Doctoral Dissertation

A framework for developing requirements engineering tools for computational business intelligence

March 2021

Doctoral Program in

Advanced Information Science and Engineering

Graduate School of Information Science and Engineering
Ritsumeikan University

KOVACS Mate

Doctoral Dissertation Reviewed
by Ritsumeikan University

A framework for developing requirements engineering tools for computational business intelligence

(ビジネスインテリジェンスにおける 要求工学ツール開発のためのフレームワーク)

March 2021
2021 年 3 月

Doctoral Program in Advanced Information Science and Engineering
Graduate School of Information Science and Engineering
Ritsumeikan University

立命館大学大学院情報理工学研究科
情報理工学専攻博士課程後期課程

KOVACS Mate
コバーチ　マーテー

Supervisor : Professor KRYSSANOV Victor
研究指導教員 : クリサノフ　ビクター教授

Contents

List of Tables

List of Figures

Abstract

Online reviews available on e-commerce websites (such as Amazon, Yahoo, Yelp, TripAdvisor, Rakuten, etc.) are short textual documents written by customers about the products and services they buy. This form of electronic word-of-mouth is considered to be the leading driving force of consumer purchase decision making, and can provide highly valuable information not just for the customers but also for the companies. Knowledge derived from product reviews can help companies develop and improve their products and services by integrating relevant information into the requirements engineering process. As the amount of reviews grows over time, however, both companies and customers experience an information overload. With the vast amount of customer reviews available on online platforms at present days, companies need computational tools to analyze the reviews with minimal human intervention and extract valuable information from them to assist the iterative process of requirements engineering.

In this thesis, a conceptual framework is proposed for developing requirements engineering tools for computational business intelligence. The tools and systems developed using the proposed framework would reduce the information overload associated with online reviews and extract valuable knowledge about customer needs. To make the framework suitable for requirements engineering, the following objectives have been achieved:

- *Investigating and improving data quality is essential for industrial applications. In the presented study, a theoretical model with a novel measure called pertinence is introduced to assess the quality of reviews for requirements engineering purposes.*
- *In order to obtain detailed information on customer needs, companies often require using small, hand-labeled datasets. In this study, an original approach is proposed to incorporate external knowledge into machine learning models and use deep learning algorithms more efficiently with small datasets.*

Two case studies have been conducted to test the practicality and effectiveness of the proposed framework. While the first one implements a system for estimating sentence level review informativeness, a multi-class sentiment classification tool is developed in the second case study. Experimental results indicate that the framework proposed in this work is applicable for developing requirements engineering tools.

Chapter 1

Introduction

Businesses gather customer information to enhance their current goods and services, facilitate new product innovation, and detect trends of customer behavior for marketing activities. Analyzing the customer's voice and knowing their needs and expectations will increase customer satisfaction and reinforce customer loyalty, which is strongly linked to the company's long-term growth, providing an edge over rivals in the global market [1, 2]. Therefore, acquiring data about customers can boost the efficiency of information-based decision making and make Business Intelligence (BI) more efficient [3].

BI includes technologies, applications, methodologies, and tools that allow efficient business data analytics by optimizing operational and strategic business decisions. Efficient BI is crucial to the success of every modern company in the global economy [4]. One of the most essential areas of BI is dealing with the acquisition and implementation of customer requirements. Requirements engineering is the process of gathering, defining, validating and

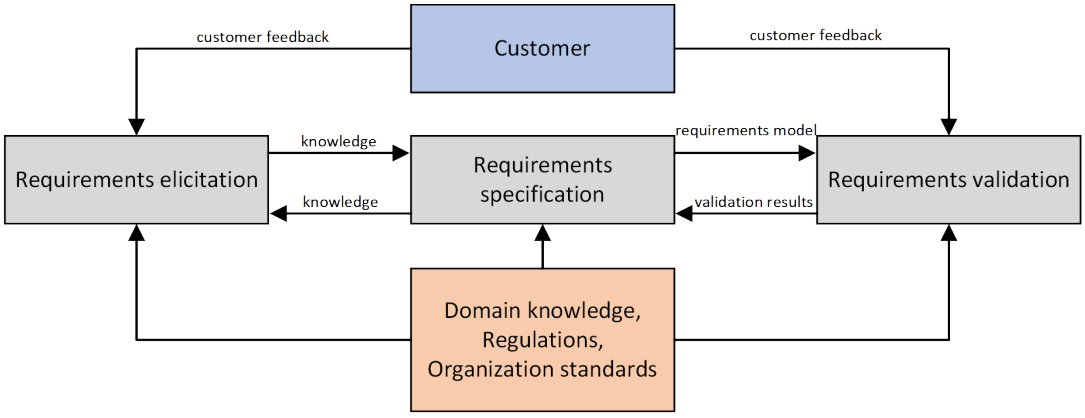

Figure 1.1: Iterative model of the requirements engineering process

maintaining requirements during the design process of a product or service [5]. Figure 1.1 shows the process of requirements engineering, based on the iterative model originally proposed by Loucopoulos and Karakostas [6]. The model consists of three basic stages: elicitation, specification, and validation. Requirements elicitation is the process of investigating and

exploring customer requirements. Requirements specification refers to a collection of techniques and methods used to acquire information about the project domain and specifications. During this stage, all functional and non-functional requirements are defined formally to produce requirements models ready to be deployed in design and production. Requirements validation ensures that the requirements set out are complete and satisfy customer needs, by eliminating the inconsistencies of requirements elicitation and specification while monitoring customer feedback. Poor understanding of customer needs can lead to erroneous assumptions during requirements elicitation and product conceptualization [7]. This can have an adverse impact on the manufacturing process and may negatively affect development cost and lead time. Businesses must, therefore, be aware of and adequately respond to customer requirements, especially in the early stages of new product development [7, 8].

E-commerce appeared more than 25 years ago, and the popularity of online shopping has been growing ever since [9]. This transition in purchasing behavior has resulted in the exponential growth of web-based user-generated content. One of the key factors behind consumer behavior has always been word-of-mouth (WOM) communication [10]. These subjective opinions are often seen as a more reliable source of information than advertisements and product descriptions because WOM communications are produced by actual customers [11]. Presently, conventional marketing strategies have a lower impact on the customers' decision-making process than eWOM (Electronic Word of Mouth) [12, 13], and it is known that the opinions of other customers included in online reviews affect individual purchasing decisions [14, 15]. For their online and even offline transactions, more than 90% of customers read reviews [16] and consider other people's feedback as they can effectively search for the product or service that fits their personal preferences [17]. Being one of the most prominent eWOMs, several studies have highlighted the value of online customer reviews and their relation to product innovation and their impact on sales [18–21]. While reviews are always subjective, these opinions should be addressed in product development if such subjective views tend to occur regularly [22].

Online reviews are reasonably simple to obtain and posted by customers without much corporate effort [23], while being a reliable and comprehensive source of information on customer needs, as customers often describe personal observations, opinions, and feelings on various product features [24, 25]. Nowadays, manufacturing is becoming more and more customer-driven [26]. Customarily, companies use interviews, polls, and surveys to receive feedback

from their consumers, but the information on customer requirements derived from product reviews varies from and complements customer intelligence gathered by conventional methods [27, 28].

For the reasons mentioned above, many e-commerce websites rely on this type of user-generated content to attract new customers and enhance user experience, and retailers encourage customer to leave their feedback on the products they bought. Extracting consumer intelligence and constructing useful product and service requirements from such user-generated content, however, is a time and money consuming task, since it requires handling of natural language data [7, 29]. Moreover, the sheer number of reviews available makes it almost impossible for businesses to assess the quality of user-generated content manually. Therefore, companies need systems that allow for a large number of reviews to be analyzed and relevant content to be identified.

While both knowledge-based and machine learning systems are used in present days, systems of the latter kind are more popular, due to the benefits modern machine learning tools offer (limited feature engineering yet superior performance). Since deep learning applications, however, require huge datasets to build robust models, it is not always feasible to develop high-performance systems, especially in the case of low-resource languages. Even with popular languages like English or Chinese, the labeling process to create adequate-sized datasets can be long and resource-intensive, especially when the prediction problem is complex. Some of these issues have been addressed by hybrid learning methods, as inclusion of external knowledge into machine learning methods can be effective to improve classification performance [30, 31]. The majority of prior work incorporating knowledge from lexicons, ontologies, etc., however, focused primarily on binary classification of opinions [32], which is usually considered as an imprecise measure often not suitable for real-word applications. Another approach is to implement hand-made rules about the target domain and task (before or after machine learning takes place) to increase system performance. Although it can be effective for certain application areas [33], such methods usually require substantial human intervention.

While websites with a large volume of helpful reviews attract more buyers [34], the immense amount of reviews of varying quality are often overwhelming for both the customers and the companies [35, 36]. The amount of customer feedback available on web platforms makes it a challenging task for businesses to collect and process relevant knowledge on customer requirements. Popular goods receive thousands of reviews from buyers, and

review quality differs greatly across the reviews [35, 36]. Addressing this problem, also referred to as *information overload* [37–39], is critical for using customer reviews effectively for improving products and services [40]. The most popular approach to deal with this issue is to measure review helpfulness. The term "review helpfulness" has an ambiguous meaning in the related literature. However, it mainly represents how much customer uncertainty can be reduced while shopping via a useful product or service evaluation [41–43]. The helpfulness of online reviews could be affected by many qualitative and quantitative factors [41, 44, 45]. Review helpfulness determinants can be divided into two main groups: content and non-content related factors (see Figure 1.2). While non-content-related properties (e.g. reviewer-related

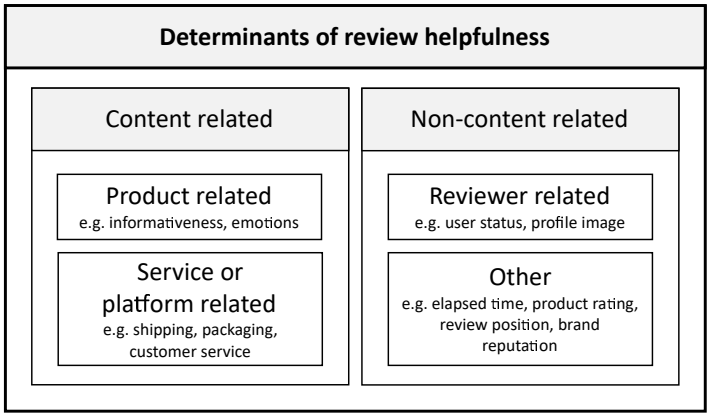

Figure 1.2: Conceptual overview of the determinants of review helpfulness

properties) also influence the customer-perceived quality of reviews, content-related factors (e.g. user experience) are often seen as the major factors of review helpfulness. Review helpfulness is particularly affected by "review informativeness" [37, 41, 46, 47]. Review informativeness is often interpreted as a combined measure of information quality and quantity relevant to the review content. While among other review features, informativeness has the greatest effect on general review helpfulness [46], there is an obvious disparity between the helpfulness experienced by the consumers and the helpfulness perceived by the product designers [48]. For example, for product designers and other people participating in requirements engineering, service or platform related factors are irrelevant, while some features related to informativeness are critical. The majority of studies concerned with the helpfulness of customer reviews only take into account the customer's perspective, and little work dealt with the quantification of design knowledge

present in reviews to assist product or service designers, engineers and other experts working together on the requirements engineering process. Without filtering relevant and high-quality material, however, it is an arduous task to successfully develop or adopt new technologies for BI applications, and many data-mining projects fail because the quality of data is not adequate for industrial applications [49].

In this thesis, a conceptual framework is proposed for developing requirements engineering tools for computational BI. The framework would be used to build systems and tools that can potentially reduce the information overload of customer reviews and extract meaningful knowledge about the needs of the customers for requirements engineering purposes. The following criteria had to be satisfied to make the framework fit for requirements engineering tool development:

1. The framework must include the assessment of review quality, preferably in an automatic manner.
   Addressing the quality of online customer feedback is essential to reduce information overload and to select useful reviews for requirements engineering. Since "review helpfulness" is a relative and ambiguous concept, the measure of quality must be specialized for requirements engineering.

2. The framework must be able to handle both small and large datasets according to the target task, and should have the capability to utilize external knowledge sources (e.g. processing models, lexicons, etc.).
   While for some tasks, huge datasets are available and feasible to use, acquiring detailed information about customer needs often requires dealing with small, human-annotated datasets with external knowledge sources involved.

3. The framework should be technology-independent enough to be usable with both state-of-the-art and near-future technologies.
   This is important because technologies change rapidly, and building an entirely new framework for requirements engineering tool development is a more complex task than replacing technologies.

Two case studies have been conducted with English and Japanese customer reviews to investigate the effectiveness of the proposed framework.

The rest of this thesis is structured as follows. Chapter 2 surveys related work on computational approaches to learn about customer needs, and

presents prior work on analyzing review quality. The chapter also introduces the technological background related to this study. The proposed framework for developing requirements engineering tools is presented in Chapter 3. Case studies performed to examine the effectiveness of the proposed framework are explained, and experimental results are presented and discussed in Chapter 4. Chapter 5 gives an overall discussion on the proposed framework, based on the results obtained from the case studies. Chapter 6 draws conclusions and outlines the limitations of the given study.

Chapter 2

Background and literature survey

This chapter describes previous work dealing with the extraction of meaningful information from customer feedback to potentially enhance the effectiveness of BI operations by learning about customer needs. Prior approaches to analyze review quality and helpfulness are introduced, and technological challenges related to the implementation of state-of-the-art tools for efficient requirements engineering are outlined.

It has been known for a long time that companies should interact with their customers to track customer satisfaction, as it is strongly related to their purchasing decisions. Customer need assessment is traditionally done by conducting interviews with the customers (e.g. focus groups). Although such qualitative methods can be fine-grained, one of the problems is that the subjects often hold back their responses and tend not to answer sincerely. The significance of collecting customer feedback through anonymous surveys was pointed out in various studies [50, 51]. Such quantitative methods perform well at uncovering what is required to satisfy the current needs of the customers, but are restricted by many factors. For instance, the number of participants is typically low, the range of questions is limited, and the content is constrained by the responsible persons' background and their ability to recognize hidden needs and new opportunities [3]. Godes et al. [52] manually conducted customer feedback mining and analysis through customer forums of Usenet. Although online conversations are relatively easy to obtain, the approach proved to be methodologically inefficient in the long run, as human assessment of eWOM is a tedious and costly task to perform. When dealing with extensively large datasets, it is practically unfeasible to analyze customer feedback manually. Human involvement, therefore, must be minimized, and domain experts of marketing and product design must cooperate with data scientists for effective requirement engineering [53–55].

2.1 Computational approaches to learn about customer needs

Due to the possible industrial implications, using computational methods to extract customer needs was always a popular topic in the data mining community. Chen et al. [56] built a system for design specification generation and product conceptualization. The system is built of a customer requirement elicitation and a customer marketing analysis module. It uses the laddering technique to create a customer attribute hierarchy, and implements a radial basis function neural network trained on manually-annotated attribute importance ratings for further marketing analysis. In a later study, the authors proposed a customer utility prediction system [57]. This system also consists of two modules. One creates design options in a hierarchical form using general sorting. The other is for measuring "customer desirability", by using conjoint analysis and a neural network to make predictions about the customers' preferences on basic product features. Lee et al. [58] developed a system that can process customer reviews semi-automatically. The system summarizes the reviews, and creates word vectors using word co-occurrences. The vectors are clustered according to their Euclidean distances, and conjoint analysis is applied to elicit different attribute levels for the products. Although the system developed requires manual feature engineering, one of the proposed approach's strength is that it allows for generating product attributes from reviews rather than from the descriptions by the manufacturers. Zhang et al. [59] developed a tool that can potentially assist engineers capturing product or service design information to meet customer expectations. After manually defining a set of features relevant to the domain, the system creates a feature specific product graph and ranks products based on their quality. Later streams of research focused on topics such as consumer behavior prediction [60–62], and customer satisfaction estimation [22, 63, 64], but Sentiment Analysis (SA) became the most popular method to learn about customer needs.

SA is the process of computational identification of sentiments, emotions, and perceptions toward entities and their aspects [65]. In present days, SA is one of the most fundamental ways to learn about customer needs computationally [66], and therefore, it must be in every middle to big company's BI toolbox to enhance the requirements engineering process. SA from customer reviews is one of the most prominent study fields of text mining that draws growing interest from both the research community and

industry, as it proved to be a reliable measure of overall customer satisfaction [67]. SA can be done at three distinct levels [68]. Although on the document level, sentiments are evaluated in the context of the entire document, sentence level sentiment analysis deals with the extraction of opinions from individual sentences. Finally, aspect-based techniques aim to identify people's sentiments on the target features of an entity. Although regression methods do exist, usually SA is defined as a straightforward binary classification task to assess sentiment polarity, or as a multi-class classification problem, sometimes involving more abstract categories (e.g., emotions, attitudes), creating a more difficult machine learning task.

Most early studies in SA utilized polarity dictionaries, opinion lexicons, and linguistic rules to perform sentiment classification, or used these jointly with conventional machine learning algorithms to achieve superior performance (Support Vector Machines, Naive Bayes, Maximum Entropy, etc.). Recent years' research, however, shows that purely machine learning-based approaches, particularly deep learning algorithms, are typically more powerful than knowledge-based methods [69]. One of the reasons for the success of deep learning in text mining applications is that language modeling involves learning the relations between sequential components, and Recurrent Neural Networks (RNN) are exceptionally good at learning such temporal dynamics [70]. Deep learning algorithms, however, typically require a significant volume of labeled data to achieve adequate performance [71], which is often unfeasible to acquire. For this reason, researchers and developers often convert user ratings (usually stars) to labels from the reviews, which mitigates the need for human annotation.

Various neural network models were proposed for SA, making use of the star system. A joint architecture of recurrent and Convolutional Neural Network (CNN) models has been proposed by Wang et al. [72] for sentiment classification on short textual data. Experiments were conducted on two binary and a 5-class dataset (with the additional "very negative", "neutral" and "very positive" categories), utilizing the 5-star system fully. Glorot et al. [73] investigated the issue of domain adaptation for document sentiment polarity classification. The authors introduced a system using a stacked Denoising Autoencoder with built-in sparse rectifier units to extract textual features in an unsupervised manner. Experimental results have shown that these features can be used efficiently with supervised sentiment classifiers. Chen et al. [74] combined customer information and product information on the global level for sentiment classification. A Long Short-Term Memory (LSTM) neural network is used to generate document and sentence

representations, where customer and product information is incorporated into the model via word and sentence-level attentions.

Although utilizing user ratings alongside with the review text is a common and convenient way to conduct SA, there are several downsides of this approach. While it allows for using large datasets (due to the fact that the reviews essentially include the labels), it is a crude way to measure customer opinion since it is on the document level, aiming to assign an overall polarity score to a document (that is, a full customer review). Often, there are both positive and negative assessments of a product or service in a review, and using the stars for labels cannot account for such cases. Furthermore, the most common categories used for classification are a.) negative or positive, and b.) negative or neutral or positive [65], which are not fine-grained enough to acquire detailed knowledge about customer sentiments. The reason companies still use binary or three-class document level SA is that annotating on the sentence or aspect level using more abstract categories (e.g. emotions, attitudes) requires time and money, and the created datasets are usually not large enough to build robust machine learning models.

There are a couple of studies aim at acquiring detailed information about customer needs via SA. Dos Santos and Gatti [75] used a CNN where one of the convolutional layers extracts word-level features, and the other tries to capture sentence-level characteristics to perform SA on the sentence level. The model was applied to binary and a more fine-grained classification scheme (very negative, negative, neutral, positive, very positive). Wang, et al. [76] proposed a joint CNN and RNN architecture in which recurrent layers learns long-term dependencies, while the convolutional layer captures local features. Besides binary classification, model performance was also tested on a dataset labeled for both valence and arousal on the sentence level, and experimental results showed that the combination of these two network architectures is a powerful method for SA. A few studies investigated the efficiency of hybrid methods by combining multiple classification strategies. Poria et al. [77], combined linguistics cues with machine learning to enhance the accuracy of sentiment polarity classification. Their strategy was to use the linguistic patterns present on SenticNet [78] if applicable, and use a supervised machine learning model otherwise. A different kind of hybrid approach was proposed by Appel et al. [79] for sentence level SA. The system uses SentiWordNet [80] and a sentiment lexicon with fuzzy sets to classify the polarity of sentences into the categories of positive or negative, with identifying the strength of the sentiment. Huang et al. [81] suggested a tree-structured LSTM network in which Part of Speech (POS) tags handle the recurrent network gates to

encode additional syntactic information about phrases and sentences. The authors managed to encode POS tags into conventional RNN and also LSTM networks, and achieved superior performance compared to the vanilla models in the typical three-class (negative-neutral-positive) classification.

While some of the previous work recognizes the utility of using external data sources alongside machine learning models, current methods do not suggest to include external knowledge directly into the network architecture to acquire more efficient sentence or document representations, especially when the classification task is more complex than a conventional three-class or binary sentiment polarity classification scheme.

2.2 Methods for assessing review quality

Ranking reviews, based on a certain quality criteria, improves customer experience and helps companies in the requirements engineering process by filtering out weakly-related information. In SA studies, review content is rarely debated [82], even if most reviews appear virtually useless for designers and marketing experts. Many e-commerce websites have adopted helpfulness voting, where users rank reviews from other users based on their perceived helpfulness. Amazon, for example, asks readers, "Was this review helpful?" to encourage users to show whether they think a particular review was helpful or not. Although can be beneficial, such social voting mechanisms are highly biased by the Matthew effect [83] and are widely argued being unreliable measures for estimating the helpfulness of online reviews [36, 37, 47, 83–87]. Since most customers only read and vote for the already top reviews, these tend to remain at the top, and older reviews naturally receive more votes than freshly-posted ones. Furthermore, most of the currently deployed helpfulness voting mechanisms are highly imprecise indicators of quality, as the vote function is for the entire review.

Studies addressing the topic of predicting review quality based on various criteria mostly tried to classify reviews into two or more classes [88–91]. Other methods include using regression to score or rank reviews [23, 92–94]. Another line of research has focused on exploring the features that affect review helpfulness [42–45, 95–97]. A variety of features were suggested for predicting the helpfulness of online reviews, such as readability [88, 89, 98], polarity [99], subjectivity [37], extremity [98], syntactic, semantic and lexical features [88–90, 100–102], meta-data [100, 101], etc. Descriptions of different product features are generally associated with more detailed and long reviews, and review length is widely regarded as a significant aspect of review

helpfulness [42, 91, 96]. However, using review length as a feature is not an ideal method for measuring review helpfulness if information quality is not addressed [37, 41]. Mostly, long reviews include only a handful of phrases about product attributes and usability, and "useless" text decreases classification performance [41, 44, 103].

Krishnamoorthy [88] has suggested a hybrid approach to predict review helpfulness based on review metadata, linguistic attributes (e.g. the usage of particular verbs and adjective types), with subjectivity and readability features. Liu et al. [47] trained an SVM model on a human-annotated dataset of digital cameras reviews from Amazon. The authors concluded that informativeness features, such as the number of product aspects and the number of brand names, increase helpfulness classification accuracy. In order to create a ranking among mobile phones, Gobi and Rathinavelu [104] developed an approach to recognize basic features from product reviews via soft clustering techniques. Although the approach mainly identifies fundamental features, it is capable of detecting attributes expressed both explicitly and implicitly. Cohen and Tseng [105] introduced an information quality framework to identify product features and determine the quality of customer reviews. The authors adopted features such as the volume of information, believability, understandability, timeliness, reputation, etc., and performed experiments on a human-annotated corpus. Saumya et al. [106] conducted binary quality classification experiments on Snapdeal.com and Amazon.in reviews. Findings indicate that in addition to using review text, including information from question-answer data can boost review classification performance. Using product reviews from JD.com, Sun et al. [41] investigated the role of review informativeness on helpfulness. Review informativeness, defined by a selection of quantitative features (for example, the number of attributes), was found to improve the prediction performance of helpful reviews as a determinant of review helpfulness. Lee and Choeh [101] used a neural network regression model to estimate the helpfulness of Amazon.com customer reviews. According to their findings, textual features (review length, number of words, etc.) and contextual variables (e.g. product sales rank, price) are important factors for predicting user-perceived helpfulness of a review.

