*

Andoni, A., & Indyk, P. (2006, October). Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. In *Foundations of Computer Science, 2006. FOCS'06. 47th Annual IEEE Symposium on* (pp. 459-468). IEEE.

Banerjee, S., & Chua, A. Y. (2014, August). Applauses in hotel reviews: Genuine or deceptive?. In *Science and Information Conference (SAI), 2014* (pp. 938-942). IEEE.

Bastien, F., Lamblin, P., Pascanu, R., Bergstra, J., Goodfellow, I., Bergeron, A., ... & Bengio, Y. (2012). Theano: new features and speed improvements. *arXiv preprint arXiv:1211.5590*.

Bengio, Y. (2009). Learning deep architectures for AI. *Foundations and trends® in Machine Learning*, *2*(1), 1-127.

Bengio, Y., Ducharme, R., Vincent, P., & Jauvin, C. (2003). A neural probabilistic language model. *journal of machine learning research*, *3*(Feb), 1137-1155.

Bengio, Y., Ducharme, R., Vincent, P., & Jauvin, C. (2006). Neural probabilistic language models. In *The Journal of Machine Learning Research*, *3*,(pp. 1137-1155).

Bergstra, J., Bastien, F., Breuleux, O., Lamblin, P., Pascanu, R., Delalleau, O., ... & Bengio, Y. (2011). Theano: Deep learning on gpus with python. In *NIPS 2011, BigLearning Workshop, Granada, Spain*.

Borowiec, S. (2016, March 15). AlphaGo seals 4-1 victory over Go grandmaster Lee Sedol. Retrieved June 23, 2016, from *https://www.theguardian.com/technology/2016/mar/15/googles-alphago-seals-4-1-victory-over-grandmaster-lee-sedol*

Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K., & Kuksa, P. (2011). Natural language processing (almost) from scratch. *The Journal of Machine Learning Research*, *12*, 2493-2537.

Crawford, M., Khoshgoftaar, T. M., Prusa, J. D., Richter, A. N., & Al Najada, H. (2015). Survey of review spam detection using machine learning techniques. *Journal of Big Data*, 2(1), 1.

Fayazbakhsh, S. K., & Sinha, J. (2012). Review Spam Detection: A Network-based Approach. *Final Project Report: CSE*, *590*.

Fernández-Delgado, M., Cernadas, E., Barro, S., & Amorim, D. (2014). Do we need hundreds of classifiers to solve real world classification problems?. *The Journal of Machine Learning Research*, *15*(1), 3133-3181.

Furnas, G. W., Deerwester, S., Dumais, S. T., Landauer, T. K., Harshman, R. A., Streeter, L. A., & Lochbaum, K. E. (1988, May). Information retrieval using a singular value decomposition model of latent semantic structure. In*Proceedings of the 11th annual*

international ACM SIGIR conference on Research and development in information retrieval (pp. 465-480). ACM.

Furnas, G. W., Deerwester, S., Dumais, S. T., Landauer, T. K., Harshman, R. A., Streeter, L. A., & Lochbaum, K. E. (1988, May). Information retrieval using a singular value decomposition model of latent semantic structure. In *Proceedings of the 11th annual international ACM SIGIR conference on Research and development in information retrieval* (pp. 465-480). ACM.

Goodfellow, I., Bengio, Y., & Courville, A. (n.d.). Deep learning. Retrieved October 4, 2016, from *http://www.deeplearningbook.org/*

Ho-Dac, N. N., Carson, S. J., & Moore, W. L. (2013). The effects of positive and negative online customer reviews: do brand strength and category maturity matter? *Journal of Marketing*, *77*(6), 37-53.

Japkowicz, N., & Stephen, S. (2002). The class imbalance problem: A systematic study. Intelligent data analysis, 6(5), 429-449.

Jindal, N., & Liu, B. (2008, February). Opinion spam and analysis. In*Proceedings of the 2008 International Conference on Web Search and Data Mining* (pp. 219-230). ACM.

