ScuDo
Scuola di Dottorato ~ Doctoral School
WHAT YOU ARE, TAKES YOU FAR

Doctoral Dissertation
Doctoral Program in Computer and Control Engineering (30^{th}cycle)

Text-based Sentiment Analysis and Music Emotion Recognition

By

Erion Çano

******

Supervisor(s):
Prof. Maurizio Morisio

Doctoral Examination Committee:
Prof. Bart Dhoedt, *Referee*, Ghent University, Belgium
Prof. Loïc Barrault, *Referee*, Le Mans University, France
Prof. Rosa Meo, *External Member*, University of Turin, Italy
Prof. Fulvio Corno, *Internal Member*, Politecnico di Torino, Italy
Prof. Marco Torchiano, *Internal Member*, Politecnico di Torino, Italy

Politecnico di Torino
2018

Declaration

I hereby declare that the contents and organization of this dissertation constitute my own original work and does not compromise in any way the rights of third parties, including those relating to the security of personal data.

Erion Çano
2018

To my parents for their unconditional love and support

Acknowledgements

This thesis ends a very important journey of my life and is a result of hard work, profound commitment as well as support from many people I wish to thank. My deepest gratitude goes to my parents who have loved and supported me unconditionally during all these years of my long education. I am thankful to Politecnico di Torino for believing in me, for their generous support and comfortable working environment. Special thanks go to my supervisor, Prof. Maurizio Morisio for his continuous guidance throughout my research work. I had the opportunity to learn a lot from his long experience. Furthermore, the freedom he gave me in conducting research strengthened my self-confidence and motivation.

I could not forget to express my gratitude to my SoftEng group colleagues, Prof. Marco Torchiano, Luca Ardito, Oscar Rodriguez, Cristhian Figueroa, Iacopo Vagliano, Mohammad Rashid, Riccardo Coppola, and Diego Monti, for their valuable suggestions and collaboration. I am also highly thankful to Ihtesham Ul Islam, Andrea Martina, Amirhosein Toosi, Francesco Strada, Alysson Dos Santos as well as the other LAB 1 colleagues and friends for the multicultural and pleasant atmosphere they created and shared during these years.

Sincere thanks go to TIM (formerly Telecom Italia) for the financial support of my research. I highly appreciate the collaborative and creative work I conducted within their JOL MobiLab, in cooperation with wonderful people like Marco Marengo, Lucia Longo, Eleonora Gargiulo, Mirko Rinaldini, Luisa Rocca, Lucia Laferla and Martina Francesconi. I finally thank the HPC@POLITO team for providing their high-performance computing infrastructure which fulfilled the heavy computational requirements of my experimental activity.

Abstract

Nowadays, with the expansion of social media, large amounts of user-generated texts like tweets, blog posts or product reviews are shared online. Sentiment polarity analysis of such texts has become highly attractive and is utilized in recommender systems, market predictions, business intelligence and more. We also witness deep learning techniques becoming top performers on those types of tasks. There are however several problems that need to be solved for efficient use of deep neural networks on text mining and text polarity analysis.

First of all, deep neural networks are data hungry. They need to be fed with datasets that are big in size, cleaned and preprocessed as well as properly labeled. Second, the modern natural language processing concept of word embeddings as a dense and distributed text feature representation solves sparsity and dimensionality problems of the traditional bag-of-words model. Still, there are various uncertainties regarding the use of word vectors: should they be generated from the same dataset that is used to train the model or it is better to source them from big and popular collections that work as generic text feature representations? Third, it is not easy for practitioners to find a simple and highly effective deep learning setup for various document lengths and types. Recurrent neural networks are weak with longer texts and optimal convolution-pooling combinations are not easily conceived. It is thus convenient to have generic neural network architectures that are effective and can adapt to various texts, encapsulating much of design complexity.

This thesis addresses the above problems to provide methodological and practical insights for utilizing neural networks on sentiment analysis of texts and achieving state of the art results. Regarding the first problem, the effectiveness of various crowdsourcing alternatives is explored and two medium-sized and emotion-labeled song datasets are created utilizing social tags. One of the research interests of Telecom Italia was the exploration of relations between music emotional stimulation and

driving style. Consequently, a context-aware music recommender system that aims to enhance driving comfort and safety was also designed. To address the second problem, a series of experiments with large text collections of various contents and domains were conducted. Word embeddings of different parameters were exercised and results revealed that their quality is influenced (mostly but not only) by the size of texts they were created from. When working with small text datasets, it is thus important to source word features from popular and generic word embedding collections. Regarding the third problem, a series of experiments involving convolutional and max-pooling neural layers were conducted. Various patterns relating text properties and network parameters with optimal classification accuracy were observed. Combining convolutions of words, bigrams, and trigrams with regional max-pooling layers in a couple of stacks produced the best results. The derived architecture achieves competitive performance on sentiment polarity analysis of movie, business and product reviews.