Although there has been limited research on evaluating review quality for different target groups (such as product designers), there are still a few studies addressing this issue. Liu et al. [48] measured the utility of reviews from the viewpoint of the product designer to investigate what are the most important factors of review helpfulness in the case of product design. Four types of

features have been classified as relevant. These include linguistic features (e.g., word count), product features (aspects of a given product), informational attributes (e.g., number of entities referred to), and information-theoretic features (e.g., review sentiment polarity). Yagci and Das [28] suggest that design intelligence that benefits both designers and consumers can be derived from product reviews. The authors performed sentence level SA (with negative, neutral, positive classes), and used noun-adjective and noun-verb association heuristics to identify the possible cause of an opinion. In another study, the authors implemented the design-level information quality (DLIQ) metric to determine the amount and quality of design information present in online customer reviews [107]. Reviews were analyzed based on quantity (total number of words), complexity (e.g. noun count), and relevance (number of nouns matching manually defined features), and promising results were reported for assisting companies in design and development.

Most of the preceding work does not distinguish between the helpfulness seen by the customers and product designers. While informativeness is frequently considered to be one of the essential factors determining the helpfulness of reviews, only a handful of studies have dealt with it explicitly, excluding non-content-related and service-related factors. Recognizing product features is crucial for assessing the quality of product reviews for requirements engineering. In the related literature, this was usually achieved by manual feature engineering of explicit attributes based on manually-constructed feature lexicons and pattern-lists, applying linguistic heuristics, and other labor-intensive yet arbitrary and frequently imprecise methods. In natural languages, the same aspect can be represented in different ways (explicitly or indirectly, with different terms and phrases, etc.), and pattern matching techniques are not suitable to deal with such cases. In addition, the presence of a particular word does not inherently indicate high-quality material since the context of a word plays a crucial role in its meaning and interpretation. Although sentence level assessment has been commonly used for text processing in a variety of fields (e.g. sentiment analysis, discourse parsing, novelty identification, etc.), this type of fine-grained content evaluation is seldom examined in customer review helpfulness research.

2.3 Technological background and challenges

Training a neural network from scratch to acquire adequate results is a time -and resource-consuming process, as often millions of labeled datapoints

have to be collected to train a robust model. For this reason, researchers and developers frequently use pretrained deep learning models. The performance of the model, however, is often severely impaired when it is used for a prediction task other than the original. Transfer learning allows one to use knowledge from a previously learned task, and adapt it to a new but similar task. Transfer learning is formalized as follows [108]. A domain $\mathcal{D}$ has a feature space $\mathcal{X}$ with a marginal probability distribution $P(X)$, where $X \in \mathcal{X}$ and $X = \{x_1, x_2, \cdots, x_n\}$. Given a domain $\mathcal{D} = \{\mathcal{X}, P(X)\}$, the task consists of a label space Y and an objective function $f : \mathcal{X} \rightarrow Y$. The latter is used to predict the label $f(x)$ of a new datapoint x. The task $\mathcal{T} = \{Y, f(x)\}$ is learned from the training data $\{x_i, y_i\}$, where $x_i \in X$, and $y_i \in Y$. Given a source domain and task $\mathcal{D}_S$ and $\mathcal{T}_S$, the goal of transfer learning is to increase the learning performance of the objective function of the target $f_T(\cdot)$ in $\mathcal{T}_T$, applying the knowledge learned from the source domain and task. The typical transfer learning setup is illustrated in Figure 2.1.

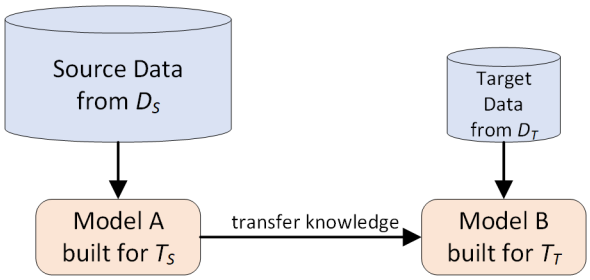

Figure 2.1: The typical setup for transfer learning

In practice, this means that the weight matrices of the original model are used for the initialization of the weights in the new model, instead of setting them randomly. In an ideal scenario, this would mean that researchers and developers can utilize a pretrained, open-source deep learning models which eliminate the need of training a network from scratch, and can fine-tune previous models even on small datasets successfully.

In various situations, however, there is an unavoidable change in the data distribution of the source and the target domains, or tasks $\mathcal{T}_S$ and $\mathcal{T}_T$ are greatly different, so transfer learning will inherently lead to a performance decrease [109]. The term *negative transfer* applies to situations when transferring previously learned knowledge to a new model results in a decrease in performance compared to a model where transfer learning is not utilized [110].

Transfer learning is widely used in various fields of computer science, and in Natural Language Processing (NLP), the most popular transfer learning applications are pre-trained word embedding models. Most deep learning models for text processing require word embeddings as inputs representing natural language words and sentences. Word embeddings are real number vectors, capable of capturing word meaning in a high-dimensional continuous vector space. Although each dimension of the vectors represents a feature, these are abstract, low-level features interacting with each other, usually impossible to grasp by humans. While these vectors are "high-dimensional" (usually, between 128-1024), modern word embedding techniques are far more sophisticated than early attempts to create word vectors, e.g. one-hot encoding, where all words in the dictionary occupy a dimension. One of the main purposes of advanced word embedding methods is to capture not just the meaning of words with syntactic and morphological information, but also to incorporate the contextual relationships between words. Words appearing in similar contexts tend to be semantically relatable, and their vectors should reflect this characteristic. Unlike the Euclidean distance, cosine distance can be reliably used to compare vectors of high dimensionality, and it is a well-accepted, common measure of word vector similarity [111, 112]. Cosine similarity *cossim* between vectors $\mathbf{G}$ and $\mathbf{H}$ is defined as:

$$cossim = \cos(\theta) = \frac{\mathbf{G} \cdot \mathbf{H}}{\|\mathbf{G}\|\|\mathbf{H}\|} = \frac{\sum_{i=1}^{n} G_i H_i}{\sqrt{\sum_{i=1}^{n} G_i^2}\sqrt{\sum_{i=1}^{n} H_i^2}}. \tag{2.1}$$

Mikolov et al. [111] introduced Word2Vec in 2013, a huge advancement in creating word vectors utilizing the representative power of neural networks. The authors proposed two variations of the embedding model: Continuous Bag-of-Words (CBOW) and Skip-Gram. CBOW predicts the central word in a window of words around it, while Skip-Gram is the opposite, predicting the context words of the central word. The basic idea of CBOW and Skip-Gram is depicted in Figure 2.2, where $w(p)$ refers to word w observed at position p. The words in the input layers are one-hot-encoded. Compared to previous word embedding approaches, both versions of Word2Vec highly outperformed previous language models, with slightly superior performance of Skip-Gram over CBOW. Word2Vec, however, cannot vectorize Out Of Vocabulary Terms (OOV) words and unable to address word disambiguity.

In comparison to predictive embedding models, such as Word2Vec, word vectors of GloVe [112] are learned by creating a word-word co-occurrence

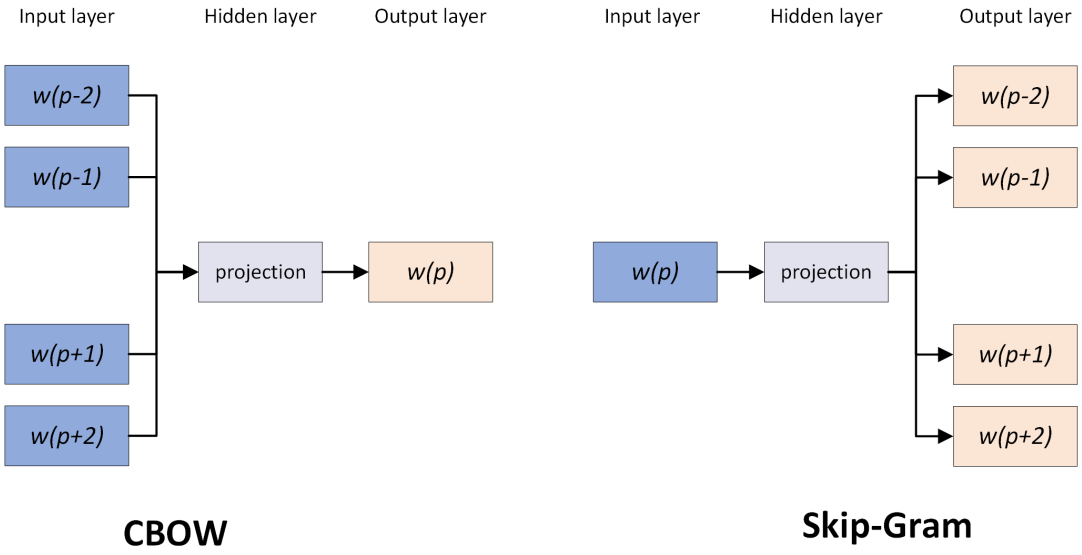

Figure 2.2: High-level overview of the two models of Word2Vec

matrix, then using matrix factorization to acquire vectors in a lower-dimensional space. Although predictive methods typically depend on local information (direct word context), GloVe extracts not only local but also global information from the whole corpus, alleviating some of the problems with Word2Vec. Before the training takes place, the word co-occurrence matrix G is built, where G_{ij} denotes how often word i observed together word j. For each word combination, the embedding vectors $\mathbf{e}$ for words i and j are explained as

$$\mathbf{e}_i^T \mathbf{e}_j + b_i + b_j = \log(G_{ij}), \tag{2.2}$$

where b_i and b_j are bias terms for word i and word j. The goal is to minimize the following objective function J:

$$J = \sum_{i,j=1}^{V} f(G_{ij})(\mathbf{e}_i^T \tilde{\mathbf{e}}_j + b_i + \tilde{b}_j - \log(G_{ij}))^2, \tag{2.3}$$

where $\mathbf{e}_i$ is the word vector for word i and $\tilde{\mathbf{e}}_j$ is the associated context vector with biases b_i and $\tilde{b}_j$, and V denotes the vocabulary size. f is a weighting

function with parameter α usually set to 0.75, defined as:

$$f(x) = \begin{cases} (x/x_{max})^{\alpha} & \text{if } x < x_{max} \\ 1 & \text{otherwise} \end{cases} \tag{2.4}$$

The output is the two sets of word vectors, W and $\tilde{W}$, only different because their initialization was done randomly. This property, however, helps reduce possible overfitting, as to get the final, unique vectors, the corresponding vector pairs are summed up. Due to its ability to capture global statistics about the corpus, GloVe tends to perform better in word similarity benchmarks and is considered to be a preferable choice over Word2Vec for certain NLP tasks. Nevertheless, both models are commonly used for creating word vectors for machine learning applications in various fields in data science.

Advanced language modeling methods allow for context-dependent, sentence specific, and dynamic word representations. Embeddings from Language Models (ELMo) [113] computes word vectors based on entire sentences, as oppose to using static, predefined vectors. In short, ELMo deploys a pre-trained deep Bidirectional LSTM (BiLSTM), to calculate word vectors in an unsupervised way. Unlike conventional LSTM, BiLSTM networks process the input text in forward and backward directions at the same time. The input layer for the bidirectional architecture are vectors from a character level CNN, and the output vectors are weighted averages of the internal layers. Although proved to be a superior method compared to previous embedding techniques, the method was outperformed by BERT (Bidirectional Encoder Representations from Transformers) [114] in the same year, establishing a new, long-standing standard in language modeling. In order to handle sequential data and represent contextual relations between words, BERT utilizes a transformer with self-attention mechanism. Originally, transformers involve two different parts: the textual input is fed into the *encoder*, and the *decoder* produces predictions. Since the objective of BERT is to build a language model, the decoder part is not required. The pretrained BERT-Base model involves 12 stacked encoder layers with 12 attention heads per layer. The model input is a word sequence, and after applying self-attention at an encoder layer, the outputs from the attention heads are moved to a 768-unit feed-forward network before passing the outputs to the next encoder layer. Figure 2.3 shows the stacked arrangement of encoders used in BERT.

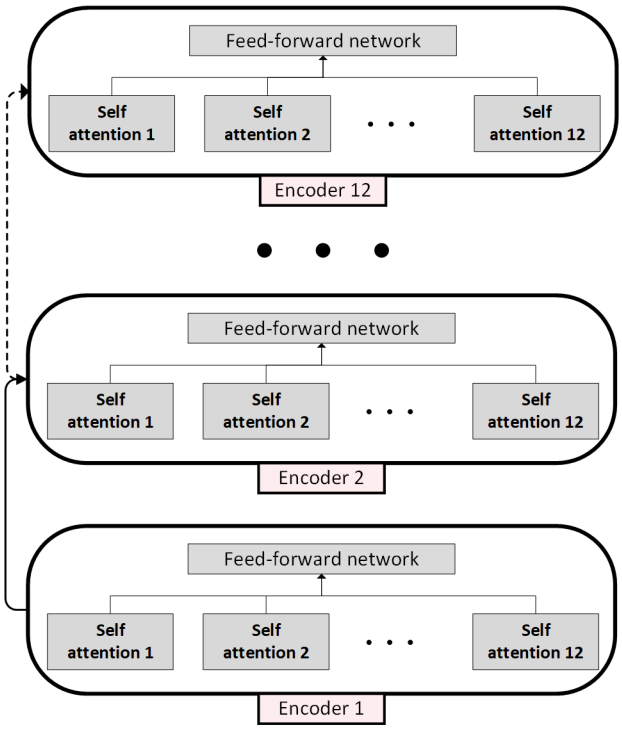

Figure 2.3: The stacked encoder structure of BERT

Most language models predict the next word in a word-sequence to pretrain the model in a one-directional way. Instead of using this method, BERT applies masked language modeling. 15% of words are originally hidden (masked) from the input sequence, and non-hidden words are used to infer masked words based on word positions. During training, BERT calculates the probability of each word in a sequence with softmax, but only the prediction of hidden words are considered in the loss function. The input sequences are given to the model in the form of pairs of sentences to formulate a second task: to predict if the sentences are subsequent or not (50% true subsequent and 50% random for training). Before a sequence is fed into the encoder-stack, it is processed to have segment and position embeddings besides token embeddings (vocabulary IDs). While segment embeddings indicate sentence order, position embeddings signify word positions. There are three kinds of special tokens used for the input for BERT:

1. [MASK] is a pretraining token for masked words.
2. [SEP] is a pretraining sequence delimiter for sentence-pair tasks. In the case of a single sentence, it is used as the last token.

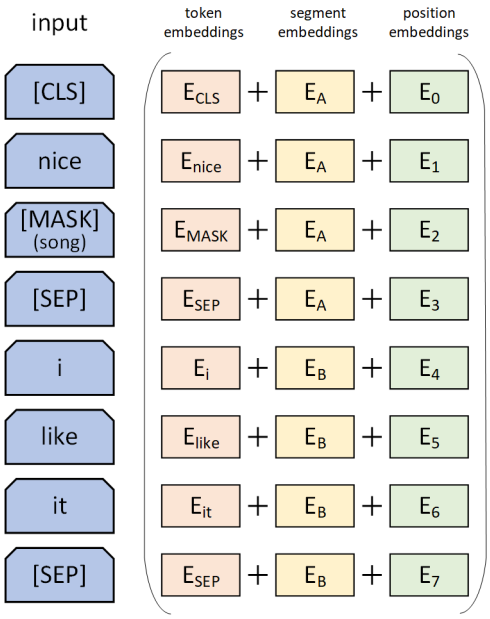

Figure 2.4: The input representation for BERT (see main text for notations)

3. [CLS] is the first token of the sequences used for classification tasks (with a softmax layer).

Accordingly, the input representation for training BERT is shown in Figure 2.4 with an example sentence. After the whole input sequence has been fed into the model, the probability of the subsequent sentence is calculated. The final cost function is the combined loss obtained from the prediction of masked words and next sentence predictions.

In BERT, the raw embeddings from the pre-trained model are transformed by using the context of words of the input sentence, making the vectors context-dependent. First, all the raw word token vectors of a sequence $U = \{u_i \mid i = 1, ..., n\}$ representing the n tokens of the input are put into the attention heads at the first layer. These vectors are then transformed into Query, Key, and Value vectors by multiplying the token vectors with three matrices learned during the pretraining phase, W_Q, W_K, and W_V, respectively. The self-attention weights between all the word pair combinations are calculated by taking the dot product between the Query vectors q and Key vectors k of word token vector u, then normalizing it by the softmax function. The output of an attention head for

word token vector u_i is the weighted sum of the Value vectors v:

$$output(u_i) = \sum_{j=1}^{n} \frac{\exp(q_i k_j^T)}{\sum_{m=1}^{n} \exp(q_i k_m^T)} v_j, \tag{2.5}$$

As BERT uses multi-headed attention (12 in total for BERT-Base) to give the layers multiple representation spaces, there are multiple output matrices for each layer. These matrices are concatenated and multiplied by a fourth weight matrix W_O that is trained together with the other weight matrices. With this operation, a single output matrix is obtained, possessing condensed information from all attention heads. This matrix is then put into a feed-forward network before entering the next layer.

BERT is a general language model that supports transfer learning applications, and usually, it is not utilized purely for feature extraction and acquiring word vectors. BERT is mostly built into the network architecture, so that the model can utilize the sentence specific embeddings directly. For certain applications, however, it would be desirable to obtain stationary vectors, where one word is bound to a single vector, while still holding information about word usage in the target domain. Using the raw embeddings from the pre-trained model, however, would not be ideal, as these vectors do not contain any domain specific information, similarly to the previously introduced Word2Vec or GloVe.

2.4 Related literature summary

Most of the studies dealing with extracting or evaluating customer needs did not consider the feasibility of the approaches in real-world scenarios.

- Assessment of customer reviews should be on the sentence or aspect-level to obtain usable information about requirements.
- While it can be applied to big data without human annotation, the usability of binary classification of sentiments is questionable.
- Although some of the studies consider using external data sources with machine learning systems, integrating them into deep learning models directly is usually not considered, although offering new opportunities methodologically.

While review quality is rarely discussed in customer reviews studies, a

considerable amount of information is not just useless for requirements engineering purposes, but also depreciates the overall value of reviews.

- Most of the research dealing with the quality/helpfulness of customer reviews does not define "quality" or "helpfulness" clearly.
- Although there is a clear difference between the two concepts, the majority of studies do not make a distinction between the helpfulness seen by the customers and the helpfulness perceived by product or service designers.
- Recognizing product or service features to filter out high-quality material is usually done by methods involving a considerable amount of human intervention while still being an imprecise measure of quality.

There are multiple technological challenges researchers and developers face when building systems for industrial applications, and make them relevant for requirements engineering.

- While advanced word embedding techniques allow one to capture the contextual meaning of words, it is a challenging task to obtain stationary vectors from them, and keep domain knowledge incorporated into the embeddings.
- Although transfer learning methods attempt to alleviate the need to use large datasets for deep learning applications, because of the potential source-target task discrepancy, transfer learning is not always a feasible option.

Chapter 3

Proposed approach

This chapter introduces the proposed framework for developing requirements engineering tools for computational business intelligence, aiming to mitigate the issues summarized in Section 2.4.

3.1 Overview

Figure 3.1 provides an overview of the proposed framework. There are two main parts of the framework, development, and deployment. It is assumed that the users can acquire customer review data and have access to the necessary facilities (e.g. storage, server, network, workstations, etc.). Since this work focuses on the development stage, the deployment part is not described in detail, and it is only outlined on a high level. Nevertheless, it is necessary to include it in the framework, as there are iterative processes embedded in a loop structure.

The data collector interface transmits the raw data into the Data transformation module, where the data is transformed into the required format for model creation. Model creation can have several iterations internally until model performance reaches an acceptable level, depending on the specific task at hand. The stored model(s) are read in by an interface connected to a live server that gives the output on newly acquired and transformed data. Model outputs are stored in a database, ready to be used by the responsible individuals involved in the requirements engineering process (e.g. product designers, market researchers, etc.). Model outputs are also used to validate the deployed model and to further fine-tune it if necessary. After a large amount of new raw data (unseen during initial model training) is collected, these are merged with the old raw data, and some parts of the model creation phase are repeated to potentially achieve better performance.

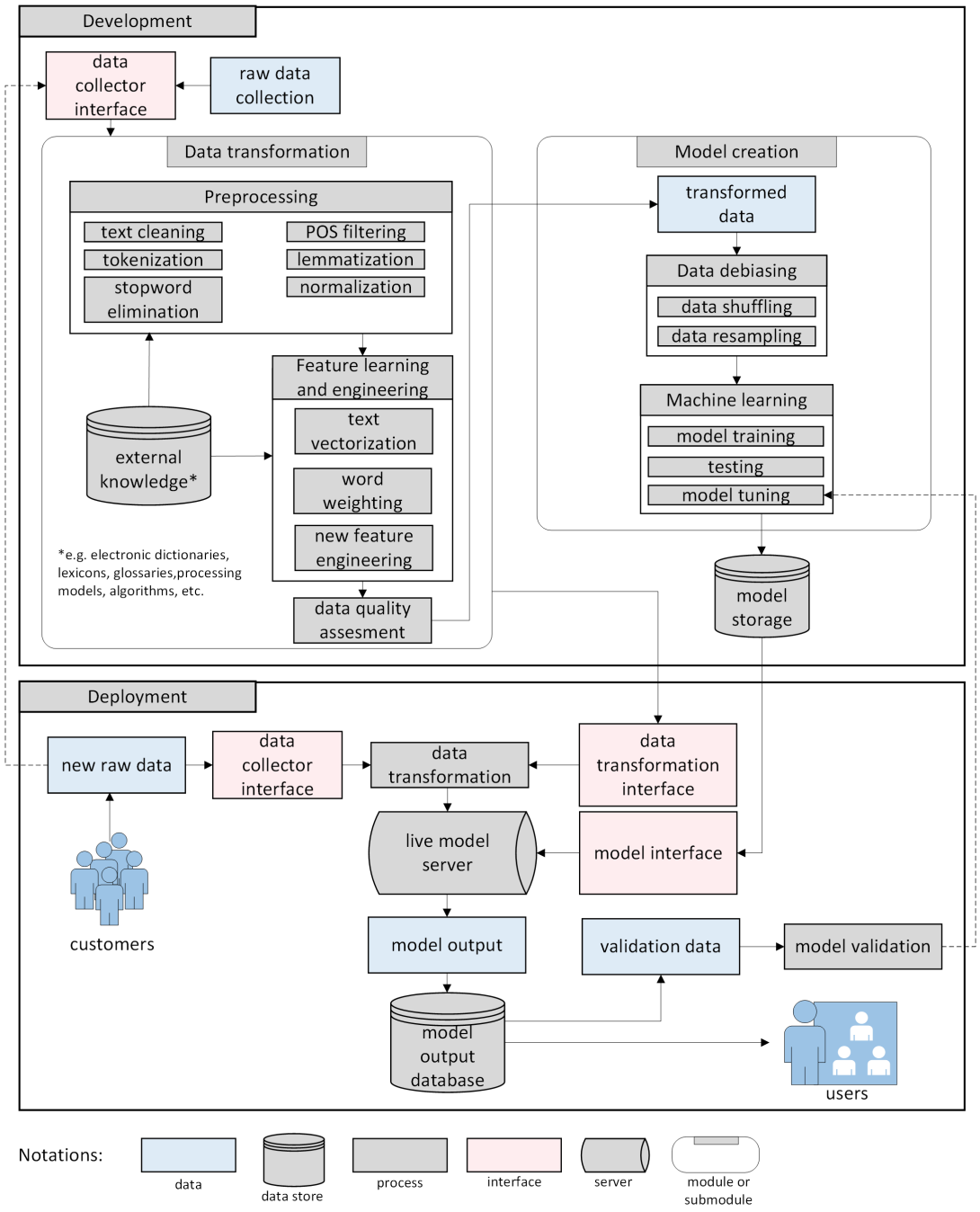

Figure 3.1: Overview of the proposed framework

3.2 Theoretical model of document pertinence

Instead of defining what "helpfulness" is in the context of customer reviews, a more general, widely applicable theoretical model is proposed to describe the quality of reviews. The *pertinence* of a document defines the document's relevancy and applicability in a certain domain, for a given target audience. The concept of pertinence is also used to create domain and task specific word vectors, and assign weights to the words, explained in later sections.