Johnson, R., & Zhang, T. (2015). Effective use of word order for text categorization with convolutional neural networks. arXiv preprint arXiv:1412.1058.

Kalchbrenner, N., Grefenstette, E., & Blunsom, P. (2014). A convolutional neural network for modelling sentences. arXiv preprint arXiv:1404.2188.

Kieler, A. (2016). Amazon Sues Five Sites Promising Reviews For Cash. Retrieved April 14, 2016, from https://consumerist.com/2016/04/26/amazon-sues-five-sites-promising-reviews-for-cash/

Kim, Y. (2014). Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*.

Kim, Y. (2014). Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*.

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems* (pp. 1097-1105).

Kuhn, M., & Johnson, K. (2013). *Applied predictive modeling* . New York: Springer.

Langford, J. (n.d.). Vowpal Wabbit. Retrieved April 11, 2016, from *https://github.com/JohnLangford/vowpal_wabbit/wiki*

Lau, R. Y., Liao, S. Y., Kwok, R. C. W., Xu, K., Xia, Y., & Li, Y. (2011). Text mining and probabilistic language modeling for online review spam detection.*ACM Transactions on Management Information Systems (TMIS)*, *2*(4), 25.

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature*, *521*(7553), 436-444.

Leskovec, J., Rajaraman, A., & Ullman, J. D. (2014). Mining of massive datasets. Cambridge University Press.

Li, F., Huang, M., Yang, Y., & Zhu, X. (2011, July). Learning to identify review spam. In *IJCAI Proceedings-International Joint Conference on Artificial Intelligence* (Vol. 22, No. 3, p. 2488).

Li, H., Liu, B., Mukherjee, A., & Shao, J. (2014). Spotting fake reviews using positive-unlabeled learning. *Computación y Sistemas*, *18*(3), 467-475.

Long, N. H., Nghia, P. H. T., & Vuong, N. M. (2014). Opinion spam recognition method for online reviews using ontological features. *Tạp chí Khoa học*, (61), 44.

Luca, M., & Zervas, G. (2015). Fake it till you make it: Reputation, competition, and Yelp review fraud. *Harvard Business School NOM Unit Working Paper*, (14-006).

Manber, U. (1994, January). Finding Similar Files in a Large File System. In*Usenix Winter* (Vol. 94, pp. 1-10).

McAuley, J., Targett, C., Shi, Q., & van den Hengel, A. (2015, August). Image-based recommendations on styles and substitutes. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval* (pp. 43-52). ACM.

Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013a). Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., & Dean, J. (2013c). Distributed representations of words and phrases and their compositionality. In*Advances in neural information processing systems* (pp. 3111-3119).

Mikolov, T., Yih, W. T., & Zweig, G. (2013b). Linguistic Regularities in Continuous Space Word Representations. In *HLT-NAACL* (pp. 746-751).

Mukherjee, A., Venkataraman, V., Liu, B., & Glance, N. (2013a). *Fake review detection: Classification and analysis of real and pseudo reviews*. UIC-CS-03-2013. Technical Report.

Mukherjee, A., Venkataraman, V., Liu, B., & Glance, N. S. (2013b). What yelp fake review filter might be doing?. In *ICWSM*.

Ott, M., Cardie, C., & Hancock, J. (2012, April). Estimating the prevalence of deception in online review communities. In *Proceedings of the 21st international conference on World Wide Web* (pp. 201-210). ACM.

Pennebaker, J. W., Boyd, R. L., Jordan, K., & Blackburn, K. (2015). The Development and Psychometric Properties of LIWC2015. *UT Faculty/Researcher Works*.

Pennington, J., Socher, R., & Manning, C. D. (2014, October). Glove: Global Vectors for Word Representation. In *EMNLP* (Vol. 14, pp. 1532-1543).

Rajaraman, A., & Ullman, J. D. (2012). *Mining of massive datasets* (Vol. 1). Cambridge: Cambridge University Press.