Given that labeled data are becoming the bottleneck of the current deep learning systems, a future research direction could be the exploration of various data programming possibilities for constructing even bigger labeled datasets. Investigation of feature-level or decision-level ensemble techniques in the context of deep neural networks could also be fruitful. Different feature types do usually represent complementary characteristics of data. Combining word embedding and traditional text features or utilizing recurrent networks on document splits and then aggregating the predictions could further increase prediction accuracy of such models.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Social networks and the Internet have increased our communication capabilities to the point that everyone with a connected device can be read or heard worldwide. Sharing opinions about politics, sport, brands or products in social networks is now a cultural trend. Consumers are especially inclined to share or consult online opinions about products they have bought or are willing to buy. On the other hand, companies are motivated to find innovative ways for getting benefit using opinion data that are daily posted online. In fact, marketing statistics suggest that 81% of shoppers conduct online research before making big purchases and 92% of marketers consider social media to be important for their business.[1]

Sentiment Analysis (SA) is considered as the process of computationally identifying and categorizing opinions expressed in a piece of text, especially in order to determine whether the writer's attitude towards a particular topic or product is *positive*, *negative*, or *neutral*.[2] In other words, it is a set of techniques and practices for automatic identification of sentiment polarity in texts. From the business perspective, analyzing opinions of clients is essential for several reasons, such as improve product or service quality, measure and improve the success of marketing campaigns, determine or adjust marketing strategy, etc. Sentiment analysis results are in fact widely used as a component of recommendation engines that generate advertisements for online users in many websites.

[1] https://www.hubspot.com/marketing-statistics
[2] https://en.oxforddictionaries.com/definition/sentiment_analysis

Back in the early 2000s, text polarity was mostly analyzed starting from words or phrases and going up to entire document. Creating lexicons of affect terms and using them to infer word polarity was a common practice [1, 2]. Later on, the popularization of the web, social networks, and cloud services lured users to give more and more feedback on a daily basis. User opinions about different types of items come in various ways. Besides text posts or comments, social tags have become very popular as well, especially as an instrument for performing sentiment analysis of songs. They are basically single word descriptors like "rock", "sweet" or "awsome" that express users' opinion for a certain song or another object type. Tags of *Last.fm* (an online radio station and web platform for music listening with open API) were frequently used in various Music Emotion Recognition (MER) research papers [3, 4]. At the same time, the rising popularity of microblogs that are rich in user opinions, motivated many researchers to create datasets of emotionally labeled texts. As a result, focus gradually shifted from unsupervised to supervised learning methods for performing SA. This trend was also propelled by the development of highly effective machine or deep learning techniques and the popularization of high-level libraries or frameworks for using them.

Neural networks offer wide applicability and are able to automate feature extraction and selection in many domains and tasks. The release of powerful graphics processing units for scientific computing enhanced quick training and usability of deep neural networks for heavy tasks such as language modeling for predicting word combinations. One of the first neural language models was proposed by Bengio et *al.* in 2003 [5]. Simpler and easier to train models were proposed in the following years. Huge and increasing amounts of text started to be used for training them and generating word feature representations also known as word vectors or embeddings. These word features offer significant advantages over the traditional bag-of-words text representation such as density, reduced dimensionality, preserved semantic relations between words, etc. Word embeddings and neural networks started to be used as classification features on several text mining tasks.