Intuitively speaking, the model proposed is based on the assumption that there is always a chance to encounter a certain word in natural languages, regardless of the type of the document. Depending on the document's target population T (intended target audience) and the domain V, occurrences of words w for a certain language $L = \{w_1, w_2, ..., w_m\}$ are probabilistic rather than binary, as although not all words carry the same importance in a certain domain, any word can appear in any kind of document d for all possible documents $D = \{d_1, d_2, ..., d_n\}$. Besides a word's literal, "hard-coded" semantics in a certain language, the meaning of a word is always influenced by the context. This pragmatic relationship between a word and its context is highly dependent on T and V.

For instance, one may find the word *sensor* in any type of text, but if the domain is *digital cameras*, and the target population is the *product designers*, this word is of great importance. If the domain stays the same and the target population combination is *university students*, the term *sensor* may still be significant, but it does not have the same weight. On the other hand, even if T is the same (*product designers*), if the domains are *speech recognition* and *digital cameras*, the usual context of the word *sensor* will be quite different, and this should be reflected in the corresponding word vectors in the different domains. Because it is such a general word, the term *comparison* is probably not highly-relevant for V = *digital cameras* and T = *product designers*, but this does not imply that it does not convey any information about V and T, particularly in the proper context, e.g. "Comparisons of megapixel ratings...".

Let us represent the probabilistic nature of word occurrence with word weights in accordance with V and T as $\Phi_V^T = \{\varphi_{V_1}^T, \varphi_{V_2}^T, ..., \varphi_{V_m}^T\}$, and the contextual meaning of words for V and T as $\Psi_V^T = \{\psi_{V_1}^T, \psi_{V_2}^T, ..., \psi_{V_m}^T\}$. The above can be formally defined as follows:

$$\forall V \forall T \forall w \forall d[d \mathrm{E} D \wedge w \mathrm{E} L \wedge \Diamond(w \mathrm{E} d) \wedge \exists \varphi_V^T \exists \psi_V^T (\varphi_V^T \mathrm{E} \Phi_V^T \wedge \psi_V^T \mathrm{E} \Psi_V^T \models (\varphi_V^T, \psi_V^T) \mathrm{B} w)], \tag{3.1}$$

where E defines the relationship *element of*, B signifies the association *belongs to*,

and $\Diamond$ is denoting the modal logical proposition of what cannot be disproved, therefore, is possible. In relation to domain V and a target population T, the pertinence of a document can be represented as the distance between the contextual meaning of words Ψ_d in a document d and Ψ_V^T, weighted by Φ_V^T. The concept of pertinence was introduced by the author in [115].

3.2.1 The inferential problem of approximating document pertinence

The inferential problem is to define Ψ_V^T and Φ_V^T in a numerical format, for a given V and T, to approximate the pertinence of document d. In the case of requirements engineering of products, T mostly refers to the product designers who participate in the requirements engineering process at some point. Since technical documents are presumed to include a large amount of knowledge relevant to product designers, the author argues that technical documents of V should be analyzed to define Ψ_V^T and Φ_V^T, where $T =$*product designers*. The conceptual process flow of calculating document pertinence is shown in Figure 3.2. Practically speaking, Ψ_V^T consists of context-dependent stationary vectors

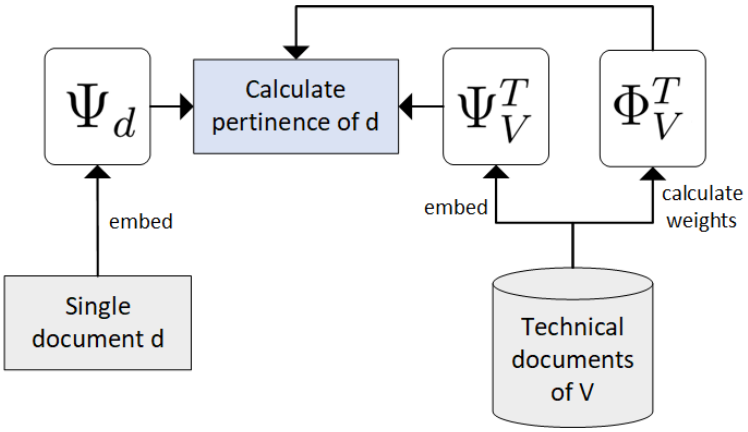

Figure 3.2: Conceptual process flow of calculating document pertinence

associated with the words in technical documents of V. Φ_V^T can be represented as scalar weights, computed by analyzing the same documents. The embedding model for creating Ψ_d should be the same that was used to extract vectors of Ψ_V^T to assure compatibility for the distance calculation.
The main idea of document pertinence approximation was introduced by the author in [115].

3.3 Data transformation

Data transformation focuses on converting the raw data into a format that can be used by the model creation phase. Preprocessing and feature engineering reduce data quality issues, and aims at producing representative inputs for further processing steps.

3.3.1 Preprocessing

The goal of preprocessing is to reduce noise and filter out evidently irrelevant information from the dataset. It is always task specific, and while some tasks require extensive preprocessing, some do not require it at all (for instance, when using external data sources that is already in the desired format).

- *Text cleaning* involves processes like removing typos and unneeded characters (e.g. emoticons, quotation marks), deleting duplicate entries, removing unnecessary parts of structured text (for example, author information).
- *Tokenization* is the task of slicing up documents and sentences into small pieces called tokens (i.e. words and punctuation).
- *Stopword elimination* is the process of filtering out "useless", commonly used words to reduce noise in the data.
- *Part of Speech (POS) filtering* means tagging every word with their corresponding POS, and removing unneeded categories (e.g. interjections).
- *Lemmatization* aims at reducing inflectional and derivational forms of a word to its common, dictionary form called lemma, based on morphological analysis and a dictionary (for example, all word "plays", "played", "playing" becomes "play").
- *Normalization* focuses on reducing inconsistencies in the dataset, e.g. removing too short and too long sentences, converting all synonyms of a certain entity to a single word, etc.

3.3.2 Feature learning and engineering

Feature engineering refers to a set of techniques used to extract features from data by applying domain knowledge, in order to enhance machine

learning performance. Feature engineering usually involves human intervention, but semi-automatic approaches exist to reduce the number of manual processes. Feature learning, on the other hand, is the process used to create feature representations without the need of manual feature engineering. While feature learning nowadays is considered to be a superior approach to discover and encode features, feature engineering techniques are still relevant due to the fact that high-performance feature learning usually requires a large amount of input data.

3.3.2.1 Extracting domain and target population specific word vectors

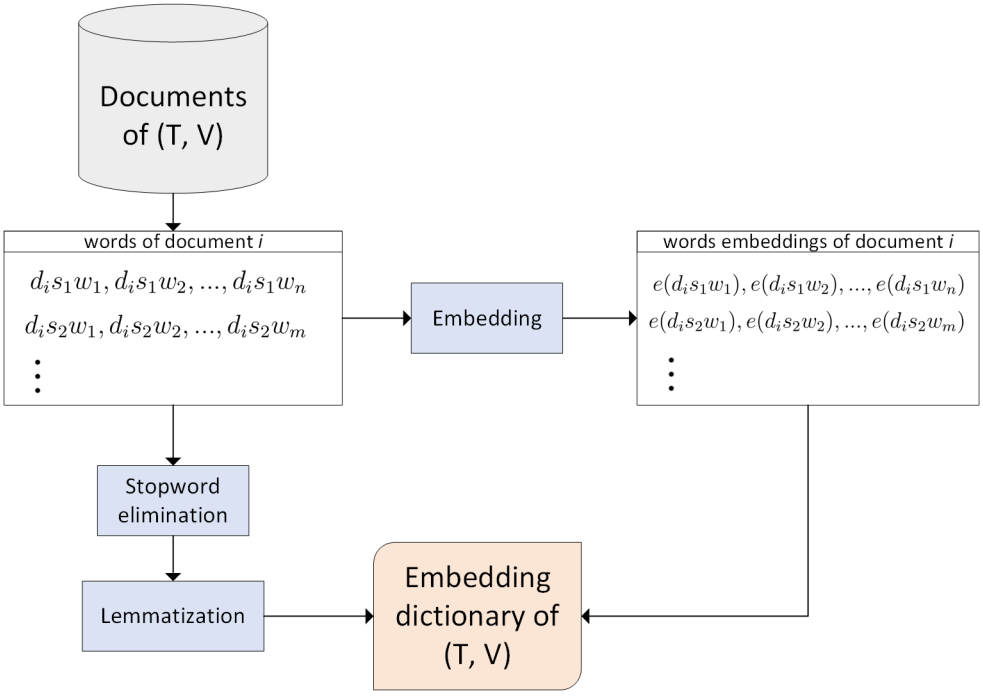

Figure 3.3: Extraction process of domain and task specific word vectors

The importance of word embeddings in modern BI systems and the related recent technologies were introduced in Section 2.3. While remarkable results were achieved by models like BERT, some applications would require extracting feature vectors from the model, instead of just using it as a part of a whole deep learning architecture. There are multiple ways to obtain the embeddings from a transformer models [114], but since word vectors of state-of-the-art models

are sentence specific and not stationary, the problem of acquiring single vectors for words while keeping most of the contextual information in a certain domain remains.

The approach proposed utilize the concept of pertinence, introduced in Section 3.2. The overview of the process with a sentence specific contextualized language model for embedding is shown in Figure 3.3. After a corpus of documents applicable for the target population and domain are collected, instead of retraining the language model that would likely end in negative transfer, all sentences s of the documents d are vectorized ($e(\cdot)$) using the pre-trained embedding model EMB. After stopwords are removed, all unique words w from the corpus vocabulary are put into the Embedding dictionary in their lemmatized form, with the corresponding word vectors. Since at this point, the contents of the Embedding dictionary are the unique word lemmas with all the word vectors $\mathbf{e}$ associated with them, the embeddings are averaged for their words to capture T and V as accurately as possible with stationary vectors $\psi \in \Psi$:

$$\psi_i = \frac{1}{R} \sum_{j=1}^{R} \mathbf{e}_{i,j}, \tag{3.2}$$

where R is the number or embeddings $\mathbf{e}_i$ associated with word w_i. Using this method, the "customary" contextual usage of words in the domain will be integrated into the embeddings. The computational process of calculating the vectors and putting them to the embedding dictionary *emb_dic* is defined in Algorithm 1.

The function EMB.**embed** is model specific. For example, in the case of BERT, the process of obtaining the word vectors is depicted in Figure 3.4.

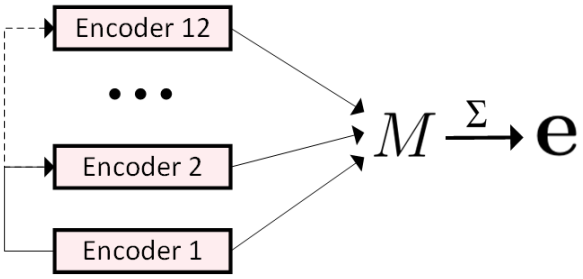

Figure 3.4: Process of extracting the embeddings $\mathbf{e}$ from BERT

Algorithm 1 Calculation of domain and target population specific word vectors

```
1: procedure VECTOR COMPUTATION(corpus, EMB)
2:     initialize dictionary emb_dic
3:     for all document in corpus do
4:         for all sentence in document do
5:             embed_sent = EMB.embed(sentence)
6:             for all word, vector in sentence, embed_sent do
7:                 lemma = lemmatize(word)
8:                 if lemma not in stopwords then
9:                     if lemma not in emb_dic then
10:                        insert lemma into emb_dic
11:                        emb_dic[lemma]= vector
12:                    end if
13:                else
14:                    append vector to emb_dic[lemma]
15:                end if
16:            end for
17:        end for
18:    end for
19:    for all term in emb_dic do
20:        average(emb_dic[term])
21:    end for
22:    return emb_dic
23: end procedure
```

For BERT, after the attention heads passed the weight matrices into the feed-forward network, the output will be a 768-dimensional vector for all 12 encoder layers. These vectors are put to a 12x768 embedding matrix M:

$$M = \begin{bmatrix} m_{1,1} & m_{1,2} & ... & m_{1,768} \\ m_{2,1} & m_{2,2} & ... & m_{2,768} \\ \vdots & \vdots & \vdots & \vdots \\ m_{12,1} & m_{12,2} & ... & m_{12,768}. \end{bmatrix} \tag{3.3}$$

The $N = 12$ rows of M (the vector outputs from the encoder layers) are summed to create a single embedding $\mathbf{e} = \{e_1, e_2, e_3, ..., e_{768}\}$, where

$$e_i = \sum_{j=1}^{N} m_{j,i}. \tag{3.4}$$

The word vectors computed this way can be used for any tool or application that requires domain and target population specific stationary vectors, and would constitute Ψ_V^T of the previously introduced document pertinence model.
The main idea behind extracting domain and target population specific word vectors is introduced by the author in [115], where the ELMo model [113] is used for embedding.

3.3.2.2 Word weighting

Word weighting is the process of assessing the importance of each word in a document collection. From a practical standpoint, it means assigning numerical values to words, in order to enhance the performance of information systems (e.g. for text classification, information retrieval, etc.). Usually, word weights reflect the extent of how relevant a particular word is to a document or a certain type of document in a given corpus.

Since the proposed method to calculate word weights is based on the pertinence model (Section 3.2), it aims to represent how valuable a word is in a collection of documents for T and V, i.e. establish Φ_V^T of scalar weights for the words observed in all documents combined. The weight φ_w of word w is determined as

$$\varphi_w = \ln\left(\frac{N}{o_s^w}\right)\frac{o_w}{N}, \tag{3.5}$$

where o_w is the frequency of word w in all documents, o_s^w is the number of occurrences of sentences s including word w. N stands for the total number of sentences in the documents. The weights with their corresponding words are put into the Weight dictionary for further processing.
The main idea of weight calculation was introduced by the author in [115].

3.3.2.3 Incorporating external knowledge into a deep learning model

The potential in incorporating external knowledge explicitly into deep learning models when the volume of data is low has been discussed in Section 2.1. The outline of the proposed approach (including model training for transparency) is depicted in Figure 3.5. The basic idea is that the two inputs (the textual data of training samples and the external knowledge source) are processed in two separate pipelines, and merged inside the network for further processing to drive the network to capture latent relationships between the two inputs, and potentially increase performance.

The training and test samples are embedded into word vectors to build the

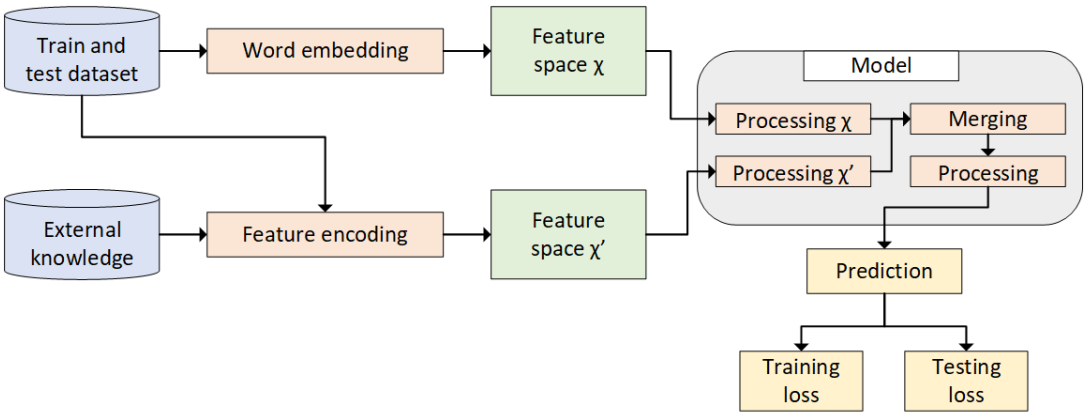

Figure 3.5: Incorporating external knowledge into a deep learning model

feature space of $\mathcal{X}$. In a separate pipeline, external knowledge is used to create an additional feature space $\mathcal{X}'$ for the words in the train and test datasets. The deep learning model first processes them without any connection to adjust representations independently during the training. Then these internal outputs are merged and further processed to extract the most meaningful representation possible in order to decrease training and testing loss. Thus, instead of using only the word-embedded feature space $\mathcal{X}$, it is used together with separate feature space $\mathcal{X}'$ to define the classification problem $f : \{\mathcal{X}, \mathcal{X}'\} \rightarrow Y$.

The author argues that the method described above would be useful when the number of training samples is low, but external knowledge sources exist for not particularly the same but similar classification (or regression) tasks. Feature encoding of the external knowledge to create feature space $\mathcal{X}'$ is data and task specific, and can be performed either by feature learning or feature engineering.

A practical example is multi-class sentence level SA, when the classification task is more complex than the usual binary or three-class polarity determination on the whole document level. While fine-grained SA is more beneficial for companies than simple binary polarity determination (Section 2.1), datasets annotated with detailed sentiments like emotions on the sentence level are usually small, especially in the case of low-resource languages. In such cases, using sentiment lexicons and polarity dictionaries labeled either binary (negative and positive) or on a continuous scale (e.g. between 0 and 1) would be beneficial in the proposed processing scheme. Since the network does not have enough training samples to learn the intricate differences and nuances between the classes, even if the classification task is different, a sentiment lexicon can still convey some high-level information

about the words that can potentially be advantageous during the learning process.
The main idea was proposed by the author in [116], where incorporating external data into a deep learning model was implemented using a custom network architecture with a sentiment lexicon.

3.3.3 Quantifying document pertinence to assess review quality

Before entering the model creation stage, the quality of the dataset is addressed by quantifying the pertinence of reviews for a given domain V and target population T. The pertinence of a single document is calculated according to the steps in Algorithm 2. Besides a corpus of documents and embedding model EMB, the inputs are the domain and target population specific word vectors Ψ_V^T from the Embedding dictionary *emb_dic* (Section 3.3.2.1) and their corresponding weights Φ_V^T from the Weight dictionary *weight_dic* (Section 3.3.2.2). Pertinence scores *pert_scores* are calculated on the word level. Word scores are allocated by calculating the cosine distances between the observed word's embedding vector *wordvec* and all term vectors $\psi \in \Psi_V^T$, then choosing the weight $\varphi \in \Phi_V^T$ of the closest term. Although merely summarizing the word scores would be skewed against longer reviews, taking the mean of the word or the sentence scores would totally neglect the length of the review *len*. Accordingly, review scores *sum_score* are divided by $len^{1/len}$ to get an adjusted score for pertinence. Since the calculated sentence scores are not within a predefined range, the scores are normalized between the interval [-1,1] before returning the final pertinence scores. Besides enhancing interpretability, the normalization helps with using the scored sentences in possible further processing steps.
The main idea of document pertinence calculation was introduced by the author in [115].

Algorithm 2 Calculation of document pertinence

procedure CALC_PERTINENCE(corpus, EMB, *emb_dic*, *weight_dic*)
  initialize list *pert_scores*
  for all document **in** corpus **do**
    $sum\_score = 0$
    for all word **in** document **do**
      $len = 0$
      if word **not in** stopwords **then**
        $len+ = 1$
        initialize array *cos_vector*
        $wordvec = EMB$.**embed**(word)
        for all ψ **in** *emb_dic* **do**
          $cosdis = 1 - \frac{\psi \cdot wordvec}{\|\psi\| \|wordvec\|}$
          append *cosdis* **to** *cos_vector*
        end for
        $closest\_word = emb\_dic[\mathbf{argmin}(cos\_vector)]$
        $\varphi = weight\_dic[closest\_word]$
      end if
      $sum\_score+ = \varphi$
    end for
    $pert\_score = \frac{sum\_score}{len^{1/len}}$
    append *pert_score* **to** *pert_scores*
  end for
  for all *pert_score* **in** *pert_scores* **do**
    $pert\_score = 2\frac{pert\_score - \mathbf{min}(pert\_scores)}{\mathbf{max}(pert\_scores) - \mathbf{min}(pert\_scores)} - 1$
  end for
  return *pert_scores*
end procedure

3.4 Model creation

Although it can take a relatively long time to transform data to the desirable format, model creation is sometimes even longer due to the iterative nature of the machine learning phase. Since the goal is to achieve model performance as high as possible, multiple iterations of training are performed with several combinations of hyperparameters. Also, to realistically assess model performance, often multiple, entirely different models are built and compared.

3.4.1 Data debiasing

Reduction of bias in the dataset is important to achieve reliable results at the end of model creation. While shuffling the datapoints should be done in almost all cases (an exception would be time series prediction), resampling is only necessary if the dataset is imbalanced (skewed class proportions). Downsampling means only including a subset of the samples from the classes with a large amount of instances, and upsampling refers to the process of duplicating a portion of datapoints randomly in the classes with a low volume of data. If the dataset is divided into train and test sets of some proportion, resampling must be performed separately on the divided sets. While resampling the dataset can be particularly useful when there is an abundance of labeled data available, in the case of smaller datasets, using k-fold cross-validation (explained in Section 4.4.2) during training is more beneficial.

3.4.2 Machine learning

While the detailed procedure can vary based on the prediction task and data available, supervised machine learning is a well-established process. The difference in developing tools for requirements engineering compared to other applications is that besides model testing, model validation is also necessary to assess the performance of the model in the deployment stage. Although in the case of a large dataset, it is possible to create a separate dataset and perform validation during the development phase, evaluating the efficiency of the deployed model is imperative. By doing so, developers can further tune the model if necessary to accommodate present trends and needs.

3.4.2.1 Proposed architecture to incorporate external data into the network

In order to realize the concept of incorporating external data directly into the learning process (Section 3.3.2.3), it is critical to build a network architecture that can learn from the external data source while still focusing on the embeddings to reduce the classification error of the target task. The high-level structure of the proposed architecture is shown in Figure 3.6. Unlike fully-connected neural networks, recurrent neural networks (RNN) excel in processing sequential information (explained in Section 4.4.4). Therefore, recurrent layers (RECL) are optimal for handling both the word embeddings and the encoded features from the external knowledge source. The output of the RNN layers are concatenated (CONCAT) to merge the two

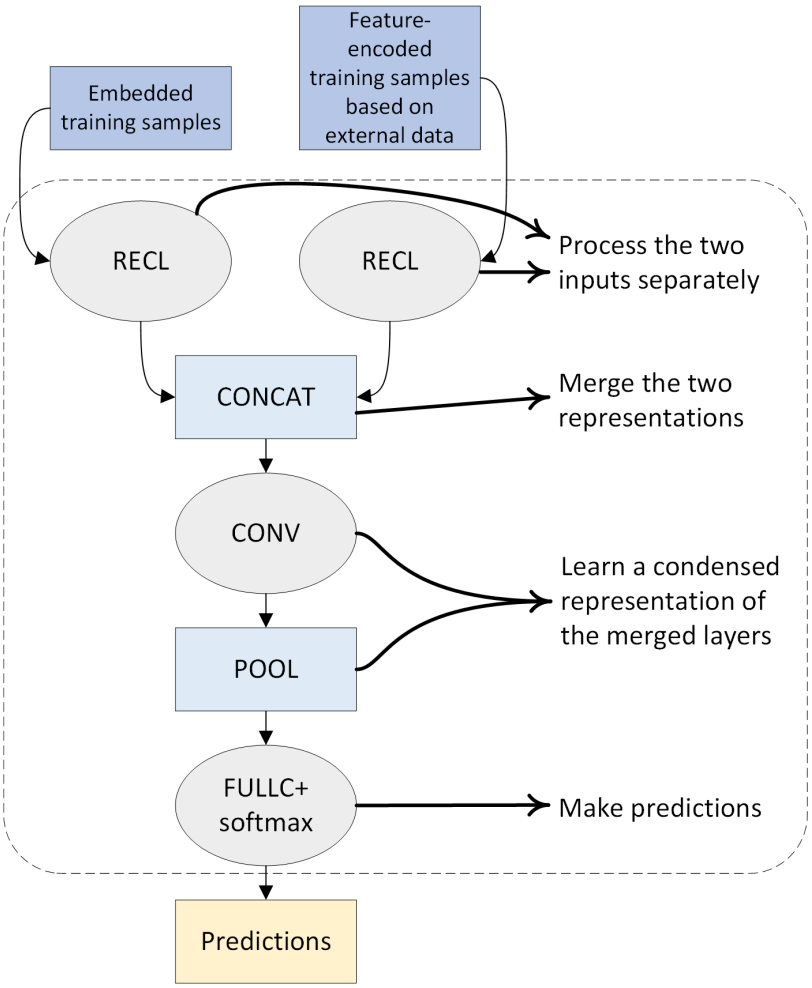

Figure 3.6: High-level structure of the proposed network architecture

representations. In order to discover local features in the merged layers regardless of their location, a 1-dimensional convolutional layer (CONV) is applied after the concatenation (explained in Section 4.4.5). To help reduce the spatial scale of the feature representations while keeping a reasonable amount of information on the classes, a pooling layer (POOL) is put after the convolutional layer. Finally, a fully-connected layer (FULLC) with softmax (explained in Section 4.4.6) is applied to make the predictions. It is to note that the exact architecture (e.g. type of the recurrent layers) and the complexity of the network (number of layers, neurons, convolutional filters, etc.) may vary based on the target task, choice of word embedding model, and the nature of the external knowledge source. Nevertheless, regardless of

future advancements in recurrent types of networks, word embeddings, etc., the above architecture would be generally applicable for classification tasks with textual inputs when there is a low volume of annotated data available. The main idea was proposed by the author in [116], where the above architecture was implemented using a LSTM+BiLSTM+CNN architecture with a sentiment lexicon of word polarities on a continuous scale.