Rakhlin, A. (2016). CNN for Sentence Classification in Keras. Retrieved July 17, 2016, from https://github.com/alexander-rakhlin/CNN-for-Sentence-Classification-in-Keras

Rong, X. (2014). word2vec parameter learning explained. *arXiv preprint arXiv:1411.2738*.

Schmidhuber, J. (2015). Deep learning in neural networks: An overview. *Neural Networks*, *61*, 85-117.

Scott, S., & Matwin, S. (1999, June). Feature engineering for text classification. In *ICML* (Vol. 99, pp. 379-388).

Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

Streitfeld, D. (2012). The Best Book Reviews Money Can Buy. Retrieved June 02, 2016, from *http://www.nytimes.com/2012/08/26/business/book-reviewers-for-hire-meet-a-demand-for-online-raves.html*

Streitfeld, D. (2012, August 26). The Best Book Reviews Money Can Buy. Retrieved March 11, 2016, from *http://www.nytimes.com/2012/08/26/business/book-reviewers-for-hire-meet-a-demand-for-online-raves.html*

Sun, H., Morales, A., & Yan, X. (2013, August). Synthetic review spamming and defense. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining* (pp. 1088-1096). ACM.

Voutilainen, A. (2003). Part-of-speech tagging. *The Oxford handbook of computational linguistics*, 219-232.

Yelp. (n.d.). Retrieved March 11, 2016, from *https://www.yelp.com/dataset_challenge*

Zhang, X., Zhao, J., & LeCun, Y. (2015). Character-level convolutional networks for text classification. In Advances in Neural Information Processing Systems (pp. 649-657).

8 Appendix

	Our study	Jindal & Liu (2008)	Ott et al. (2011)	Mukherjee et al. (2013)	Li et al. (2011)
Data Source	Amazon	Amazon	AMT	Yelp	epinions
Spam Reviews	54,377	4488	400	9,170	1,398
Features					
BOW	x		x	x	x
TF-IDF					
POS	x		x	x	
LIWC			x	x	
% pos. & neg. words		x			x
product name mentioned					x
Review Meta Data					
First review	x	x			x
Standalone review	x	x			
Character Length	x				
Word length	x	x			x
Summary length	x	x			
Review rank by time	x	x			
Review Age	x				
Overall Product Rating	x	x			x
Standard dev. Product R.	x	x			x
#Helpfulness votes	x	x			
#Helpful votes	x	x			
#Helpfulness in %	x	x			
Opposite 2nd review flag	x	x			
Reviewer	x				
First vs. 2nd Person used					x
Product mentioned flag	x				
Product	x				
Reviewer Meta Data					
#Reviews	x				x
Average rating	x	x			
Standard dev rating	x	x			x
Date first review					
Date last review					x
Reviewer lifetime	x				
% same brand reviewed					
Max. reviews per day					
Ratio of first reviews	x	x			
Ratio of standalone rev.	x	x			
Always good, bad, avg. ratings flag	x	x			
Only good & bad flag	x	x			
Only bad & avg. flag	x	x			
Only good & avg. flag	x	x			

Good, bad and avg. flag	x	x		
Ratio of opposite 2nd reviews	x	x		
% most common category	X			
Product Meta Data				
Price	x	x		
Sales rank	x	x		
Avg rating	x	x		x
Standard dev. rating	x	x		
#Reviews	x			x

Table 8.1 - Features used in selected studies on review spam detection. Lau et al. (2011) use the exact same feature set as Jindal and Liu (2008).

Erklärung zur Urheberschaft

Hiermit erkläre ich, Martin Müller, dass ich die vorliegende Arbeit allein und nur unter Verwendung der aufgeführten Hilfsmittel angefertigt habe.

Die Prüfungsordnung ist mir bekannt. Ich habe in meinem Studienfach bisher keine Bachelorarbeit eingereicht bzw. diese nicht endgültig bestanden.

Martin Müller

Berlin, 28.12.2016