This thesis addresses current problems in sentiment analysis of music and different types of texts, such as movie reviews, product reviews, etc. A particular interest of Telecom Italia was the exploration of social tags and other crowdsourcing alternatives for creating emotionally labeled song datasets of considerable size. Those datasets form the basis for music emotion recognition, music recommender systems, and other relevant applications. Quality assessment and effectiveness of word em-

beddings on sentiment analysis tasks was another concern. Utilizing different types of neural network layers for text feature generation, feature selection, and sentiment prediction is not straightforward. To reach the desired performance, a high number and type of hyperparameters need to be tuned. The last part of the thesis addresses the possibility of encapsulating such complexity in a network architecture that can be used to quickly prototype accurate models for sentiment analysis. In brief, the research questions we pose are the following:

RQ1 *Are affect tags of songs and other crowdsourcing alternatives applicable to music emotion recognition or music emotion dataset creation?*

RQ2 *What effect do training method, corpus size, corpus topic, and other characteristics have in the performance of generated word vectors used in sentiment analysis tasks?*

RQ3 *How could design and hyperparameter tuning complexity of the neural network models for sentiment analysis be reduced?*

Regarding RQ1, *Last.fm* user tags of about one million recent songs are crawled and analyzed. Our observations, same as those of previous related works such as [6] show that indeed, social tag emotional spaces are in consonance with those of psychologists (e.g., the popular model of Russell [7]). Tags are thus a viable means for exploring emotions in music. We also create two relatively big datasets of song emotions which are publicly released for research use. Other crowdsourcing mechanisms such as online games or services and Mechanical Turk are explored and found useful for gathering labels about songs or similar types of items.

Considering RQ2, several experiments with song lyrics and movie reviews were conducted, using Glove and Skip-Gram methods for generating word vectors of texts. We noticed that corpus size and its vocabulary richness have a significant impact on the quality of word vectors that are generated. Word vectors trained on big text bundles tend to perform better on sentiment analysis of song lyrics and movie reviews. Thematic relevance between training corpus and task seems to have a lower importance on performance and should be further assessed in a more comprehensive experimental framework. We also observed that Glove and Skip-Gram training methods have slightly different performance patterns. The former is a slightly better player on big text bundles when used to analyze song lyrics whereas the latter gives best results on small or average text sources used on movie reviews.

Responding to RQ3, various experiments with convolution and pooling layers were conducted. Convolutions of different kernels seem to be highly capable of capturing *n*-gram word vector combinations of texts. Furthermore, pooling layers (e.g., max-pooling) are very effective for selecting the most salient word feature maps for sentiment classification. Combining generic word vectors trained on billion-sized text bundles with convolution and pooling stacks produced state-of-the-art results in sentiment analysis of song lyrics, movie reviews or other types of texts. This thesis proposes a neural network architecture of convolutional and max-pooling neural layers that can be used as a template for rapid prototyping of highly efficient sentiment analysis models. It adapts to various text lengths and dataset sizes with little need for hyper-parameter adaptions.

The thesis is structured as follows. Chapter 2 introduces different concepts about sentiment analysis, summarizes milestone achievements in artificial intelligence and presents important facts which highlight the current shift towards data-driven supervised learning techniques. In Chapter 3 psychological viewpoints about emotions in music and their representation using simplification models are first summarized. Later on, crowdsourcing possibilities for harvesting emotional labels of songs are discussed. Finally, *Last.fm* user tags are utilized for constructing two public and big datasets of song emotions. Chapter 4 introduces recommender systems as important information filtering tools, explores hybrid and context-based recommenders and presents the design of a recommender system of songs in the context of car driving. Influence of various text characteristics on the quality of generated word vectors is explored and presented in Chapter 5 in details. Chapter 6 presents the results of various experiments with convolution and max-pooling neural layers on sentiment analysis of song lyrics, movie reviews and other types of texts. NgramCNN neural network architecture variants and their performance scores are discussed in Chapter 7. Finally, Chapter 8 presents the main contributions, derived conclusions, and possible future research directions.

Chapter 2

Background

"In god we trust, all others must bring data."
– W. Edwards Deming, statistician

In the last fifteen years, enormous amounts of user opinion texts have been posted on the Web, especially on social media websites. All these data that keep growing exponentially have created incentives for developing automatic methods and tools that are able to analyze user opinions about different brands, products, services or other entities. *Sentiment Analysis* that is also known as *Opinion Mining* is a collection of methods, techniques, and practices used for automatic analysis of opinions, sentiments or attitudes about those entities. Business intelligence, market predictions, customer care and online marketing are some of its most popular and common application realms. Sentiment analysis is also highly interrelated with other fields such as *Natural Language Processing* or *Artificial Intelligence* that are recently enjoying rapid progress as well, strongly driven by the online data revolution that is happening. In particular, statistical language models, lexicons, and text representation methods, together with machine learning and neural networks are highly important for the correct and intelligent interpretation of opinions.