3.4.2.2 Adjusting a pretrained language model to a different target task

As it was discussed in Section 2.3, when there is a discrepancy between the source and target domains or tasks, transfer learning might result in negative transfer, causing a decrease in machine learning performance. The proposed approach to use pretrained language models trained for a different task than the target classification problem is shown in Figure 3.7.

The original language model (e.g. BERT, ELMo, etc.) is pretrained on an extensively large collection of documents from the source domain $\mathcal{D}_S$ (e.g. Wikipedia, One Million Word Benchmark, BookCorpus, etc.) for the source task $\mathcal{T}_S$ (usually, next word/sentence prediction, to make $\mathcal{T}_S$ unsupervised). Since these models are available online for multiple languages, it is not necessary to train them from scratch. In order to avoid potential negative transfer, the pretrained language model is retrained or fine-tuned on documents from the target domain $\mathcal{D}_T$, depending on automatic pre-annotation is possible or not. Automatic pre-annotation is the process of assigning labels to a dataset without human intervention. Although automatic annotation methods are not accurate and hard to test their performance, if it is feasible to label large volumes of data in an unsupervised way, the produced pre-annotated dataset can be used to retrain a pretrained language model to a different task. Retraining, in this case, means that all layers of the pretrained model are involved in the training process to adjust the weight matrices to the target task $\mathcal{T}_T$. If automatic pre-annotation is not a feasible option for the target task, the documents for $\mathcal{D}_T$ are first fine-tuned for the source task $\mathcal{T}_S$, i.e. in the same manner pretraining was conducted. Fine-tuning means that some of the layers are frozen, and not all of them are involved in the training process (the exact number of frozen layers is task specific). The reason for fine-tuning is to avoid overfitting on the new dataset, which would mean losing the robustness of the original model. The last phase is the same for both cases, fine-tuning a small but human-annotated dataset for the target task $\mathcal{T}_T$. Since the retrained model's prediction task is the same or similar to the final target task, it is possible to fine-tune the retrained model without negative transfer, even on a small human-annotated dataset.

In the scenario when pre-annotation is not possible, the proposed procedure attempts to mitigate negative transfer by fine-tuning the model in two stages, first for $\mathcal{T}_S$, then for $\mathcal{T}_T$.

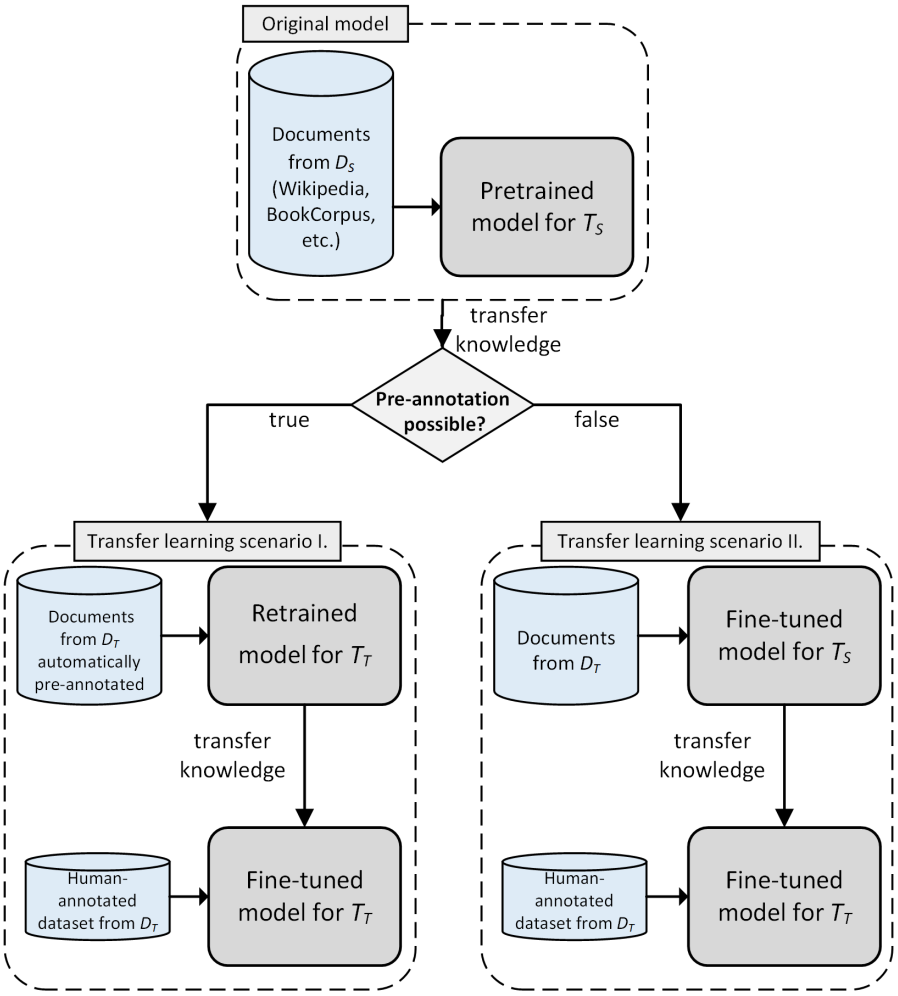

Figure 3.7: Proposed transfer learning strategy

Chapter 4

Case studies

This chapter describes the case studies undertaken within this research to demonstrate the effectiveness of the given framework. The two case studies cover the following parts of the proposed approach.

1. **Predicting sentence level review informativeness of search products** (Case study I.): Testing the applicability of the pertinence model in the context of the proposed framework, including the suggested methods to acquire domain and target population specific vectors and weights. Additionally, the feasibility of the proposed transfer learning strategy using an automatically pre-annotated dataset is also assessed.

2. **Expanding the feature space of deep neural networks for multi-class sentence level sentiment classification** (Case study II.): In order to investigate the usefulness of incorporating external data explicitly into deep learning models in an unbiased manner, this part of the framework has been tested independently. Since SA is widely used in service-related contexts, experiments were conducted using online hotel reviews.

4.1 Computing environment

The case studies were implemented in Python version 3.6 and 3.7, and the code was run on Ubuntu 16 (Xenial Xerus) and 18 (Bionic Beaver). The workstation used is equipped with an Intel Core i9-9920X 12 core/24 thread CPU with 128GB DDR4 RAM, and two Nvidia RTX 2080TI GPUs connected with NVLink for GPGPU computing. Essential Python libraries used in the implementation of the case studies are listed below:

- **pandas**: Pandas is a library for data manipulation and analysis, used to read in and manipulate data in numerical tables.

- **numpy**: NumPy is a module for scientific computing in Python, offering a large set of mathematical functions, supporting up to high-dimensional tensors.

- **scipy**: SciPy is a library offering modules for linear algebra and optimization for various subfields in information science and engineering.
- **matplotlib**: Matplotlib is a plotting library built upon NumPy, to create both static and interactive visualizations in Python.
- **sklearn**: Scikit-learn is a machine learning library, offering tools for predictive data analysis and algorithm validation, built upon NumPy, SciPy, and matplotlib.
- **nltk**: The Natural Language Toolkit is a collection of NLP tools (e.g. tokenization, lemmatization, parsing, etc.) for Python, mainly for the English language.
- **mecab**: MeCab is a Japanese text-segmentation library used mainly for tokenization, lemmatization, and Part of Speech tagging.
- **tensorflow**: TensorFlow is a machine learning library focusing on the implementation of various type of neural networks, including custom-designed architectures.
- **keras**: Keras works as an interface for TensorFlow, built to enable user-friendly and fast experimentation with deep learning systems.
- **multiprocessing**: Multiprocessing is a Python package for parallel computing, allowing one to run functions on multiple cores/threads.
- **numba**: Numba is a just-in-time (JIT) compiler that translates low-level Python code directly to optimized machine code, speeding up algorithms potentially to the speed of pure C code.
- **gc**: The Garbage Collector enables the user to release unreferenced memory manually. It is useful when working with large data structures to free up system memory in loops.
- **re**: The re library allows for regular expression pattern matching for various preprocessing and data transformation tasks.

4.2 Predicting sentence level review informativeness of search products

In this case study, the framework introduced in Chapter 3 is applied to develop a system that would reduce the information overload of eWOM by

assessing review informativeness on the sentence level. The system assigns pertinence scores (Section 3.2) to the review sentences without manual feature engineering, and uses the proposed transfer learning strategy (Section 3.4.2.2) on a small human-annotated dataset for informativeness prediction.

In the context of online reviews, the information needs of both customers and companies depend on the product type [23, 45, 117]. Nelson separated products into two main types: *search* and *experience* products [118]. In contemporary requirements engineering and customer behavior research, this is one of the most used and acknowledged classification scheme of product types. The reason for focusing on search products in this case study is the following. While the evaluation of experience products (e.g. movie DVDs, books) is usually based on subjective and emotional assessments, the value judgment of search products (e.g. digital cameras, PCs) should be based on verifiable facts and impersonal evaluations [83, 119], much more useful for product designers and other people involved in requirements engineering.

While aspects such as reviewer status, shipping time, return policy, etc. can be crucial for customers, these carry little significance for requirements engineering. Therefore, service and platform-related factors (e.g. packaging, shipping, website customer-service), as well as non-content-related factors (e.g. reviewer's user status, product rating in stars) will not be considered as valid features of review informativeness in the presented case study. Technical attributes, however, such as physical characteristics (e.g. display size) and other specifications (e.g. processing power) are considered as prominent features of review informativeness, regardless of whether expressed explicitly or implicitly in a review. For more information on the notion of informativeness, see Chapter 1 and Section 2.2.

4.2.1 System design

Figure 4.1 shows the overview of the proposed system, based on the framework proposed in Chapter 3. The system has two phases: development and operation. The development involves two modules: one for pertinence quantification and another for developing the prediction model. The operation phase simulates the potential deployment of the proposed system: takes product reviews (unseen during model training) as input, and labels the sentences in the reviews, according to their level of informativeness.

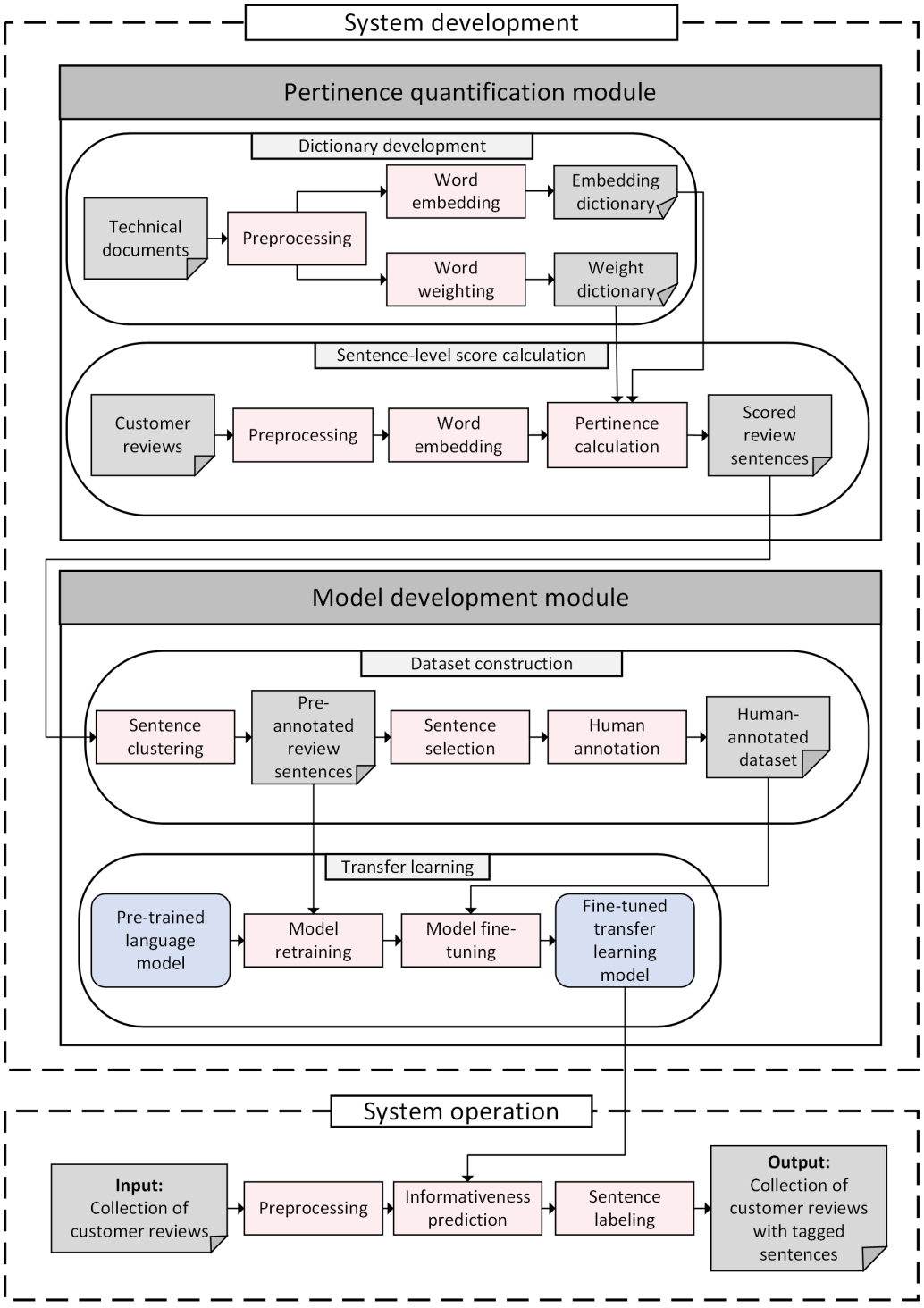

Figure 4.1: Overview of the system developed to predict search product review informativeness on the sentence level

4.2.1.1 Pertinence quantification module

The pertinence quantification module has two inputs: technical documents from domain V, and a large amount of customer review sentences. As it was described in Section 3.2.1, technical documents are presumed to represent the target population T well, when T = *product designers*. Therefore, after preprocessing the technical documents, all word lemmas are put into the embedding dictionary *emb_dic* with their corresponding contextualized, domain specific embeddings (Section 3.3.2.1) created using model EMB. The preprocessed technical documents are also used to assign scalar weights (Section 3.3.2.2) to all unique words, signifying their importance for V and T. The weights with their matching words are put into the weight dictionary *weight_dic*. The two dictionaries are used together with the preprocessed customer reviews to calculate pertinence scores for the review sentences.

Algorithm 3 Calculation of sentence-wise pertinence

procedure CALC SENTSCORES(reviews, EMB, *emb_matrix*, *emb_dic*, *weight_dic*)
  initialize array *sen_scores*
  for all sentence **in** reviews **do**
    $sum\_score = 0$
    for all word **in** sentence **do**
      $len = 0$
      if word **not in** stopwords **then**
        $len+ = 1$
        $wordvec = EMB$.**embed**(word)
        $distances$ = cosdis_computation($wordvec$, *emb_matrix*)
        $closest\_word$ = *emb_dic*[**argmin**($distances$)]
        $weight = weight\_dic[closest\_word]$
      end if
      $sum\_score+ = weight$
    end for
    $adj\_score = \frac{sum\_score}{len^{1/len}}$
    append *adj_score* **to** *sen_scores*
  end for
  return *sen_scores*
  for all *sen_score* **in** *sen_scores* **do**
    $sen\_score = 2\frac{sen\_score - \mathbf{min}(sen\_scores)}{\mathbf{max}(sen\_scores) - \mathbf{min}(sen\_scores)} - 1$
  end for
end procedure

The algorithm introduced for pertinence calculation in Section 3.3.3 (Algorithm 2) is readjusted in the following manner for the given case study:

1. Since in this case study, informativeness prediction is performed on the sentence level, the procedure returns pertinence scores of sentences instead of whole reviews, regardless of their document origin (i.e. the same sentence has the same score in all documents).

2. The algorithm requires computing cosine distances between every word in the review sentences and each word in the embedding dictionary *emb_dic*. Because this is a computationally expensive process for a large number of reviews, the embeddings from *emb_dic* are combined to a $H \times dim$ matrix *emb_matrix*, where H is the total number of entries in the embedding dictionary, and *dim* is the dimensionality of the individual embedding vectors. The cosine distance calculation between *emb_matrix* and a word vector from the review sentences *wordvec* is parallelized, speeding up the process significantly.

Algorithm 4 Fast calculation of cosine distances

```
1: procedure COSDIS_COMPUTATION(vector u, matrix M)
2:     M_len = length(M.rows)
3:     u_len = length(u)
4:     initialize array distances of length M_len populated with zeros
5:     for i in range(M_len) do in parallel
6:         v=M[i]
7:         u_dot_v = 0
8:         u_norm_t = 0
9:         v_norm_t = 0
10:        for j in range (u_len) do
11:            u_dot_v += u[j] * v[j]
12:            u_norm_t += u[j] * u[j]
13:            v_norm_t += v[j] * v[j]
14:        end for
15:        u_norm = sqrt(u_norm_t)
16:        v_norm = sqrt(v_norm_t)
17:        theta = u_dot_v/(u_norm * v_norm)
18:        distances[i] = 1 - theta
19:    end for
20:    return distances
21: end procedure
```

The final pertinence calculation procedure for the case study is shown in Algorithm 3. The embedding model EMB used in this study is BERT-Base, introduced in Section 2.3 The algorithm for computing cosine distances (cosdis_ccomputation) between the embedding matrix and the observed word's vector is specified by Algorithm 4. The function returns an array of cosine distances *distances* between *emb_matrix* (matrix M) and the observed word's vector *wordvec* (vector u).

4.2.1.2 Model development module

To be able to define a classification problem of informativeness prediction, the scored review sentences are clustered according to their pertinence scores, by k-means. The number of clusters is determined with the elbow method [120], that assumes that model performance will inevitably decline as k increases. This process of clustering based on pertinence scores can be considered as automatic pre-annotation. Next, a small number of sentences are selected from each class (cluster), and re-annotated by humans to build a small but representative dataset.

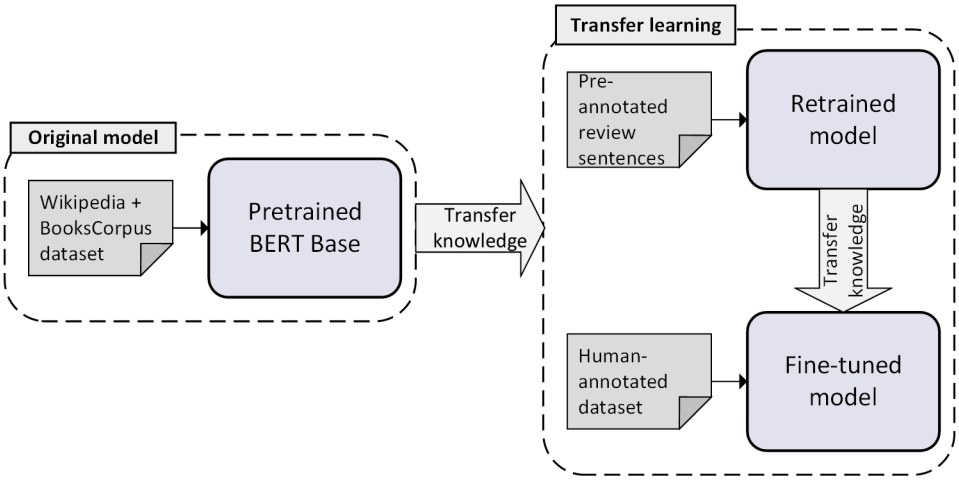

Figure 4.2: The transfer learning process

The transfer learning strategy suggested in Section 3.4.2.2 is applied to the pre-annotated sentences and the human-annotated dataset, to adjust a pretrained language model to the target classification task $\mathcal{T}_T$ (informativeness prediction). The transfer learning process used in the proposed system is depicted in Figure 4.2. The original model, BERT-Base is pretrained on the BooksCorpus (∼800 million words) [121] and Wikipedia (∼2,500 million

words). In order to avoid negative transfer, BERT is first retrained on the automatically pre-annotated but huge dataset. To specialize the model to the task of informativeness prediction without overfitting, only the last 6 layers plus the classification layer are fine-tuned on the human-annotated dataset, and the weight matrices of the first half of the 12 encoder layers are frozen.

4.2.2 Data

The domain V chosen for the experiments is *Digital Cameras and Accessories*. The technical documents used to create the term dictionary are articles from Wikipedia on digital camera photography. There is a vast amount of useful technical information freely accessible on Wikipedia, and all articles can be downloaded through Wikimedia dumps[1]. 1,039 articles related to digital photography equipment and technical jargon have been collected using Wikipedia metadata. Unneeded sections, such as "References" and "See also" were discarded during preprocessing, resulting in a total of 24,134 sentences. 13,166 unique words lemmas were vectorized and weighted, then inserted into the Embedding and Weight dictionaries. The pretrained BERT-Base language model was retrieved from Google Research[2].

Customer reviews used in this research come from the "Electronics" part of Amazon Review Data [122] (ranging from May 1996 to July 2014). An example of an Amazon review on the website is shown at Figure 4.3.

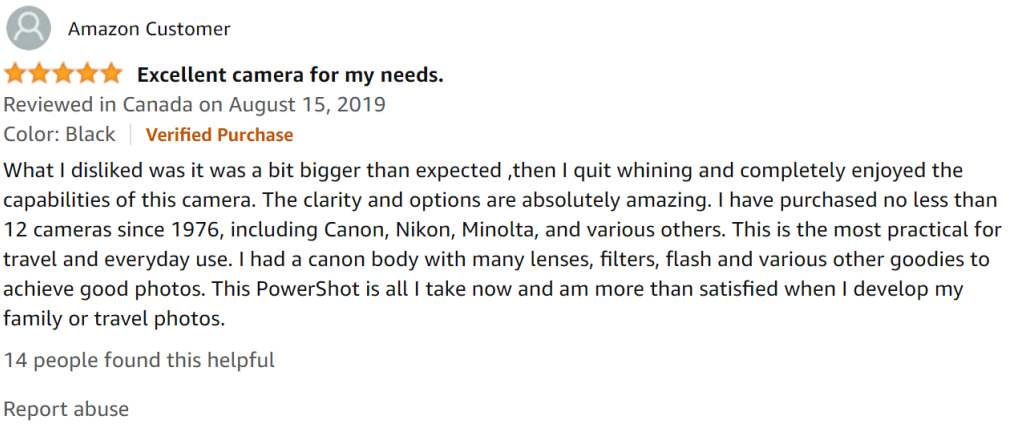

Figure 4.3: An example of a digital camera review from Amazon

[1]https:/dumps.wikimedia.org/
[2]https:/github.com/google-research/bert

	asin	description	categories	title
0	0132793040	The Kelby Training DVD Mastering Blend Modes i...	ElectronicsComputers & AccessoriesCables & Acc...	Kelby Training DVD: Mastering Blend Modes in A...
1	0321732944	NaN	ElectronicsComputers & AccessoriesCables & Acc...	Kelby Training DVD: Adobe Photoshop CS5 Crash ...
2	0439886341	Digital Organizer and Messenger	ElectronicsComputers & AccessoriesPDAs, Handhe...	Digital Organizer and Messenger
3	0511189877	The CLIKR-5 UR5U-8780L remote control is desig...	ElectronicsAccessories & SuppliesAudio & Video...	CLIKR-5 Time Warner Cable Remote Control UR5U-...
4	0528881469	Like its award-winning predecessor, the Intell...	ElectronicsGPS & NavigationVehicle GPSTrucking...	Rand McNally 528881469 7-inch Intelliroute TND...
...	...	...	...	...
498191	BT008V9J9U	Vehicle suction cup mount (replacement) NOTICE...	ElectronicsGPS & NavigationGPS System Accessor...	Suction Cup Mount
498192	BT008SXQ4C	Quatech - 1 Port PCMCIA to DB-25 Parallel Adap...	ElectronicsComputers & AccessoriesCables & Acc...	Parallel PCMCIA Card 1PORT Epp
498193	BT008G3W52	C2G - 5m Ultma USB 2.0 A Mini B Cble	ElectronicsComputers & AccessoriesCables & Acc...	C2G / Cables to Go 5M Ultima USB 2.0 Cable
498194	BT008UKTMW	Keyboard drawer.	ElectronicsComputers & AccessoriesCables & Acc...	Underdesk Keyboard Drawer
498195	BT008T2BGK	Garmin USB to R232 Converter CableUSB to RS232...	ElectronicsComputers & AccessoriesCables & Acc...	USB To R232 Converter Cable

498196 rows × 4 columns

Figure 4.4: Amazon review metadata

Algorithm 5 Extract relevant product IDs

```
procedure EXTRACT IDS(metadata_t)
    initialize list asins
    initialize list brands of camera brand names
    initialize list desc of strings "camera", "digital", "battery", "lens"
    for all row in metadata_t do
        if string "camera" not in row.categories then
            if any in brands not in row.title then
                if any in desc not in row.description then
                    delete row
                end if
            end if
        end if
    end for
    for all id in metadata_t.asin do
        append id to asins
    end for
    return asins
end procedure
```

In order to extract reviews connected to digital camera and a accessories (such as lenses, flashes, etc.), relevant product IDs had to be acquired from

the review metadata. Figure 4.4 shows the metadata after removing unneeded information. Relevant product IDs (*asin* in the metadata) were obtained using the procedure described by Algorithm 5. Using the extracted IDs, 300,170 reviews were retrieved from the Electronics review dataset. Sentences with 35+ tokens were presumed to be from reviews without punctuation, therefore, these were excluded from further processing steps. This resulted in a total of 1,160,601 unique sentences collected for the pertinence quantification module.

4.2.3 Experiments

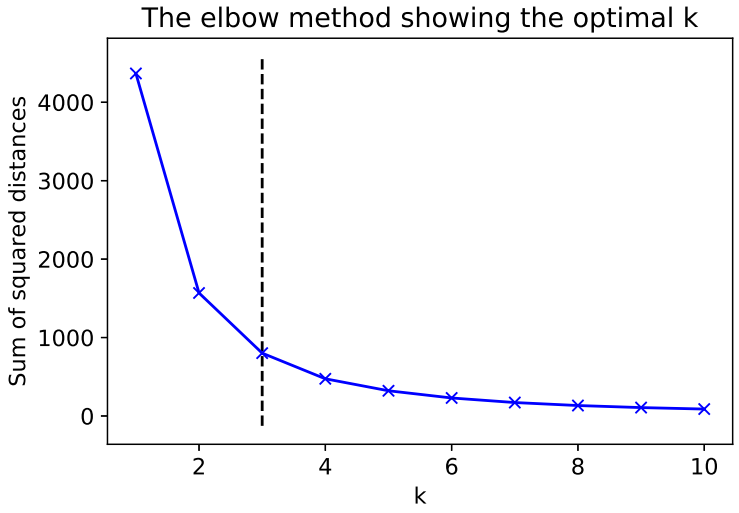

Figure 4.5: The elbow point is detected at $k = 3$

The issue of deciding on the number of k-means clusters was addressed by applying the Kneedle algorithm [123], to detect the elbow point of the sum of squared distances measured for various k. The elbow point was defined at $k = 3$, as seen in Figure 4.5.

The resulting distribution of the sentences is depicted in Figure 4.6. While Class 1 includes sentences with a low pertinence, Class 3 sentences are supposed to be highly relevant for the given V, and T. Since the uneven class distribution could lead to decreased training performance and results would be biased, retraining BERT-Base required up-sampling Class 3 and down-sampling Class 1 for the number of instances in Class 2, after dividing the data into 70% training and 30% test sets (details on data debiasing are described in Section 3.4.1).

1000 sentences were drawn from each class to build a small dataset of 3000 sentences for human annotation. While the sentences were originally chosen at random, manual intervention was necessary to create a representative dataset, as the pre-annotated dataset is noisy. To obtain a large dataset, the review sentences from Amazon data were extracted using brand names and keywords in product descriptions, etc. (Algorithm 5). Due to this approach, however, the data would sometimes contain reviews that are not directly related to digital cameras (e.g. printers). These irrelevant reviews have not been selected into the final dataset for annotation.

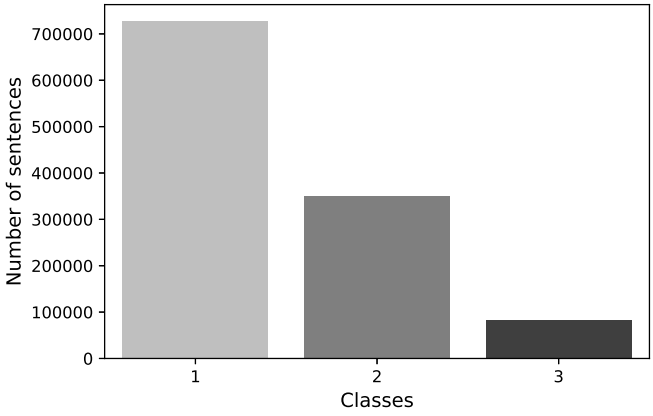

Figure 4.6: Distribution of the review sentences after automatic pre-annotation

Since the majority of sentences allocated to Class 1 are short (for example, "Price is fine."), and sentences classified as Class 3 are often lengthy, selecting sentences entirely at random would result in a biased dataset, and classification results may not be representative. To cope with this issue, sentences with varying lengths have been chosen from each group to create an unbiased dataset. Although the natural correlation between sentence length and class allocation is still reflected in the mean number of words per sentence (Class 1: 16.1, Class 2: 17.69, Class 3: 22.97), the class-wise sentence length variances are relatively high for all three classes (Class 1: 23.55, Class 2: 38.46, Class 3: 29.64). The violin plot of class-wise sentence length is depicted in Figure 4.7. The class distributions illustrated are the kernel density estimations of the real distributions, the white dots show the median values, and the black rectangular box marks the interquartile range.

Five individuals have participated in the annotation process. After training the annotators, they were instructed to tag sentences based on their informativeness, into three possible categories: "Not informative" (Class 1),

"Moderately informative" (Class 2), and "Definitely informative" (Class 3).

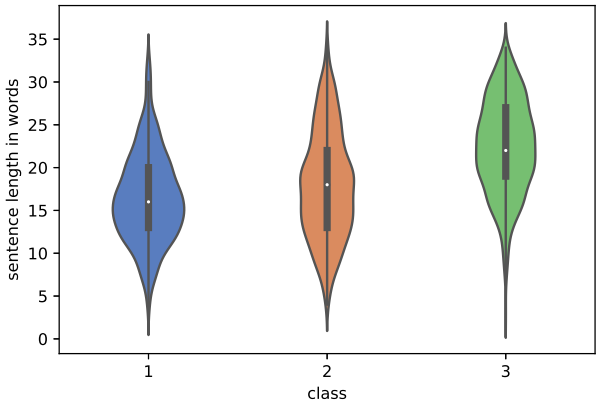

Figure 4.7: Violin plot of class-wise sentence length of the dataset created for annotation

Each sentence has been labeled by three participants. Single labels were determined by majority voting (when all annotators tagged the sentence differently, a random choice was made). The inter-annotator agreement was calculated using Krippendorff's alpha [124] (explained in Section 4.4.3). The obtained alpha coefficient is 0.75, indicating a reliable annotation. The class distribution of the annotated sentences is seen in Figure 4.8. The dataset created is publicly available for academic purposes at Elsevier Mendeley Data [125]. In order to prevent overfitting on the relatively small dataset, BERT was retrained for only 5 epochs, and fine tuning was performed for 10 epochs.

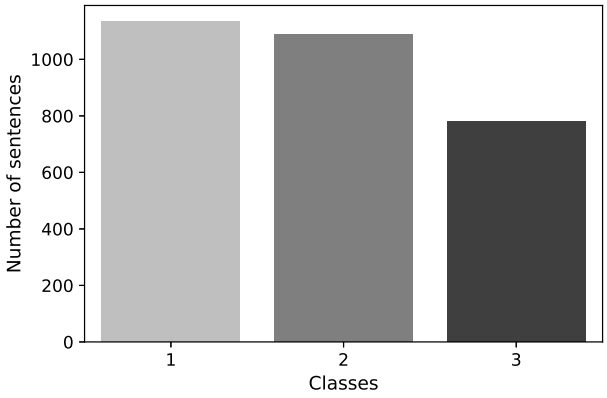

Figure 4.8: Distribution of the human-annotated dataset

To transform text into the format BERT requires, the procedure described by Algorithm 6 had to be performed three times (for the embedding dictionary, pertinence quantification, and transfer learning). The procedure returns the input IDs *glob_input_ids*, masks *glob_input_masks*, and segment IDs *glob_segment_ids* corresponding to the raw input sentences *examples*. Details of the input format for BERT are described in Section 2.3.

Algorithm 6 Data transformation required for BERT

```
1: procedure PREPARE INPUTS(examples, max_len)
2:     initialize lists glob_input_ids, glob_input_masks, glob_segment_ids
3:     for all example in examples do
4:         tokens = tokenize(example)
5:         if length(tokens) > max_len then
6:             tokens = tokens[0:max_len]
7:         end if
8:         initialize list tokens_form
9:         initialize list segment_ids
10:        append string "[CLS]" to tokens_form
11:        append int 0 to segment_ids
12:        for all token in tokens do
13:            append token to tokens_form
14:            append int 0 to segment_ids
15:        end for
16:        append string "[SEP]" to tokens_form
17:        append int 0 to segment_ids
18:        input_ids = transform tokens_form to ids
19:        in_len = length(input_ids)
20:        initialize list input_masks of length in_len populated with 1s
21:        while in_len > max_len do
22:            append int 0 to segment_ids
23:            append int 0 to input_masks
24:            append int 0 to input_ids
25:        end while
26:        append input_ids to glob_input_ids
27:        append input_masks to glob_input_masks
28:        append segment_ids to glob_segment_ids
29:    end for
30:    return glob_input_ids, glob_input_masks, glob_segment_ids
31: end procedure
```

In order to evaluate the efficiency of the fine-tuned model, four other models were developed for comparison. Details of the models are presented in Table 4.1.

Table 4.1: The models built to analyze the performance of the proposed model (model E).

Name	*Model*	*Features*	*Training dataset*
model A	Multi-class SVM	tf-idf vectors	annotated data
model B	retrained BERT-Base	embedding vectors	pre-annotated sentences
model C	BERT-Base frozen (+ classification layer)	embedding vectors	annotated data
model D	fine-tuned BERT-Base	embedding vectors	annotated data
model E (proposed model)	retrained + fine-tuned BERT-Base	embedding vectors	pre-annotated sentences + annotated data

Model A is a one-vs-one multi-class radial basis kernel support vector machine (SVM) classifier. For feature encoding, the model uses word frequency-inverse document frequency (tf-idf) vectors. Model A was built to decide if the proposed model's performance is better than a baseline, conventional machine learning classifier. Model B is BERT-Base retrained on the pre-annotated sentences, but without fine-tuning it on the human-annotated dataset. For Model C, BERT-Base is implemented with an additional layer for classification without applying transfer learning on the pre-annotated sentences. In this model, all encoder weights are fixed and untrainable to investigate the performance of the original BERT-Base model for the target classification task. Fine-tuning was performed on the last 6 of the 12 layers of BERT-Base for Model D, i.e. retraining BERT on the pre-annotated sentences was not implemented, but the model was fine-tuned. Model E is the proposed model introduced in Section 4.2.1.2. In the cases of models A, C, and D, training and testing were implemented by performing 5-fold stratified cross-validation on the annotated dataset. Although training for Model B means only retraining BERT on the pre-annotated sentences, testing is performed on the same folds of the human-annotated dataset as the other models, in order to achieve comparable results. For Model E, retraining is executed on the pre-annotated sentences, and fine-tuning and testing are

carried out using 5-fold stratified cross-validation on the annotated dataset. Besides comparing the accuracy, precision, recall, and $F1$ scores of the models, the significance of difference was calculated for all models vs. model E (proposed model). In order to determine whether the performance of the models are actually different, the testing accuracy scores of the stratified cross-validation folds were used to perform t-tests. The null hypothesis of the paired student t-test states that there is no statistically significant difference between model performances. On the other hand, the alternative hypothesis is that the models' performance is truly different. Although the t-test for comparing machine learning models is considered to have a low Type II error (unlike, for example, the 5x2 cv test [126] and the McNemar test [127]), it is susceptible to Type I error [126, 128]. The explanation behind this is that the t-test assumes independent sampling, however by definition, the folds of cross-validation are overlapping through the different folds of training and test sets. According to Nadeau & Bengio [128], the violation of the assumption of independence is related to the underestimation of the variance of differences which results in a higher chance of Type I error. The authors demonstrated that computation of variance s^2 can be modified to consider the dependency and calculate an adjusted variance s^2_{corr} as follows:

$$s^2_{corr} = s^2\Big(\frac{1}{k} + \frac{n_2}{n_1}\Big), \tag{4.1}$$

where k is the number of cross-validation folds, n_1 is the number of samples used for training, and n_2 is the total number of data points used for testing.

4.2.4 Results and Discussion

Table 4.2 lists the accuracy, precision, recall, and F_1 scores obtained (class-wise metrics are averaged). The two-tailed p-values for the t-statistics are listed in the last row, suggesting that there are significant differences between models A-D and model E. The null hypothesis, therefore, was rejected in all four cases.

Table 4.2: Performance comparison of the five models.

Performance metric	**model A**	**model B**	**model C**	**model D**	**model E**
accuracy	70.02%	72.20%	64.42%	69.30%	79.39%
precision	0.70	0.72	0.64	0.72	0.81
recall	0.69	0.74	0.63	0.68	0.79
F_1 *score*	0.69	0.73	0.63	0.70	0.80
statistical significance	p<0.005	p<0.005	p<0.001	p<0.001	-

Without retraining BERT, and just adding a classification layer after the stacked encoder structure (model C) resulted in the poorest results of 64% accuracy averaged over all folds. Although fine-tuning BERT without retraining (model D) improved the accuracy by 5%, both models struggled to match the baseline SVM's (model A) accuracy. This suggests that the weight matrices of BERT must be retrained first to obtain better results. 72% accuracy was achieved by retraining all layers of BERT on the pre-annotated sentences (model B). The second-best accuracy was obtained by model B, even without fine-tuning the retrained BERT. This shows that the pertinence quantification module functions adequately as an automated pre-annotation tool for predicting informativeness. For the retraining phase itself, an accuracy of 87% was achieved, showing that the weight matrices of BERT have been successfully adjusted for the target task. Class-wise performance scores for the retraining process is shown in Table 4.3. Fine-tuning BERT on the human-annotated dataset after retraining resulted in the highest accuracy of 79% and higher overall performance metrics (model E).

Table 4.3: Class-wise performance scores for the retrained model.

	precision	*recall*	F_1
class 1	0.91	0.88	0.89
class 2	0.86	0.86	0.86
class 3	0.8	0.92	0.86

The class-wise F_1 scores for the models, averaged over all the cross-validation folds, are shown in Figure 4.9.

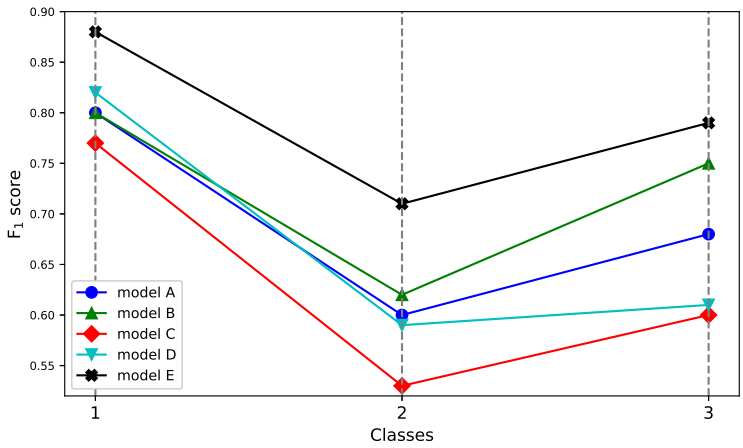

Figure 4.9: Class-wise F_1 scores for all models

The same tendency can be observed for all models. While each model has shown acceptable results for class 1, the most difficult to predict and recall is class 2. As in other classification tasks in text analysis (e.g. often in multi-class SA or emotion recognition), classes in this case study are not entirely distinct categories. The annotated labels of the following sentences, for instance, are all class 3 ("Definitely informative"):

1. *"Wide, deep images with pinpoint depth of field focus and low light beauty."*
2. *"Due to the D7000 more sensitive iso capabilities with more megapixels compacted in a smaller sensor this lens clearly needs VR II when shooting at F2.8 in low light."*
3. *"Only issue is that you may develop some edge distortion depending on the focal range and setting, and at 52mm it does suffer a bit in low light conditions."*

Although all phrases are informative and address multiple product features, the second and third sentence is much more detailed and includes more technical content. The following sentences are all Class 2 ("Moderately informative"):

1. *"Poorly designed cable on end that plugs into camera."*
2. *"It struggles to focus even in the simple center focus mode."*
3. *"It's done very well for daytime shoots of wildlife and I've gotten many national geographic type photos with it."*

In the first two sentences, the customers described issues regarding the camera accessory cable and the autofocus. While being short sentences, these still contain potentially useful pieces of information. The third sentence is more lengthy, and while it is unclear what is a "national geographic type photo", the customer mentions that the product worked adequately for daylight wildlife photography. The following sentences belong to class 1 ("Not informative"):

1. *"We got this camera as a Christmas present for my daughter."*
2. *"And by virtue of using this lens to its full benefit, I am learning even more on how shutter speeds, apertures and ISO all work together for maximum benefit."*
3. *"I bought it for my trip to Beunos Aires, and also used it at the Iguazu Falls, and could not have asked for more perfect performance!!!"*

According to the definition, none of these sentences are informative, but the rationale behind why a given sentence belongs to a category varies. The first sentence refers to irrelevant information, probably as an opening phrase of the review. In the second sentence, although there are multiple product features and it is relatively long, the customer merely states that she/he is learning about them. The third sentence is also relatively lengthy, but the customer fails to mention the reason why the camera has "perfect performance". The remainder of these reviews may be useful, but the above sentences do not offer any valuable information on the products. The above examples demonstrate why both annotation and classification are difficult, particularly in the case of Class 2, which is a category between "Not informative" and "Definitely informative".

Figure 4.10 shows two examples of reviews (not included in the original dataset) with the system output of tagged sentences. As the sentence labels show, the system is capable of tagging new sentences based on their level of informativeness effectively. The sentences labeled class 1 are either irrelevant and missing factual information e.g. "The price point on Amazon for the Nikon D750 was right for me and so I went for it.", or formulated as an introduction/conclusion for the review, for example, "All bets are off, Nikon has done an insane job creating an amazing camera body!". While sentences from class 2 contain useful pieces of information, e.g. "Besides this it has excellent video capabilities with a Tilt LCD screen.", class 3 sentences are detailed and involve more technical details, for instance, "Even at 12,800 I'm able to get perfectly usable images where the D600 at 2000 ISO was debatable."

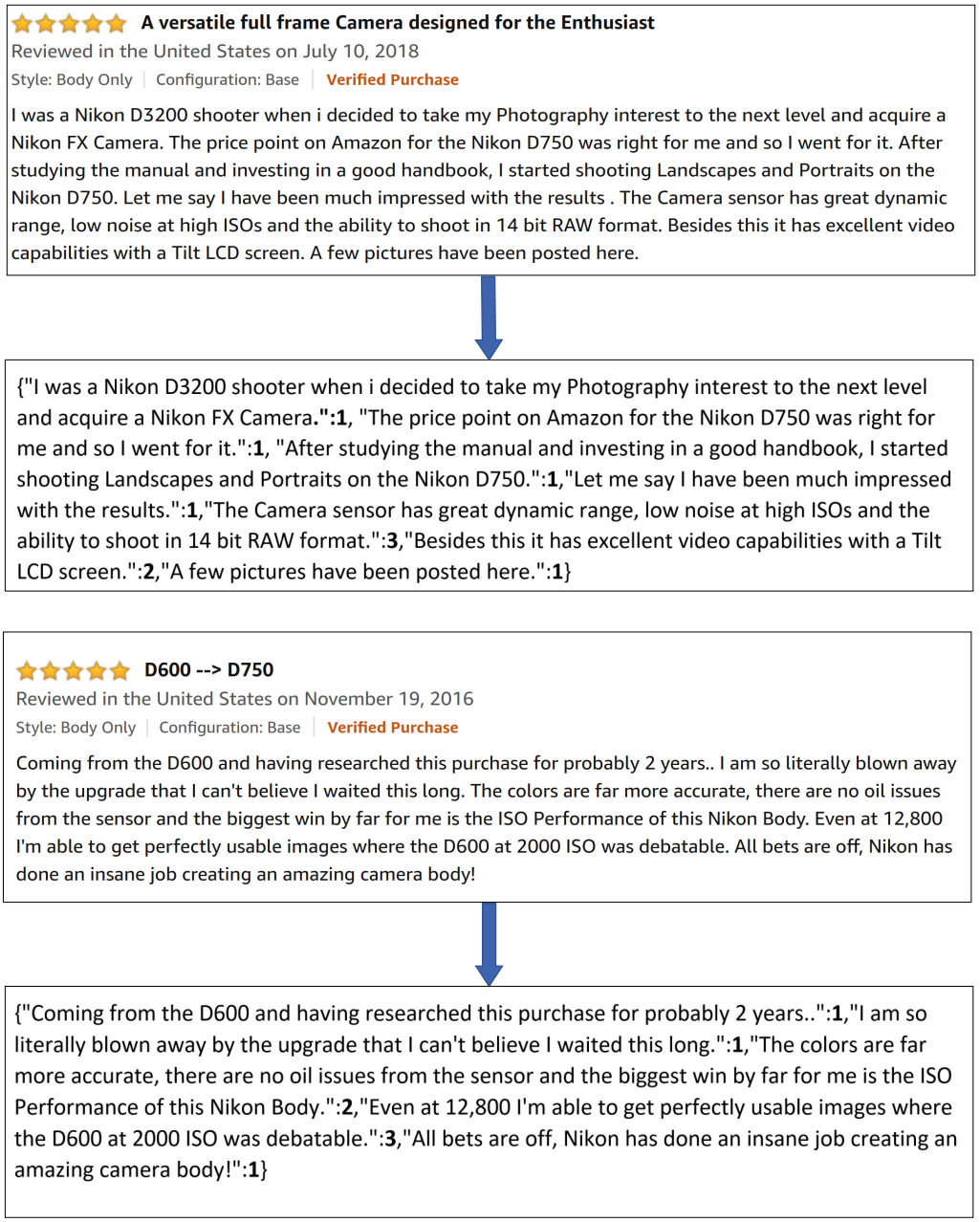

Figure 4.10: Examples of input reviews with the corresponding outputs (class labels are shown in bold numbers)

4.3 Expanding the feature space of deep neural networks for multi-class sentence level sentiment classification

In this case study, the proposed approach to incorporate external information explicitly into deep learning models (Sections 3.3.2.3 and 3.4.2.1.) is implemented, and the usefulness of using it with a limited amount of customer review data is investigated. The case study presents a neural network architecture that integrates semantic information from a sentiment lexicon to enhance prediction performance of multi-class sentence level sentiment classification of online hotel reviews.