In this chapter, we first present some basic concepts and definitions related to sentiment analysis. Later in Section 2.2, we discuss the current proliferation of data in the Web as a result of the advancing digitalization process. Section 2.3 provides an overview on some of the most common techniques and successful applications of

sentiment analysis. In Section 2.4, important milestone and transformational developments in artificial intelligence, machine learning, and natural language processing are outlined. Finally, Section 2.5 briefly presents various strategies that can be used to generate large quantities of labeled data which are essential for training effective prediction models.

2.1 Sentiment Analysis Concepts and Definitions

Sentiment Analysis (SA) can be considered as the computational examination of sentiments, opinions, emotions, and attitude expressed in text units towards an entity [8]. It is about identifying, extracting and classifying opinions and attitudes about various issues. SA represents a really wide and flourishing research realm today. A good indicator of this fact is obviously the vast collection of terms it is referenced by. Various designations like Sentiment Analysis, Opinion Extraction, Opinion Mining, Affect Analysis, Emotion Analysis or Subjectivity Analysis are interchangeably used in research publications to denote similar tasks. According to [9], *Subjectivity Analysis* is the earliest term that was first used in late 90s. It insinuates the recognition of opinion-oriented language in texts and its separation from the objective language. Later in 2001, we had the first research papers using the term *Sentiment Analysis* when referring to the automatic sentiment polarity analysis of subjective texts. Shortly after in 2003, the term *Opinion Mining* first appeared in a WWW conference publication.

Opinion is obviously the central concept in SA. Liu in [10] provides a formalized definition about it. According to him, an opinion is a quintuple $(e_i, a_{ij}, s_{ijkl}, h_k, t_l)$. Here e_i is the name of an entity and a_{ij} is an aspect (or component) of e_i. Together e_i and a_{ij} represent the target (g) of the opinion. Furthermore, s_{ijkl} is the sentiment on aspect a_{ij}, h_k is the opinion holder and t_l is the time when the opinion was expressed. Opinion sentiment s_{ijkl} is *positive*, *negative*, or *neutral*. It might also be a numeric rating that expresses the intensity of the sentiment (e.g., 1 – 5 stars rating). *Sentiment classification* task is about determining what polarity does opinion sentiment s_{ijkl} have on aspect a_{ij} (or on target g). Another important concept in SA is that of *opinion words* that are terms commonly used to express *positive* (e.g., “excellent”, “wonderful”, “great”) or *negative* (e.g., “poor”, “terrible”, “awful”) opinions. Lists

of opinion words are usually created as a means for solving sentiment classification tasks and form a *sentiment lexicon*.

In the case when a text document contains opinions of only one opinion holder about a single entity and the different aspects of that entity are irrelevant, the document is analyzed entirely. In this case the task is known as *document-level sentiment classification* [11]. This task represents the main focus of this thesis. Analysis of user reviews about online products typically falls into this category. When sentiment classification is applied to single subjective sentences the task is usually called *sentence-level sentiment classification*. Here we assume that the sentence holds only one opinion from a single opinion holder. In other words, the sentence must be simple like "Battery life of this camera is great". This task is also highly interrelated with *subjectivity classification* which determines if a sentence is subjective or objective. Document-level and sentence-level sentiment classifications do not exactly reveal what one likes or not about an entity (e.g., battery life). In complex sentences like "This camera is great and I like the picture quality", the user provides opinions about different entity aspects. Analyzing such sentences by first identifying each aspect or target is called *aspect-level sentiment classification*. Finally, it is important to mention that SA is an NLP problem that is restricted only to some aspects of word, sentence or document semantics. Nevertheless, a holistic understanding of the application domain and problems is usually an essential prerequisite. As we will see in the following chapters, text preprocessing and cleaning part is an important step in every SA solution.

2.2 Exponential Online Data Explosion

Technological progress keeps making computing gadgets faster, lighter and cheaper at the point that more and more people can afford them. One of the biggest consequences of this is the digitalization of our lives; today we use digital mail, books, photos, music, documents, maps, etc. Everyone of us relies on online learning, news or advertisements. We also order products, tickets or food and pays bills using digital devices connected to the Internet. It is interesting to note that most of the free content we daily consume online is also freely created and made available by us. The paradigm of today is "put it online, get it online." The consequences of this trend have been pointed out paradoxically in the appealing quote of Tom Goodwin:

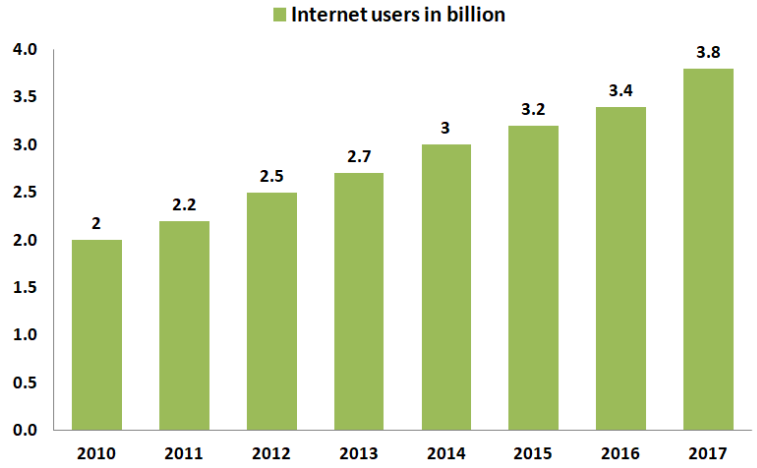

Fig. 2.1 Tendency of Internet users in the last years

"Uber, the world's largest taxi company owns no vehicles, Facebook the world's most popular media owner creates no content, Alibaba, the most valuable retailer has no inventory and Airbnb the world's largest accommodation provider owns no real estate. Something interesting is happening".[1]

In fact, Internet statistics reveal us that user-generated content in sites like Wikipedia, Twitter, Facebook, Youtube or Instagram has indeed been growing exponentially in the last years.[2] According to those statistics, as of December 2017, there are roughly 3.8 billion Internet users (the trend in Figure 2.1), almost 1.8 billion websites (the trend in Figure 2.2) and 43 million Wikipedia pages. Furthermore, every hour we have 182,000 TB of Internet traffic, 9.5 billion emails sent and 232 million Google searches served. There are also 28 million tweets sent (the trend in Figure 2.3), three million photos uploaded on Instagram, 200 million posts shared on Facebook and 30 thousand hours of video playback uploaded on Youtube.

Users also contribute with valuable content in the form of reviews on sites like Amazon, TripAdvisor, and Yelp. That content has become highly important in the decision making of consumers. People rely on informal reviews to decide where to eat, what gadgets to buy, where to sleep etc. According to statistics,[3] 92% of consumers today read online reviews and only 12% are prepared to read more than ten of them. Regarding persuasion effect of reviews, 40% of the subjects form an

[1] Tom Goodwin in http://www.techcrunch.com, March 2015
[2] http://www.internetlivestats.com
[3] https://www.vendasta.com/blog/50-stats-you-need-to-know-about-online-reviews