4.3.1 Network architecture

The network architecture is based on the concept of using two feature spaces as inputs for the neural network: $\mathcal{X}$ and $\mathcal{X}'$, as proposed in Section 3.3.2.3. One represents word embeddings, and the other is the feature-encoded training data based on external knowledge, that is a sentiment lexicon in this case study.

The proposed network architecture is illustrated in Figure 4.11. There are two types of network inputs: word embedding vectors $\mathbf{e}$ and sentiment scores q from a polarity dictionary. The sentiment scores are inputted into a Long-Short Term Memory (LSTM) layer to learn sequential relationships among the words based on their sentiment polarity. The word embedding vectors are fed into a Bidirectional LSTM (BiLSTM) [129] layer to process words from both directions and get a more comprehensive representation for deeper layers. The hidden states from the BiLSTM are concatenated (CONCAT) with the output of the LSTM layers to create a combined representation of the sentence, where both feature spaces are integrated into the network. The merged outputs are then used as inputs for a one-dimensional convolutional layer (CONV) so that the network can discover local features in sentences regardless of their location. Max pooling (MAX POOL) and the average pooling (AVG POOL) are used after the convolutional layer in order to minimize the spatial scale of the learned matrices. This helps the network to reduce the number of features yet to maintain a reasonable amount of information on the classes. Finally, the concatenated outputs after pooling are used as inputs to a fully-connected dense layer (FULLC) with a softmax activation function.

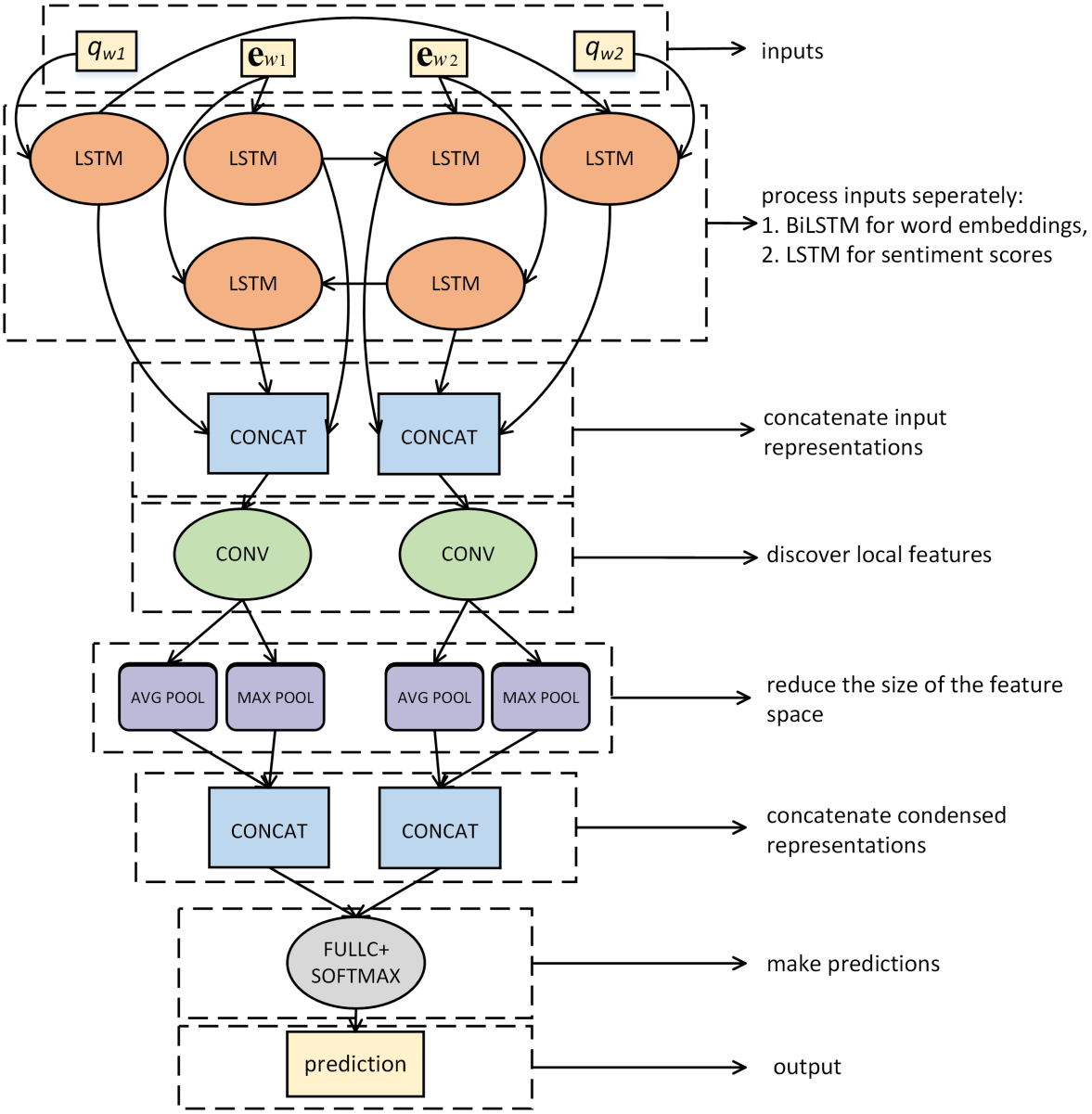

Figure 4.11: The proposed network architecture

4.3.2 Data

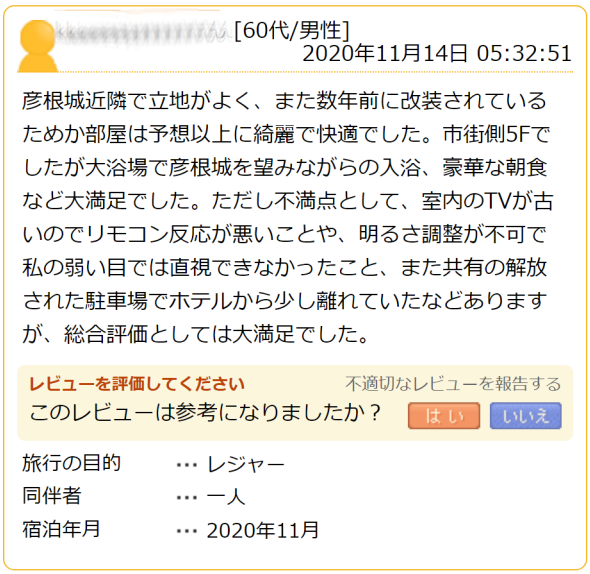

Figure 4.12: An example of a hotel review from Rakuten Travel

The dataset used in the research is the Tsukuba sentiment-tagged corpus, constructed by the Tsukuba University's "NLP on the Web" Laboratory. The data is distributed by the National Institute of Informatics[3], Japan. The data collection includes 4309 Japanese hotel review sentences labeled with sentiments from Rakuten Travel[4]. Rakuten Travel is a web-based hotel reservation site where customers can also share their opinions and recommendations. A review example is shown in Figure 4.12.

The annotation was carried out by two human subjects, and the labeling method was as follows:

- There are five categories: *praise*「褒め」, *neutral*「ニュートラル」, *complaint*「苦情」, *request*「要求」, and *no sentiment*「評価なし」.
- If there are multiple emotions in the same sentence, one annotator used a single label, and the second annotator applied plural labels.
- Sentences that are not tagged at all are marked as *other/pending*「その他/保留」.

[3]https://www.nii.ac.jp/en/
[4]https:/travel.rakuten.co.jp

Since sentences from category *other/pending*「その他/保留」are practically useless, these have been excluded from the data, resulting in 4219 sentences. When annotator 2 used multiple labels for a particular sentence, the label from annotator 1 was favored. In the case annotator 1 and annotator 2 used different singular labels, the final label was decided randomly. The number of samples for the categories are shown in Table 4.4.

Table 4.4: Number of class-wise sentences in the final dataset.

praise	**complaint**	**request**	**neutral**	**no sentiment**
1846	827	201	280	1065

The sentiment lexicon used in this study was created by Takamura et al. [130], and it includes 55,125 Japanese words with semantic orientations automatically assigned on a continuous scale. The reason for choosing this particular sentiment lexicon for experiments is that the POS considered are not just only adjectives (形容詞) and adverbs (副詞), but also verbs (動詞) and nouns (名詞). The word scores are between -1 and 1, where -1 indicates an entirely negative word, and 1 signifies a fully positive word. Below there are a few example entries from the lexicon (English translations in parenthesis are assigned by the author and are not part of the lexicon):

- 苦しめる (to inflict pain): くるしめる: 動詞: -0.999484
- 汚い (dirty):きたない: 形容詞: -0.999332
- 冷淡 (heartless): れいたん: 名詞: -0.721181
- 無論 (certainly): むろん: 副詞 :-0.133712
- 付合 (association): つけあい: 名詞: 0.260877
- 美人 (beautiful woman): びじん: 名詞: 0.358014
- 偉い (admirable): えらい: 形容詞: 0.937503
- 喜ぶ (to be glad): よろこぶ: 動詞: 0.999979

As the above examples show, the noun category also include words that are usually used as verbs (サ変接続名詞), and adjectival nouns (形容動詞).

4.3.3 Experiments

The word embedding model chosen for experiments is GloVe [112] (introduced in Section 2.3), to investigate the effectiveness of the proposed approach with a baseline, popular word embedding model that can be trained easily even for low-resource languages. The GloVe model was pretrained on the Japanese Wikipedia dump data[5] to generate 200-dimensional vectors. The output of pretraining is a text file where each line contains a word and the corresponding word vector. For example,

- 人工 0.470812 -0.678248 -0.203591 ... -0.572411 0.140961.

To be able to use it in the proposed deep learning model, an embedding matrix *embed_matrix* had to be created, that essentially works as a lookup table for locating word vectors. The full procedure is described by Algorithm 7.

Algorithm 7 Preparing embedding matrix

```
1: procedure PREPARE WORD VECTORS(glove_out, idx_dic)
2:     initialize dictionary glove_dic
3:     for all entry in glove_out do
4:         word = entry[0]
5:         vector = entry[1:]
6:         glove_dic[word] = vector
7:     end for
8:     initialize array embed_matrix of length(idx_dic) × 200
9:     for all word, idx in idx_dic do
10:        embedding = glove_dic[word]
11:        if embedding != None then
12:            embed_matrix[idx] = embedding
13:        else
14:            populate embed_matrix[idx] with zeros
15:        end if
16:    end for
17:    return embed_matrix
18: end procedure
```

First, word and vector pairs are put into a dictionary *glove_dic* from the pretrained model output *glove_out*. The embedding matrix is created using *idx_dic*, a dictionary of words of the review sentences with indexes assigned

[5] https://archive.org/details/jawiki-20180301

to them, and *glove_dic*. If the encountered word from the review is out of vocabulary (OOV), a zero vector is put the *embed_matrix*. The embedding matrix represents feature space $\mathcal{X}$.

The sentiment lexicon *sent_lex* is put into a dictionary *pol_dic*, and it is used together with *idx_dic* to create the sentiment matrix *sent_matrix* that is needed to retrieve polarity scores for the words from the review sentences. The procedure of building *sent_matrix* is given by Algorithm 8.

Algorithm 8 Preparing sentiment matrix

```
1: procedure PREPARE SENTIMENT SCORES(sent_lex, idx_dic)
2:     initialize dictionary pol_dic
3:     for all entry in sent_lex do
4:         word = entry[0]
5:         pol = entry[4]
6:         pol_dic[word] = pol
7:     end for
8:     initialize array sent_matrix of length(idx_dic) × 1
9:     for all word, idx in idx_dic do
10:        score = pol_dic[word]
11:        lemma_score = pol_dic[lemmatize(word)]
12:        if score != None then
13:            sent_matrix[idx] = score
14:        else if lemma_score != None then
15:            sent_matrix[idx] = score
16:        else
17:            sent_matrix[idx] = 0
18:        end if
19:    end for
20:    return sent_matrix
21: end procedure
```

In the case of OOV words, the sentiment score of its lemmatized form is used. If the word lemma is still OOV, the score 0 is assigned to the word (for the sentiment lexicon used, 0 indicates neutrality). The sentiment matrix obtained represents feature space $\mathcal{X}'$.

The efficiency of the proposed approach was evaluated by comparing its performance with two other models. In order to get a baseline accuracy, a vanilla neural network with one hidden layer is used. The proposed BiLSTM+CNN model was trained with and without incorporating knowledge from the polarity dictionary, to assess the effect of integrating external

linguistic information into the network. To avoid overfitting, dropout was applied after the recurrent layers with the probability of 0.3. 64 filters with the size of 3 were used for the convolutional layer, and 128-sample batch size was chosen for network training. To avoid overfitting on the training samples with the proposed architecture, L_2 regularization was applied to the convolutional layer. Owing to the limited size of the dataset and the unequal distribution of class samples, 10-fold cross-validation was implemented to train and validate all models.

4.3.4 Results and Discussion

Each model were trained for 10 epochs, and validation accuracy was recorded for all folds of cross-validation. Final accuracies were determined by taking the mean of the accuracy scores over the 10 folds. The baseline neural network with one hidden layer achieved a test accuracy of 67.02%. The BiLSTM+CNN model resulted in a significantly higher accuracy compared to the baseline, achieving 75.62%. The highest performance, however, was obtained by using the polarity scores from the sentiment dictionary, resulting in a 80.4% accuracy, outperforming the two other models by 13.38% and 4.78%. This indicates that sentiments that only capture polarities can help classifying sentiments of higher levels of abstraction, and have a complimentary function to word embeddings when the available dataset is small.

Table 4.5: Class-wise performance measures for the proposed architecture.

Performance measure	*Precision*	*Recall*	F_1
praise	0.874	0.89	0.882
complaint	0.766	0.773	0.769
request	0.758	0.63	0.688
neutral	0.447	0.426	0.436
no sentiment	0.819	0.814	0.8166

The aggregated class-wise precision, recall and F_1 values for the proposed model are shown in the Table 4.5. Based on the experimental results, classifying sentences for categories *praise* and *complaint* is a rather simple task for the given corpus. For instance, the first sentence below is from the *praise* class, and the second is from *complaint*:

1. “ホテルの周辺はサッカー場や公園で広々としており、駐車場も十分にあ

り当然無料でした。” (“There was a soccer field and a park nearby the hotel, with sufficient amount of parking space which was of course, free.”)

2. “評価などを見て予約して楽しみにしてたけど食事も全く写真と違うからガッカリしました。” (“I made reservation based on the reviews and was looking forward to it, but the food was completely different from the pictures which was rather disappointing.”)

It has proved to be much more difficult to recognize *neutral* sentences. This can be attributed to the fact that the annotation method differentiates between categories of *no sentiment* and *neutral*. The neutral sentiment label in the dataset means that the annotated statement is a mixture of positive and negative sentiments, making it “neutral”. For example, the two sentences below are labeled as *neutral*:

1. “夕食ルームサービスは全体として満足ですが、メインの鍋焼味噌カツは味が濃すぎでした。” (“I was satisfied with the dinner room service overall, but the hot pot miso pork cutlet’s taste was too strong.”).

2. “朝食のご飯が柔らかすぎ以外はとてもよかった。” (“Besides the rice served at breakfast being too mushy, everything was very nice.”)

This particular characteristic, and the small number of samples makes this class somewhat obscure, and especially difficult to predict. The inconsistency of prediction performance is also reflected in the precision and recall scores over the 10 folds of the cross validation, shown in Figure 4.13.

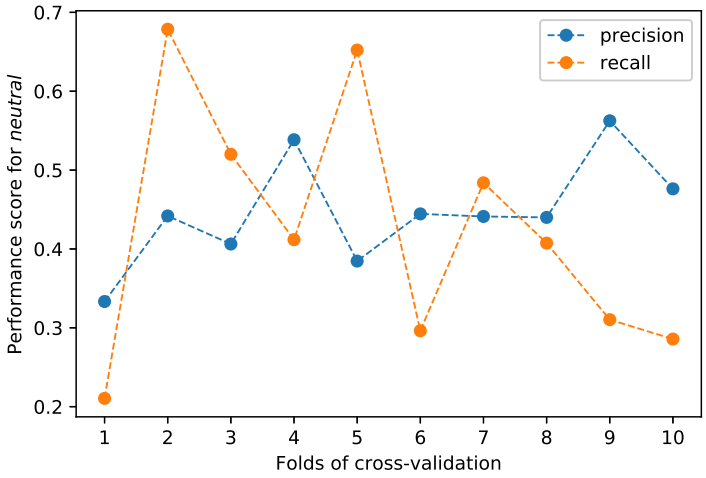

Figure 4.13: Inconsistent precision and recall scores over the cross validation folds for the *neutral* category

Typically, sentences annotated as *no sentiment* are factual observations. The lack of stronger emotional words, and the larger number of samples makes this category easier to learn for the network. For instance:

1. “12 月 25 日に宿泊しました。” (“I stayed at the hotel on the 25th of December.”)
2. “友達 3 人での旅行で利用させていただきました。” (“I stayed there with two of my friends.”)

The F_1 score (Table 4.5) of class *request* is relatively high considering the limited number of training examples. The explanation for this would be that suggesting, proposing, demanding or requesting something puts restrictions on the grammatical form of the sentence and word usage. For example; “... 改善する余地がある” (“...has a place for improvement”), or “... して欲しかった” (“wanted them to...”), etc. For example:

1. “大浴場のロッカーは壊れているところがありますので、そろそろ直して欲しいと思ってます。” (“Some of the lockers at the public bath are broken, so it would be time to fix them.”)
2. “また、風呂場にはアメニティーを備えてもいいのかもと思いました。” (“Also, I think it would be good to have bathroom amenities prepared for the guests.”)

This characteristic makes the category unambiguous, and significantly simpler to classify than *neutral*.

In this case study, the language model GloVe was implemented to acquire word vectors, in order to demonstrate the feasibility of using external knowledge in deep learning-based systems with a widely-used embedding model. Using contextualized embeddings like BERT or ELMo, however, could be used to increase overall performance.

4.4 Technical details

This section describes the technologies and techniques used in the case studies, but were outside of the scope of Section 2.3 (Technological background and challenges).

4.4.1 Performance metrics

Table 4.6: Confusion matrix for a binary classification task.

		Predicted class	
		Positive	Negative
Actual class	Positive	True Positives (TP)	False Negatives (FN)
	Negative	False Positives (FP)	True Negatives (TN)

The performance metrics used in the case studies are *precision, recall,* F_1 *score*, and *accuracy*. Given a confusion matrix for a binary classification example (shown in Table 4.6), precision and recall are calculated as follows:

$$precision = \frac{TP}{TP + FP}, \tag{4.2}$$

$$recall = \frac{TP}{TP + FN}. \tag{4.3}$$

While TPs and TNs are the correctly predicted positive and negative values, FPs and FNs are the falsely classified values. Precision defines the ratio of correctly classified positive samples to the total positive samples predicted, and recall signifies how many actual positives the machine learning model classified as positives. The F_1 score balances precision and recall, i.e. considers both FP and FN:

$$F_1 = 2\frac{recall \times precision}{recall + precision}. \tag{4.4}$$

In the case of multi-class classification, calculating aggregated precision and recall involves computing class-wise scores and averaging them for the total number of classes. *Accuracy* is the number of the correctly classified samples over the total number of samples, and can be represented in the terms of the confusion matrix in the following way:

$$accuracy = \frac{TP + TN}{TP + FP + TN + FN}. \tag{4.5}$$

4.4.2 Cross-validation

Cross-validation is a method for evaluating the performance of machine learning models. Although it is a widely used method to achieve unbiased

classification results, it is particularly useful when a limited amount of data is available, or when the class distribution is skewed or imbalanced. Instead of using a simple 70%/30% or 80%/20% train/test split, the data is resampled into k folds, where "k" refers to the number of groups the data is split into to evaluate the machine learning model, e.g. 5-fold cross-validation. The cross-validation procedure that returns performance scores $perf\_scores$ is described by Algorithm 9.

Algorithm 9 k-fold cross-validation

```
procedure CROSS-VALIDATION(k, data, model)
    initialize list perf_scores
    data = shuffle data
    list folds = divide data into k unique groups
    for all fold in folds do
        test_data = fold
        training_data = folds\fold
        fit model on training_data
        perf_score = evaluate model on test_data
        append perf_score to perf_scores
    end for
    return perf_scores
end procedure
```

With the above-described procedure, each datapoint is used in the test set one time, and k-1 times for training. The obtained $perf\_scores$ are averaged over the k folds, giving an aggregated score. In the experiments conducted in this study, *stratified* k-fold cross-validation is used to ensure obtaining unbiased results. When the class distribution stays the same in each fold, the cross-validation is stratified, and it is the preferred technique when dealing with imbalanced datasets.

4.4.3 Krippendorff's Alpha

Krippendorff's alpha coefficient [124] is a statistic used to calculate inter-annotator agreement (sometimes called inter-rater reliability). It can be applied to any number of annotators with both small and large sample sizes on various levels of measurement. The coefficient is given by:

$$\alpha = 1 - \frac{\sum_{\omega=1,\omega'=1}^{\Omega} o_{\omega\omega'}\delta(\omega\omega')}{\sum_{\omega=1,\omega'=1}^{\Omega} e_{\omega\omega'}\delta(\omega\omega')}, \tag{4.6}$$

where $o_{\omega\omega'}$ is the frequency of paired values ω and ω' from the coincidence matrix of the annotation, with a total number of value pairs Ω. $e_{\omega\omega'}$ denotes the frequencies of expected coincidences of ω and ω':

$$e_{\omega\omega'} = \frac{\sum_{i \neq i'}^{m} Z(\omega_{iu} = \omega) \cdot Z(\omega_{i'u} = \omega')}{n - 1}, \tag{4.7}$$

where m is the number of annotators, n is the number of pairable values, and ω_{iu} and $\omega_{i'u}$ are the values from coders i and i' corresponding to the annotated unit u. Function $Z(\cdot)$ returns 1 if $\cdot$ is true, 0 otherwise. $\delta(\omega\omega')$ from Equation 4.6 is a difference function between values ω and ω', depending on the level of measurement. For ordinal data, $\delta(\omega\omega')$ is defined as follows:

$$\delta(\omega\omega') = \left(\sum_{g=\omega}^{g=\omega'} n_g - \frac{n_\omega + n_{\omega'}}{2} \right)^2, \tag{4.8}$$

where n_g denotes the total number of frequencies corresponding to the paired values for g. The minimum acceptable α is task specific, but the range of the statistic is as follows:

- $\alpha = 1$ suggests perfect inter-annotator agreement,
- $\alpha = 0$ indicates that the annotation is not reliable, and the values are unrelated,
- $\alpha < 0$ implies that the disagreements among annotators are systematic, and not random.

4.4.4 Recurrent and Long Short-Term Memory neural networks

Recurrent Neural Network (RNN) is the preferred network architecture type for processing sequential data (e.g. text, audio, various time-series data, etc.), due to its internal memory that ensures that temporal dynamics is incorporated to the network. A feed-forward network processes data in one direction, and can only "remember" through the weight update during training. In contrast, an RNN considers both current and past information when making predictions. In practice, this means that a recurrent layer assigns weights to the current and previous inputs. The main issue with standard RNNs is that gradient values tend to become extremely small prematurely, which results in very long training times, or making the updates insignificant, and the network basically stops

learning. This issue called the *vanishing gradients* [131], was eventually solved by an extension of standard RNN called Long Short-Term Memory(LSTM) [70] networks.

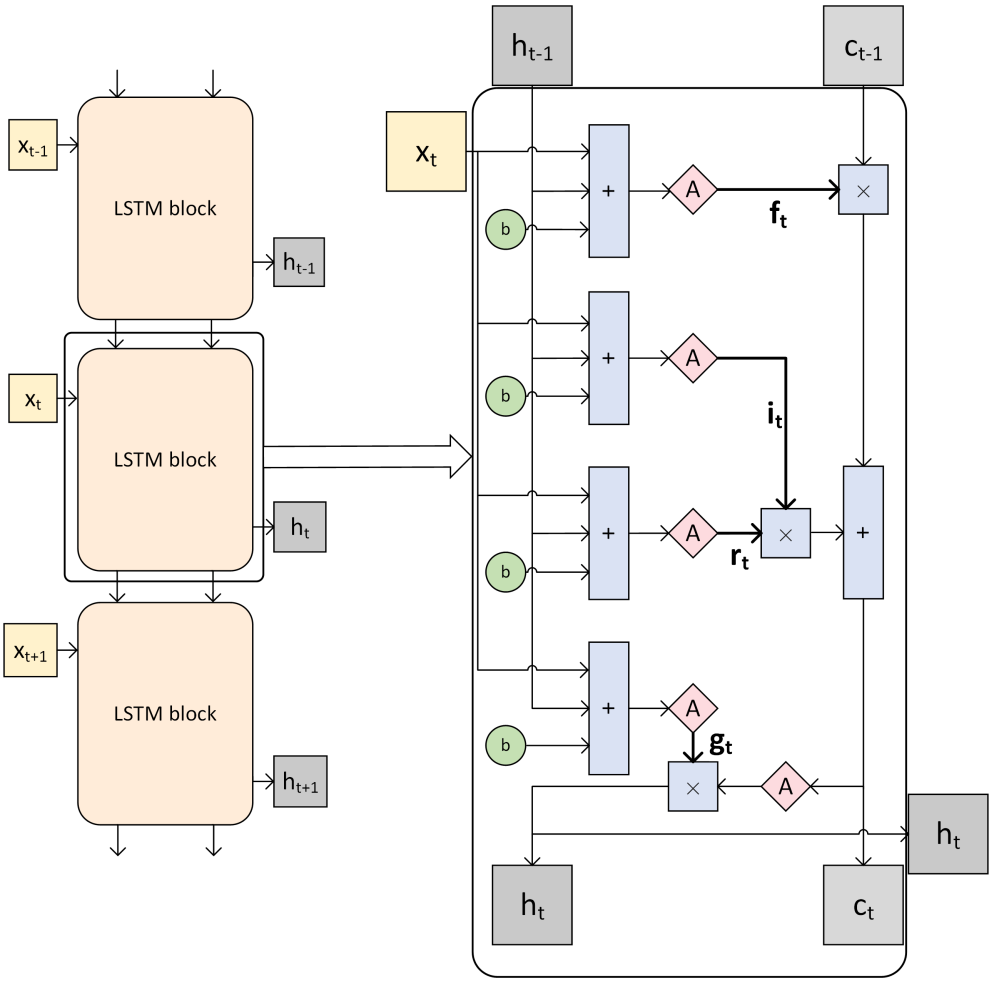

Figure 4.14: The repeating structure of LSTM building blocks with a detailed block at time t

Figure 4.14 shows the repeating structure of LSTM blocks (sometimes called a cell). The text is processed in a forward direction by the network, in the same manner as humans read. There are three different inputs for an individual LSTM block: the new input x_t at a given time t, the previous hidden state output h_{t-1}, and the internal memory state c_{t-1} from the previous block. The built-in memory determines how many new memories should be made, and the amount of information retained from previous states. The $\times$ and $+$ operations

are multiplication and summation, and b refers to the biases. A denotes the activation functions, either sigmoid (σ) or hyperbolic tangent (tanh). The outputs are the present block's internal memory state c_t and the hidden state h_t, which depends not only on the current input, but also on the previous state. In order to calculate c_t and h_t, the activation vector f_t of the *forget gate*, the activation vector i_t of the *input gate*, the internal memory state activation vector r_t, and the *output gate*'s activation vector g_t have to be computed:

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f), \tag{4.9}$$
$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i), \tag{4.10}$$
$$r_t = \tanh(W_c \cdot [h_{t-1}, x_t] + b_r), \tag{4.11}$$
$$g_t = \sigma(W_g \cdot [h_{t-1}, x_t] + b_g), \tag{4.12}$$

where W denotes the weight matrices to be learned during training. Accordingly, the internal memory cell state and the hidden state output are calculated as follows:

$$c_t = f_t \cdot c_{t-1} + i_t \cdot r_t \tag{4.13}$$
$$h_t = g_t \cdot \tanh(c_t). \tag{4.14}$$

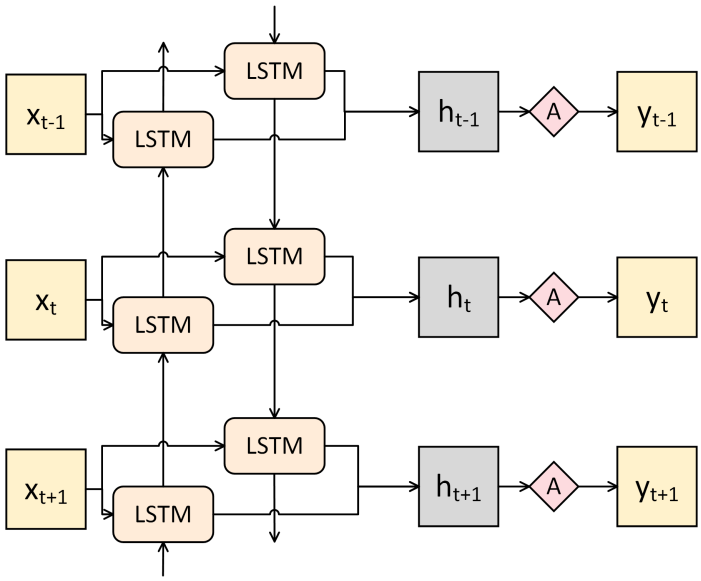

Figure 4.15: The BiLSTM architecture unfolded for three steps

Figure 4.15 shows the unfolded structure of a BiLSTM network for three

timesteps. A BiLSTM block involves two different LSTM blocks, one for forward and another for backward processing. This basically means that the text is processed from left to right by one layer and from right to left by the other. The two directions of a BiLSTM network do not possess direct links in the network, and their hidden states are concatenated before passing them into the next layer or activation function to get predictions y.

4.4.5 One-dimensional convolution and pooling

While two-dimensional Convolutional Neural Networks (CNN) are particularly popular in fields such as object and face recognition, one-dimensional CNN layers are widely used in language modeling, activity recognition, signal processing etc. In the context of machine learning, convolution is the mathematical operation of taking the dot product of a weight matrix called a *filter* and a given input. Since the filter is always smaller than the input, the operation is performed multiple times on the input, resulting in a *feature map*. The *stride* of the convolution defines how many "steps" the filter is shifted over the input for the next operation. For example, if the input is an image and the stride is 1, the filter is moved 1 pixel at a time. If the filter does not fit the input at the edges, the input is padded with zeros to make it suitable for the filter. Therefore, the size of the feature map depends on the filter size, and the stride.

The intuition behind the "filtered" representation of the input is that since the same filter is applied several times, it will discover reoccurring features regardless of their location. When the input is visual, usually two-dimensional CNN is used (with filters specified in *width* $\times$ *height*). In the case of textual inputs, the network must be one-dimensional (with filters specified in *width*).

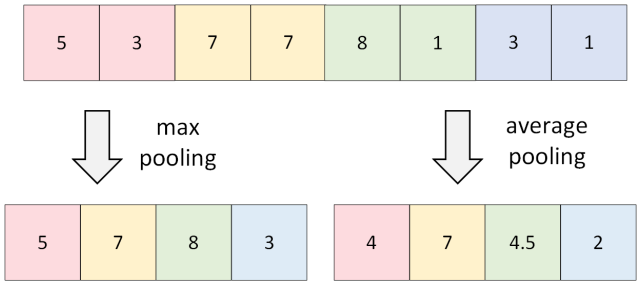

Figure 4.16: Example of max and average pooling operations for a size 2 pooling block with stride 2

Pooling operations are often performed on the acquired feature map to downscale the feature space into a condensed representation. A simple example is given in Figure 4.16 about average and max pooling for a 8×1 feature map, with a size 2 pooling block of stride 2. While max pooling takes the maximum value of the pooling block, average pooling outputs the mean of the observed block to downscale the feature space. In case study II. (Section 4.3), the dimensions of the pooling block and its stride is identical to the filter size and stride.

4.4.6 Optimizer, loss and activation functions

The optimizer used in both case studies in the training phase is Adaptive Moment Estimate (Adam) [132]. Unlike other adaptive optimizers like Adadelta and RMSprop, Adam keeps updating an exponentially decaying average of the past gradients (*first moment*) and the past squared gradients (*second moment*), m and v, respectively. The update for weight w at training iteration $t+1$ is given by:

$$w^{t+1} \leftarrow w^t - \eta\Big[\Big(\frac{m_w^{t+1}}{1-\beta_1^{t+1}}\Big)\Big/\Big(\sqrt{\frac{v_w^{t+1}}{1-\beta_2^{t+1}}} + \epsilon\Big)\Big], \tag{4.15}$$

where η is the learning rate, and ϵ is a scalar (usually 10^{-8}) to prevent division by 0. m_w^{t+1} and v_w^{t+1} are defined as:

$$m_w^{t+1} \leftarrow \beta_1 m_w^t + (1-\beta_1)\nabla_w L^t, \tag{4.16}$$

$$v_w^{t+1} \leftarrow \beta_2 v_w^t + (1-\beta_2)(\nabla_w L^t)^2, \tag{4.17}$$

where L is the loss function, and β_1 and β_2 are the decaying factors for m and v. For the experiments, both *beta* parameters are set to close to 1, as it was suggested by the original paper [132] ($\beta_1 = 0.9$, $\beta_2 = 0.999$). In both case studies, *categorical cross-entropy* was used for loss functions L, as it is suitable for multi-class classification tasks. It is defined as follows:

$$L_i = -\frac{1}{N}\sum_{j=1}^{N}\sum_{c}^{C} t_{j,c}\log(p_{j,c}), \tag{4.18}$$

where $j \in \{1, 2, 3, ..., N\}$ is a datapoint, and $c \in \{1, 2, 3, ..., C\}$ is one of the classes. While $t_{j,c}$ is the target function marking that j belongs to c, $p_{j,c}$, is the probability of j belonging to c, predicted by the network. The activation function usually used together with categorical cross-entropy is *softmax*. The softmax function converts the output of the last feed-forward layer into

probabilities associated with the classes to be predicted, and it is given by:

$$f(x)_i = \frac{e^{x_j}}{\sum_{k=1}^{K} e^{x_k}}, \tag{4.19}$$

where x is the reference vector of the output, and $i \in \{1, 2, 3, ..., K\}$ is the output index. As outputs are representing probabilities, the function is suitable for multi-class classification, unlike e.g. the sigmoid function.

Chapter 5

Overall discussion

Discovering product features is a powerful method for analyzing reviews and assessing "helpfulness" [47, 133]. The same attribute, however, can be expressed in different ways (synonyms, implicitly or explicitly, referencing, etc.), and apparently different attributes may still be referencing the same fundamental aspect. The meaning and usage of words can be different depending on the domain and the target audience, and the appearance of words is probabilistic. Therefore, even though a word is not widely used in a specific domain, there is always some chance to encounter that word. The proposed framework captures this concept by the theoretical model of pertinence (Section 3.2). Word semantics and pragmatics are incorporated into the domain V and target population T specific embeddings Ψ_V^T, and the word weights Φ_V^T are computed to represent the probabilistic nature of word occurrence for the given V and T. Accordingly, the proposed model of pertinence clearly differentiates between the quality of reviews perceived by product designers and the review quality seen by customers, which is known to be greatly different [28, 107], while still often neglected in review helpfulness studies (Section 2.2).

In case study I., although fundamental product attributes in the domain of *Digital cameras and Accessories* received the largest weights (words such as *sensor, aperture, zoom, exposure, mount, iso, etc.*), verbs, adjectives, and adverbs which are not inherently product attributes are also observed even in the top few hundred terms (of the total of ~13,000 terms). The term *compensate*, for example, can hardly be deemed as a product attribute, however, depending on the context, it could be an important word in the domain. For instance, *"shutter changes automatically to compensate", "compensate for exposure errors", "to compensate, use lower ISO"*, etc. In prior work, usually, only a couple of hand-selected features (mostly only nouns) are considered for review quality assessment, without assigning weights or incorporating semantic information into the feature representations.

The language model BERT could not be retrained for the target task without quantifying sentence pertinence and building a pre-annotated dataset. Experimental results showed that only fine-tuning the language model results in a worse performance than a baseline SVM, due to negative transfer

(Section 4.2.4). Furthermore, since the proposed framework is sentence-based to make it suitable for requirements engineering, selecting a few thousands of sentences for annotation would also entail manually scanning over thousands or even tens of thousands of reviews to construct a representative data set, if sentences are not pre-annotated first.

Although in a real word scenario when developing tools for requirements engineering, not all parts of the proposed framework would be utilized, to examine the practical viability of the proposed framework for potential deployment, system operation has been simulated for the system introduced in case study I. Figure 5.1 demonstrates how the runtime t changes with an increasing number of sentences n with a batch size of 64 sentences used for model prediction.

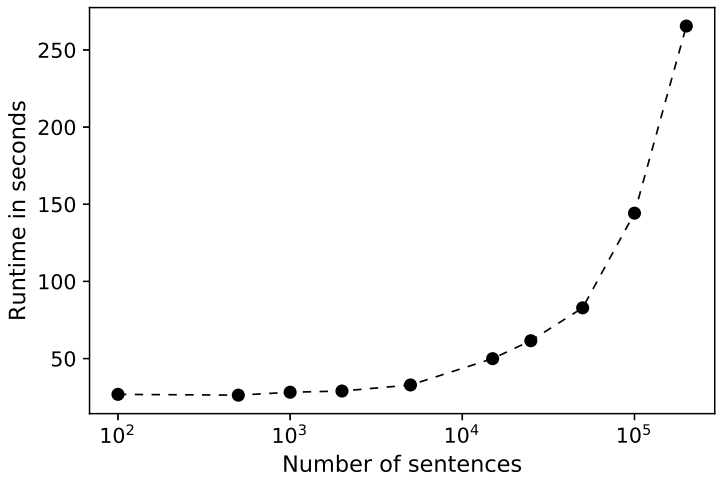

Figure 5.1: An example of system runtime for an increasing number of input sentences

While system runtime for 100 sentences to process was 26.81 s, for n=5000, it was still only 32.88 s. The reason for this is that preparing and loading the model is the same in all cases, and for a smaller to moderate amount of data, this takes a longer time than the actual processing and prediction. For a greater number of sentences, while more time is spent on applying the fine-tuned BERT-Base model, system runtime was still less than 5 minutes (for example, t=265.48 for n=200,000). This experiment showed that, once the model is loaded, the system based on the proposed framework would label the newly posted reviews in almost real-time, and could even be used to label relatively large datasets in a couple of minutes.

Considering the GPU memory needed and the time required to train BERT-Large [114], experiments have been performed with BERT-Base. While in most cases, the 24-layer BERT-Large does not improve classification performance substantially, it allows for 1 to 5% improvement depending on the application [114]. Thus, using BERT-Large with the proposed method to acquire domain and target population specific word embeddings can potentially improve the performance of the pertinence quantification algorithm.

As argued in Section 2.1, although it eliminates the need of human annotation, converting user ratings (usually stars) from the reviews into labels is not suitable for requirements engineering, as it is an imprecise and unreliable measure of customer sentiment that usually involves only binary classification and it is on the whole review level. While sentence-based approaches exist, the reviewed literature does not address the possibility of integrating external knowledge directly into deep learning models to capture categories of higher levels of abstraction (e.g. emotions), and usually only rely on features obtained from vectorizing the reviews. Case study II. addresses this issue by implementing the proposed framework to incorporate external information into a deep learning model (Section 3.3.2.3). Experimental results demonstrated that the proposed approach for expanding the feature space of deep neural networks is appropriate for multi-class sentence level sentiment analysis, especially when a limited amount of samples are available for the classes. While prediction performance will not reach the levels of a model that is trained on millions of datapoints using the review stars (particularly in the case when there are classes with extremely low amount of instances), the obtained model would be still more useful in the context of requirements engineering, as it is fine-grained and does not require substantial human intervention to analyze the reviews further. For example, sentences about requesting, complaining, suggestions, etc., could be used during requirements specification and validation. Sentences that praise some aspects of a product or service could be used to learn about customer preferences and would be useful during requirements elicitation.

Chapter 6

Conclusions

In this chapter, the contributions of this work are highlighted, and the limitations of the proposed approach with possible future work directions are discussed.

In the given study, a conceptual framework is proposed for developing requirements engineering tools that would reduce the information overload associated with online customer reviews and help companies elicit valuable knowledge about customer needs. Related studies focusing on the extraction and evaluation of customer needs suffer from several limitations:

- Despite most of the information customer reviews contain are not relevant for requirements engineering purposes, assessing review quality is a seldom-discussed topic in the related literature. The prior work addressing this issue usually apply methods that require extensive human involvement, yet do not differentiate between the quality seen by customers and the quality perceived by product designers. Furthermore, review quality assessment is usually performed on the document level, which is inaccurate and ambiguous.
- Converting the review stars to labels is a convenient way to avoid human annotation and to use large datasets, so robust deep learning models can be built. For this reason, most of the studies analyzing review sentiment aim at increasing classification performance of binary or three-class schemes on the document level. However, even if the observed classification accuracy is high, such methods are not fine-grained enough for industrial applications.

6.1 Contributions

Major contributions of the presented study are summarized as follows:

1. In order to make the framework fit for requirements engineering, it utilizes the proposed theoretical model of pertinence to quantify review quality automatically with a novel algorithm that uses domain and target population specific word embeddings and weights.

- Experimental results of case study I. demonstrate that the proposed approach is appropriate for measuring review quality on the sentence level. Besides using it in various data mining projects to assess the quality of large datasets, it can be used as an automatic pre-annotation technique for differentiating between multiple levels of review informativeness using the suggested transfer learning strategy.

2. An original approach was proposed to incorporate external knowledge into machine learning models by expanding their feature space, allowing one to efficiently apply deep learning on small datasets that are human-annotated with labels more relevant for companies than the labels created from stars.

    - Experimental results of case study II. show that the approach is suitable for integrating additional semantic information into the training process using the proposed network architecture, and would benefit companies dealing with small, human-annotated datasets for customer sentiment prediction.

6.2 Limitations and future work

6.2.1 Towards closer approximation of document pertinence

By definition, it is impossible to have access to absolutely all documents (that exist and will exist) for a given domain V and target population T to 100% accurately model Ψ_V^T and Φ_V^T. Therefore, pertinence as it is formulated in the theoretical model cannot be precisely calculated and can only be approximated.

The performance of the proposed algorithm for quantifying pertinence (Section 3.3) mainly depends on three factors:

a.) The representative power of the stationary word vectors

b.) The weight calculation method

c.) Initial review dataset size

In the case of the proposed approach (Section 3.3.2.1) to acquire stationary vectors from a contextual and sentence-based embedding model, the representative power of the vectors (a.)) depends on the performance of state-of-the-art embedding models, i.e. current state of language modeling

technology. Similar to most word weighting methods, such as the widely-used tf-idf and its variations, the weighting method (b.)) used in the proposed framework assumes mutual independence between the scored words. As it was argued for word vectors, however, this is clearly not the case for natural languages. Thus, future work would include using an alternative approach for weight calculation. Since the outputs of the pertinence quantification procedure would not be in any predefined range without normalization, it is essential to use a large review corpus (c.)) initially, to obtain representative pertinence scores for a given V and T. Although the process does not require any human annotation, this can still be an issue with low-resource languages where large datasets for a specific product category are not necessarily available.

6.2.2 Contextualized word embeddings with polarities

The part of the proposed framework that involves incorporating external knowledge into a deep learning model is designed so that the word embeddings and feature encodings are first processed in separate pipelines. The reason for this is to let the network weigh the importances of the two different feature representations $\mathcal{X}$ and $\mathcal{X}'$. The drawback of this approach is that the appropriate balance between the two feature representations may vary from sentence to sentence (even in the case of the same class), and the suggested method cannot account for such cases. As an alternative approach, the external knowledge could be integrated into a sentence-based contextualized embedding model, e.g. element-wise summation with BERT vectors, for potential performance improvement.

List of publications related to the dissertation

- The concept of pertinence and the results of preliminary experiments with pertinence quantification using the Amazon review dataset have been reported by the author in:
  M. Kovacs and V. V. Kryssanov, "Towards assessing online customer reviews from the product designer's viewpoint," in *Springer Lecture Notes in Computer Sciece: Conference on e-Business, e-Services and e-Society*, pp. 62-74, 2019.

- The main idea of incorporating external data into a deep learning model using the proposed network architecture with a sentiment lexicon was introduced by the author in:
  M. Kovacs and V. V. Kryssanov, "Expanding the feature space of deep neural networks for sentiment classification," *International Journal of Machine Learning and Computing*, vol. 10, no. 2, pp. 271-276, 2020.

Bibliography

[1] C. P. Blocker and D. J. Flint, "Customer segments as moving targets: Integrating customer value dynamism into segment instability logic," *Industrial Marketing Management*, vol. 36, no. 6, pp. 810–822, 2007.

[2] C. Lorenzo-Romero, E. Constantinides, and L. A. Brünink, "Co-creation: Customer integration in social media based product and service development," *Procedia - Social and Behavioral Sciences*, vol. 148, pp. 383–396, 2014.

[3] B. Kapoor, "Business intelligence and its use for human resource management," *The Journal of Human Resource and Adult Learning*, vol. 6, no. 2, pp. 21–30, 2010.

[4] X. Tian and L. Liu, "Does big data mean big knowledge? integration of big data analysis and conceptual model for social commerce research," *Electronic Commerce Research*, vol. 17, no. 1, pp. 169–183, 2017.

[5] G. Kotonya and I. Sommerville, *Requirements engineering: processes and techniques*. John Wiley & Sons, Inc., 1998.

[6] P. Loucopoulos and V. Karakostas, *System requirements engineering*. McGraw-Hill, Inc., 1995.

[7] J. Jiao and C.-H. Chen, "Customer requirement management in product development: a review of research issues," *Concurrent Engineering*, vol. 14, no. 3, pp. 173–185, 2006.

[8] B. Verworn, C. Herstatt, and A. Nagahira, "The fuzzy front end of japanese new product development projects: impact on success and differences between incremental and radical projects," *R&d Management*, vol. 38, no. 1, pp. 1–19, 2008.

[9] K. Laudon and C. Traver, *E-Commerce: Business, Technology, Society (3rd Edition)*. New Jersey, USA: Prentice-Hall, 2006.

[10] V. Mahajan, E. Muller, and R. A. Kerin, "Introduction strategy for new products with positive and negative word-of-mouth," *Management Science*, vol. 30, no. 12, pp. 1389–1404, 1984.

[11] R. Robinson, T.-T. Goh, and R. Zhang, "Textual factors in online product reviews: a foundation for a more influential approach to opinion mining," *Electronic Commerce Research*, vol. 12, no. 3, pp. 301–330, 2012.

[12] M. Trusov, R. E. Bucklin, and K. Pauwels, "Effects of word-of-mouth versus traditional marketing: Findings from an internet social networking site," *Journal of Marketing*, vol. 73, no. 5, pp. 90–102, 2009.

[13] L. Zhu, G. Yin, and W. He, "Is this opinion leader's review useful? peripheral cues for online review helpfulness," *Journal of Electronic Commerce Research*, vol. 15, pp. 267–280, 2014.

[14] Y. Shen, S. Li, and M. DeMoss, "The effect of quantitative electronic word of mouth on consumer perceived product quality," *International Journal of Management and Marketing Research*, vol. 5, no. 2, pp. 19–29, 2012.

[15] M. D. Sotiriadis and C. Van Zyl, "Electronic word-of-mouth and online reviews in tourism services: the use of twitter by tourists," *Electronic Commerce Research*, vol. 13, no. 1, pp. 103–124, 2013.

[16] S. Moore, "Attitude predictability and helpfulness in online reviews: The role of explained actions and reactions," *Journal of Consumer Research*, vol. 42, pp. 30–44, 2015.

[17] W. W. Moe and M. Trusov, "The value of social dynamics in online product ratings forums," *Journal of Marketing Research*, vol. 48, no. 3, pp. 444–456, 2011.

[18] Y. Liu, "Word-of-mouth for movies: Its dynamics and impact on box office revenue," *Journal of Marketing*, vol. 70, pp. 74–89, 2006.

[19] C. Dellarocas, X. M. Zhang, and N. Awad, "Exploring the value of online product reviews in forecasting sales: The case of motion pictures," *Journal of Interactive Marketing*, vol. 21, pp. 23–45, 2007.

[20] J. Jin, P. Ji, and R. Gu, "Identifying comparative customer requirements from product online reviews for competitor analysis," *Engineering Applications of Artificial Intelligence*, vol. 49, pp. 61–73, 2016.

[21] J. Qi, Z. Zhang, S. Jeon, and Y. Zhou, "Mining customer requirements from online reviews: A product improvement perspective," *Information & Management*, vol. 53, no. 8, pp. 951–963, 2016.

[22] C. C. Aguwa, L. Monplaisir, and O. Turgut, "Voice of the customer: Customer satisfaction ratio based analysis," *Expert Systems with Applications*, vol. 39, no. 11, pp. 10112–10119, 2012.

[23] Y.-J. Park, "Predicting the helpfulness of online customer reviews across different product types," *Sustainability*, vol. 10, p. 1735, 2018.

[24] O. Netzer, R. Feldman, J. Goldenberg, and M. Fresko, "Mine your own business: Market-structure surveillance through text mining," *Marketing Science*, vol. 31, no. 3, pp. 521–543, 2012.

[25] H. Zhang, H. Rao, and J. Feng, "Product innovation based on online review data mining: a case study of huawei phones," *Electronic Commerce Research*, vol. 18, no. 1, pp. 3–22, 2018.

[26] F. Ferreira, J. Faria, A. Azevedo, and A. L. Marques, "Product lifecycle management in knowledge intensive collaborative environments," *International Journal of Information Management*, vol. 37, no. 1, pp. 1474–1487, 2017.

[27] X. Yu, Y. Liu, X. Huang, and A. An, "Mining online reviews for predicting sales performance: A case study in the movie domain," *IEEE Transactions on Knowledge and Data Engineering*, vol. 24, no. 4, pp. 720–734, 2012.

[28] I. A. Yagci and S. Das, "Design feature opinion cause analysis: a method for extracting design intelligence from web reviews," *International Journal of Knowledge and Web Intelligence*, vol. 5, no. 2, pp. 127–145, 2015.

[29] J. Zhan, H. T. Loh, and Y. Liu, "Gather customer concerns from online product reviews – a text summarization approach," *Expert Systems with Applications*, vol. 36, no. 2, pp. 2107–2115, 2009.

[30] J. Polpinij and A. K. Ghose, "An ontology-based sentiment classification methodology for online consumer reviews," in *2008 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology*, vol. 1, pp. 518–524, 2008.

[31] A. R. Sulthana and R. Subburaj, "An improvised ontology based k-means clustering approach for classification of customer reviews," *Indian Journal of Science and Technology*, vol. 9, no. 15, pp. 1–6, 2016.

[32] R. Alfrjani, T. Osman, and G. Cosma, "A hybrid semantic knowledgebase-machine learning approach for opinion mining," *Data & Knowledge Engineering*, vol. 121, pp. 88–108, 2019.

[33] J. Villena-Román, S. Collada-Pérez, S. Lana-Serrano, and J. C. González-Cristóbal, "Hybrid approach combining machine learning and a rule-based expert system for text categorization," in *Twenty-Fourth International FLAIRS Conference*, 2011.

[34] A. Qazi, K. Syed, R. Raj, E. Cambria, M. Tahir, and D. Alghazzawi, "A concept-level approach to the analysis of online review helpfulness," *Computers in Human Behavior*, vol. 58, pp. 75–81, 2016.

[35] Y. Liu, X. Huang, A. An, and X. Yu, "Modeling and predicting the helpfulness of online reviews," in *Proceedings of the 2008 Eighth IEEE International Conference on Data Mining*, ICDM '08, pp. 443–452, 2008.

[36] O. Tsur and A. Rappoport, "Revrank: A fully unsupervised algorithm for selecting the most helpful book reviews," in *Proceedings of the Third International Conference on Weblogs and Social Media, ICWSM 2009*, 2009.

[37] A. Ghose and P. Ipeirotis, "Estimating the helpfulness and economic impact of product reviews: Mining text and reviewer characteristics.," *IEEE Trans. Knowl. Data Eng.*, vol. 23, pp. 1498–1512, 2011.

[38] Y. Dezhi, S. Bond, and Z. Han, "Dreading and ranting: the distinct effects of anxiety and anger in online seller reviews," in *International Conference on Information Systems 2011 Shanghai*, vol. 9, pp. 1–20, 2011.

[39] D. Godes and J. C. Silva, "Sequential and temporal dynamics of online opinion," *Marketing Science*, vol. 31, pp. 448–473, 2012.

[40] H. Lee, K. Choi, D. Yoo, Y. Suh, S. Lee, and G. He, "Recommending valuable ideas in an open innovation community: A text mining approach to information overload problem," *Industrial Management & Data Systems*, vol. 118, no. 4, pp. 683–699, 2018.

[41] X. Sun, M. Han, and J. Feng, "Helpfulness of online reviews: Examining review informativeness and classification thresholds by search products and experience products," *Decision Support Systems*, vol. 124, 2019.

[42] S. P. Eslami, M. Ghasemaghaei, and K. Hassanein, "Which online reviews do consumers find most helpful? a multi-method investigation," *Decision Support Systems*, vol. 113, pp. 32–42, 2018.

[43] R. Filieri, "What makes online reviews helpful? a diagnosticity-adoption framework to explain informational and normative influences in e-wom," *Journal of Business Research*, vol. 68, 2014.

[44] A. Huang, K. Chen, D. Yen, and T. Tran, "A study of factors that contribute to online review helpfulness," *Computers in Human Behavior*, vol. 48, pp. 17–27, 2015.

[45] H. Hong, D. Xu, G. Wang, and W. Fan, "Understanding the determinants of online review helpfulness: A meta-analytic investigation," *Decision Support Systems*, vol. 102, pp. 1–11, 2017.

[46] J. Mackiewicz and D. Yeats, "Product review users' perceptions of review quality: The role of credibility, informativeness, and readability," *IEEE Transactions on Professional Communication*, vol. 57, no. 4, pp. 309–324, 2014.

[47] J. Liu, Y. Cao, C.-Y. Lin, Y. Huang, and M. Zhou, "Low-quality product review detection in opinion summarization," in *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pp. 334–342, 2007.

[48] Y. Liu, J. Jin, P. Ji, J. A. Harding, and R. Y. K. Fung, "Identifying helpful online reviews: A product designer's perspective," *Computer-Aided Design*, vol. 45, no. 2, pp. 180–194, 2013.

[49] Q. Liu, G. Feng, N. Wang, and G. K. Tayi, "A multi-objective model for discovering high-quality knowledge based on data quality and prior knowledge," *Information Systems Frontiers*, vol. 20, no. 2, pp. 401–416, 2018.

[50] A. N. Abukhalifeh and A. P. M. Som, "Servqual: A multiple–item scale for measuring customer perceptions of restaurants' service quality," *Advances in Environmental Biology*, vol. 9, no. 3, pp. 160–162, 2015.

[51] L. Bovey, K. Holt, H. Geschka, and G. Peterlongo, "Need assessment—a key to user-oriented product innovation," *R&D Management*, vol. 15, no. 3, pp. 258a–258a, 1985.

[52] D. Godes and D. Mayzlin, "Using online conversations to study word-of-mouth communication," *Marketing science*, vol. 23, no. 4, pp. 545–560, 2004.

[53] E. T. Bradlow, *User-Generated Content: The "Voice of the Customer" in the 21st Century*, pp. 27–29. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010.

[54] J. Li and L. Zhan, "Online persuasion: How the written word drives wom: Evidence from consumer-generated product reviews," *Journal of Advertising Research*, vol. 51, no. 1, pp. 239–257, 2011.

[55] X. Li and L. M. Hitt, "Price effects in online product reviews: An analytical model and empirical analysis," *MIS quarterly*, vol. 34, no. 4, pp. 809–831, 2010.

[56] C.-H. Chen, L. P. Khoo, and W. Yan, "A strategy for acquiring customer requirement patterns using laddering technique and art2 neural network," *Advanced Engineering Informatics*, vol. 16, no. 3, pp. 229–240, 2002.

[57] C.-H. Chen and W. Yan, "An in-process customer utility prediction system for product conceptualisation," *Expert Systems with Applications*, vol. 34, no. 4, pp. 2555–2567, 2008.

[58] T. Lee and E. T. Bradlow, "Automatic construction of conjoint attributes and levels from online customer reviews," *University Of Pennsylvania, The Wharton School Working Paper*, 2007.

[59] K. Zhang, R. Narayanan, and A. N. Choudhary, "Voice of the customers: Mining online customer reviews for product feature-based ranking," *WOSN*, vol. 10, pp. 11–11, 2010.

[60] T. Lang and M. Rettenmeier, "Understanding consumer behavior with recurrent neural networks," in *Workshop on Machine Learning Methods for Recommender Systems*, 2017.

[61] H. Salehinejad and S. Rahnamayan, "Customer shopping pattern prediction: A recurrent neural network approach," in *2016 IEEE Symposium Series on Computational Intelligence (SSCI)*, pp. 1–6, 2016.

[62] L. M. Badea, "Predicting consumer behavior with artificial neural networks," *Procedia Economics and Finance*, vol. 15, pp. 238–246, 2014.

[63] A. Ansari and A. Riasi, "Modelling and evaluating customer loyalty using neural networks: Evidence from startup insurance companies," *Future Business Journal*, vol. 2, no. 1, pp. 15–30, 2016.

[64] A. Larasati, C. DeYong, and L. Slevitch, "The application of neural network and logistics regression models on predicting customer satisfaction in a student-operated restaurant," *Procedia-Social and Behavioral Sciences*, vol. 65, pp. 94–99, 2012.

[65] K. Ravi and V. Ravi, "A survey on opinion mining and sentiment analysis: tasks, approaches and applications," *Knowledge-Based Systems*, vol. 89, pp. 14–46, 2015.

[66] B. Liu, "Sentiment analysis: mining sentiments, opinions, and emotions," 2015.

[67] N. Hu, N. S. Koh, and S. K. Reddy, "Ratings lead you to the product, reviews help you clinch it? the mediating role of online review sentiments on product sales," *Decision support systems*, vol. 57, pp. 42–53, 2014.

[68] W. Medhat, A. Hassan, and H. Korashy, "Sentiment analysis algorithms and applications: A survey," *Ain Shams engineering journal*, vol. 5, no. 4, pp. 1093–1113, 2014.

[69] Z. Hailong, G. Wenyan, and J. Bo, "Machine learning and lexicon based methods for sentiment classification: A survey," in *2014 11th web information system and application conference*, pp. 262–265, IEEE, 2014.

[70] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[71] V. Gavrishchaka, Z. Yang, R. Miao, and O. Senyukova, "Advantages of hybrid deep learning frameworks in applications with limited data," *International Journal of Machine Learning and Computing*, vol. 8, no. 6, pp. 549–558, 2018.

[72] X. Wang, W. Jiang, and Z. Luo, "Combination of convolutional and recurrent neural network for sentiment analysis of short texts," in *Proceedings of COLING 2016, the 26th international conference on computational linguistics: Technical papers*, pp. 2428–2437, 2016.

[73] X. Glorot, A. Bordes, and Y. Bengio, "Domain adaptation for large-scale sentiment classification: A deep learning approach," in *ICML*, 2011.

[74] H. Chen, M. Sun, C. Tu, Y. Lin, and Z. Liu, "Neural sentiment classification with user and product attention," in *Proceedings of the 2016 conference on empirical methods in natural language processing*, pp. 1650–1659, 2016.

[75] C. Dos Santos and M. Gatti, "Deep convolutional neural networks for sentiment analysis of short texts," in *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pp. 69–78, 2014.

[76] J. Wang, L.-C. Yu, K. R. Lai, and X. Zhang, "Dimensional sentiment analysis using a regional cnn-lstm model," in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pp. 225–230, 2016.

[77] S. Poria, E. Cambria, G. Winterstein, and G.-B. Huang, "Sentic patterns: Dependency-based rules for concept-level sentiment analysis," *Knowledge-Based Systems*, vol. 69, pp. 45–63, 2014.

[78] E. Cambria, D. Olsher, and D. Rajagopal, "Senticnet 3: a common and common-sense knowledge base for cognition-driven sentiment analysis," in *Proceedings of the twenty-eighth AAAI conference on artificial intelligence*, pp. 1515–1521, 2014.

[79] O. Appel, F. Chiclana, J. Carter, and H. Fujita, "A hybrid approach to the sentiment analysis problem at the sentence level," *Knowledge-Based Systems*, vol. 108, pp. 110–124, 2016.

[80] S. Baccianella, A. Esuli, and F. Sebastiani, "Sentiwordnet 3.0: an enhanced lexical resource for sentiment analysis and opinion mining.," in *Lrec*, vol. 10, pp. 2200–2204, 2010.

[81] M. Huang, Q. Qian, and X. Zhu, "Encoding syntactic knowledge in neural networks for sentiment classification," *ACM Transactions on Information Systems (TOIS)*, vol. 35, no. 3, pp. 1–27, 2017.

[82] C. C. Chen and Y.-D. Tseng, "Quality evaluation of product reviews using an information quality framework," *Decision Support Systems*, vol. 50, no. 4, pp. 755–768, 2011.

[83] J. P. Singh, S. Irani, N. P. Rana, Y. K. Dwivedi, S. Saumya, and P. K. Roy, "Predicting the "helpfulness" of online consumer reviews," *Journal of Business Research*, vol. 70, pp. 346–355, 2017.

[84] Y. Yang, Y. Yan, M. Qiu, and F. Bao, "Semantic analysis and helpfulness prediction of text for online product reviews," in *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics ACL*, pp. 38–44, 2015.

[85] A. Ghose and P. G. Ipeirotis, "Estimating the helpfulness and economic impact of product reviews: Mining text and reviewer characteristics," *IEEE Transactions on Knowledge and Data Engineering*, vol. 23, no. 10, pp. 1498–1512, 2011.

[86] Y. Yang, Y. Yan, M. Qiu, and F. Bao, "Semantic analysis and helpfulness prediction of text for online product reviews," in *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pp. 38–44, 2015.

[87] C. Danescu-Niculescu-Mizil, G. Kossinets, J. Kleinberg, and L. Lee, "How opinions are received by online communities: A case study on Amazon.Com helpfulness votes," in *Proceedings of the 18th International Conference on World Wide Web*, pp. 141–150, 2009.

[88] S. Krishnamoorthy, "Linguistic features for review helpfulness prediction," *Expert Systems with Applications*, vol. 42, no. 7, pp. 3751–3759, 2015.

[89] L. Martin and P. Pu, "Prediction of helpful reviews using emotions extraction," in *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence*, pp. 1551–1557, 2014.

[90] M. P. O'Mahony and B. Smyth, "Learning to recommend helpful hotel reviews," in *Proceedings of the third ACM conference on Recommender systems*, pp. 305–308, 2009.

[91] M. Malik and A. Hussain, "Helpfulness of product reviews as a function of discrete positive and negative emotions," *Computers in Human Behavior*, vol. 73, pp. 290–302, 2017.

[92] S. M. Mudambi and D. Schuff, "What makes a helpful online review? A study of customer reviews on amazon.com," *MIS Quarterly*, vol. 34, pp. 185–200, 2010.

[93] D. Yin, S. D. Bond, and H. Zhang, "Anxious or angry? effects of discrete emotions on the perceived helpfulness of online reviews," *MIS quarterly*, vol. 38, no. 2, pp. 539–560, 2014.

[94] Y.-J. Park and K.-j. Kim, "Impact of semantic characteristics on perceived helpfulness of online reviews," *Journal of Intelligence and Information Systems*, vol. 23, no. 3, pp. 29–44, 2017.

[95] Q. Cao, W. Duan, and Q. Gan, "Exploring determinants of voting for the "helpfulness" of online user reviews: A text mining approach," *Decision Support Systems*, vol. 50, no. 2, pp. 511–521, 2011.

[96] A. Y. Chua and S. Banerjee, "Understanding review helpfulness as a function of reviewer reputation, review rating, and review depth," *Journal of the Association for Information Science and Technology*, vol. 66, no. 2, pp. 354–362, 2015.

[97] S. Zhou and B. Guo, "The order effect on online review helpfulness: A social influence perspective," *Decision Support Systems*, vol. 93, pp. 77–87, 2017.

[98] M. Siering, J. Muntermann, and B. Rajagopalan, "Explaining and predicting online review helpfulness: The role of content and reviewer-related signals," *Decision Support Systems*, vol. 108, pp. 1–12, 2018.

[99] M. Salehan and D. J. Kim, "Predicting the performance of online consumer reviews: A sentiment mining approach to big data analytics," *Decision Support Systems*, vol. 81, pp. 30–40, 2016.

[100] M. Weimer and I. Gurevych, "Predicting the perceived quality of web forum posts," in *Proceedings of the conference on recent advances in natural language processing*, pp. 643–648, 2007.

[101] S. Lee and J. Y. Choeh, "Predicting the helpfulness of online reviews using multilayer perceptron neural networks," *Expert Systems with Applications*, vol. 41, no. 6, pp. 3041–3046, 2014.

[102] M. E. Haque, M. E. Tozal, and A. Islam, "Helpfulness prediction of online product reviews," in *Proceedings of the ACM Symposium on Document Engineering 2018*, pp. 1–4, 2018.

[103] Y. Liu, J. Jin, P. Ji, J. A. Harding, and R. Y. Fung, "Identifying helpful online reviews: a product designer' s perspective," *Computer-Aided Design*, vol. 45, no. 2, pp. 180–194, 2013.

[104] N. Gobi and A. Rathinavelu, "Analyzing cloud based reviews for product ranking using feature based clustering algorithm," *Cluster Computing*, vol. 22, no. 3, pp. 6977–6984, 2019.

[105] C. C. Chen and Y.-D. Tseng, "Quality evaluation of product reviews using an information quality framework," *Decision Support Systems*, vol. 50, no. 4, pp. 755–768, 2011.

[106] S. Saumya, J. P. Singh, A. M. Baabdullah, N. P. Rana, and Y. K. Dwivedi, "Ranking online consumer reviews," *Electronic Commerce Research and Applications*, vol. 29, pp. 78–89, 2018.

[107] I. A. Yagci and S. Das, "Measuring design-level information quality in online reviews," *Electronic Commerce Research and Applications*, vol. 30, pp. 102–110, 2018.

[108] Y.-P. Lin and T.-P. Jung, "Improving eeg-based emotion classification using conditional transfer learning," *Frontiers in human neuroscience*, vol. 11, p. 334, 2017.

[109] M. T. Rosenstein, Z. Marx, L. P. Kaelbling, and T. G. Dietterich, "To transfer or not to transfer," in *NIPS 2005 workshop on transfer learning*, vol. 898, pp. 1–4, 2005.

[110] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Transactions on knowledge and data engineering*, vol. 22, no. 10, pp. 1345–1359, 2009.

[111] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Advances in neural information processing systems*, pp. 3111–3119, 2013.

[112] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation," in *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pp. 1532–1543, 2014.

[113] M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer, "Deep contextualized word representations," *CoRR*, vol. abs/1802.05365, 2018.

[114] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[115] M. Kovacs and V. V. Kryssanov, "Towards assessing online customer reviews from the product designer's viewpoint," in *Springer Lecture Notes*

in Computer Sciece: Conference on e-Business, e-Services and e-Society, pp. 62–74, 2019.

[116] M. Kovacs and V. V. Kryssanov, "Expanding the feature space of deep neural networks for sentiment classification," *International Journal of Machine Learning and Computing*, vol. 10, no. 2, pp. 271–276, 2020.

[117] G. O. Diaz and V. Ng, "Modeling and prediction of online product review helpfulness: a survey," in *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 698–708, 2018.

[118] P. Nelson, "Information and consumer behavior," *Journal of political economy*, vol. 78, no. 2, pp. 311–329, 1970.

[119] L. Huang, C.-H. Tan, W. Ke, and K.-K. Wei, "Do we order product review information display? how?," *Information & management*, vol. 51, no. 7, pp. 883–894, 2014.

[120] R. L. Thorndike, "Who belongs in the family?," *Psychometrika*, vol. 18, no. 4, pp. 267–276, 1953.

[121] Y. Zhu, R. Kiros, R. Zemel, R. Salakhutdinov, R. Urtasun, A. Torralba, and S. Fidler, "Aligning books and movies: Towards story-like visual explanations by watching movies and reading books," in *Proceedings of the IEEE international conference on computer vision*, pp. 19–27, 2015.

[122] R. He and J. McAuley, "Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering," in *Proceedings of the 25th International Conference on World Wide Web*, pp. 507–517, 2016.

[123] V. Satopaa, J. Albrecht, D. Irwin, and B. Raghavan, "Finding a" kneedle" in a haystack: Detecting knee points in system behavior," in *2011 31st international conference on distributed computing systems workshops*, pp. 166–171, 2011.

[124] K. Krippendorff, "Computing krippendorff's alpha-reliability," *Annenberg School for Communication Departmental Papers: Philadelphia*, 2011.

[125] M. Kovacs, "Customer review informativeness dataset," 2020. Mendeley Data, V1, `doi:10.17632/r286xxc7hz.1`.

[126] T. G. Dietterich, "Approximate statistical tests for comparing supervised classification learning algorithms," *Neural computation*, vol. 10, no. 7, pp. 1895–1923, 1998.

[127] Q. McNemar, "Note on the sampling error of the difference between correlated proportions or percentages," *Psychometrika*, vol. 12, no. 2, pp. 153–157, 1947.

[128] C. Nadeau and Y. Bengio, "Inference for the generalization error," *Machine Learning*, vol. 52, no. 3, pp. 239–281, 2003.

[129] A. Graves, S. Fernández, and J. Schmidhuber, "Bidirectional lstm networks for improved phoneme classification and recognition," in *International Conference on Artificial Neural Networks*, pp. 799–804, 2005.

[130] H. Takamura, T. Inui, and M. Okumura, "Extracting semantic orientations of words using spin model," in *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL' 05)*, pp. 133–140, 2005.

[131] A. H. Ribeiro, K. Tiels, L. A. Aguirre, and T. B. Schön, "The trade-off between long-term memory and smoothness for recurrent networks," *CoRR*, vol. abs/1906.08482, 2019.

[132] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[133] P. Racherla and W. Friske, "Perceived 'usefulness' of online consumer reviews: An exploratory investigation across three services categories," *Electronic Commerce Research and Applications*, vol. 11, no. 6, pp. 548–559, 2012.