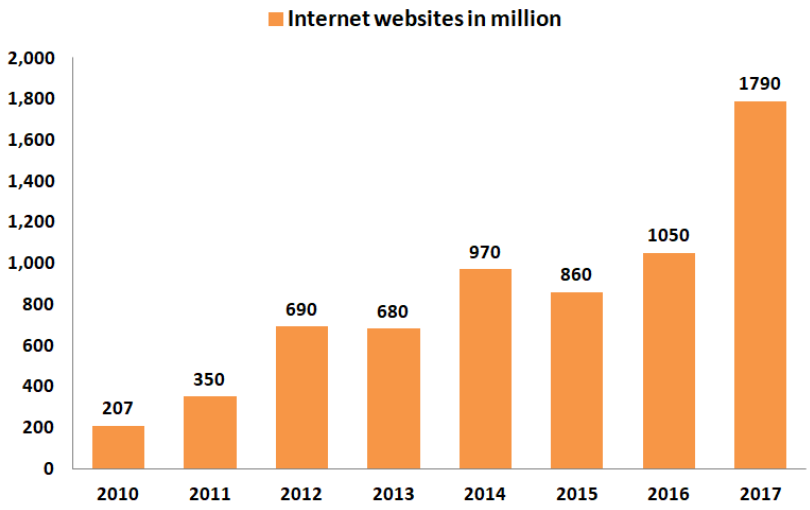

Fig. 2.2 Tendency of Internet websites in the last years

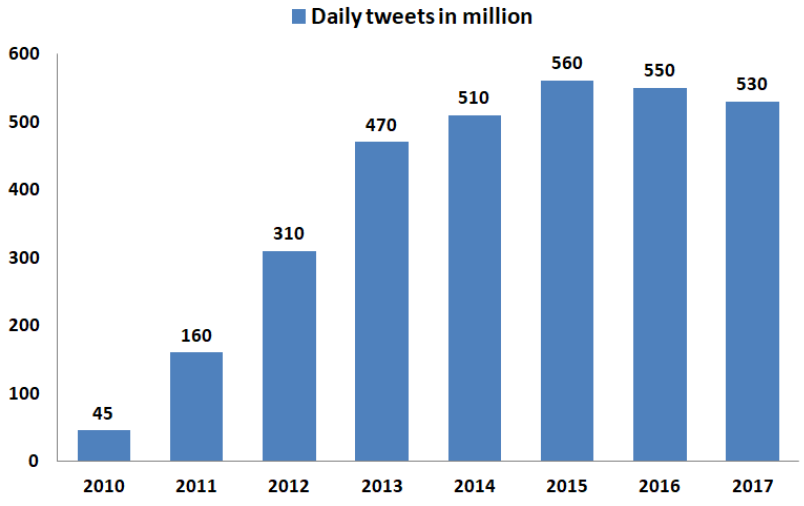

Fig. 2.3 Tendency of daily tweets in the last years

opinion by reading just one to three reviews and 73% of them form an opinion by reading up to six reviews. Also, one up to three bad online reviews would be enough to deter most of (67%) shoppers from buying a product or service. There are of course credibility problems created by fake reviews (e.g., from AI trained bots) that come out once in a while. In fact, statistics reveal that 95% of consumers doubt faked reviews when they do not see bad scores, or that 30% of them assume online reviews are tricked if there is no negative feedback at all. Despite the credibility issues, all that free online content that is growing exponentially has greatly motivated the invention and utilization of novel SA solutions. In the following section, basic methods for solving the sentiment classification problem are discussed.

2.3 Sentiment Analysis Techniques and Applications

As mentioned above, sentiment classification problem is central to SA. This problem has been addressed using a high variety of techniques. Survey works like [8] or [12] categorize those techniques as *machine learning*, *lexicon-based* or *hybrid*. The first to consider sentiment polarity analysis as a type of document classification were Pang and Lee in [13]. They exercised naïve Bayes, Maximum Entropy and Support Vector Machine algorithms on movie reviews they had collected and prepared. Since that time, researchers have extensively explored all kinds of supervised learning algorithms like Bayesian networks or k-nearest neighbors, linear classifiers like logistic regression, decision trees, random forests, rule-based classifiers or even the most recent deep neural networks. In the early 2000s, there was a tendency of trying unsupervised or hybrid approaches to overcome the need for labeled data. For example, authors of [14] implemented a hybrid approach by first dividing each document into sentences which are classified as *positive* or *negative* using keyword lists of each category. Labeled sentences are then used as data for training traditional supervised algorithms and making the overall prediction.

A popular unsupervised method for review polarity classification was proposed by Turney in [15]. First, a POS tag is assigned to each word of the reviews to recognize adverb or adjective phrases. Then semantic orientation of words or phrases is computed using PMI-IR algorithm. Pointwise Mutual Information (PMI) between

two words w_1 and w_2 is given by Equation 2.1:

$$PMI(w_1, w_2) = log_2(p(w_1 \, and \, w_2)/p(w_1)p(w_2)) \tag{2.1}$$

Here $p(w_1 \, and \, w_2)$ is the probability of the two words appearing near each other. The Semantic Orientation (SO) of each word or phrase is thus computed using Equation 2.2 below:

$$SO(phrase) = PMI(phrase, \text{“}excellent\text{”}) - PMI(phrase, \text{“}poor\text{”}) \tag{2.2}$$

This way average semantic orientation is computed for every review. Values greater than zero are associated to *positive* reviews whereas negative semantic orientation values are associated to *negative* reviews. Experimental results of Turney revealed accuracy scores ranging from 65% to 84%, depending on the types of items analyzed. In the following years, unsupervised sentiment analysis has been little explored, in contrast with supervised techniques that gained exceptional popularity motivated by the rapid growth of online user reviews and posts. Lexicon-based approaches extensively utilize opinion words or phrases found in texts that express *positive* or *negative* polarity. The first step here is filling the list of words of each polarity with a couple of seed words. Next, the lists are extended by searching for synonyms or antonyms in dictionaries. Another common practice is to search on large text corpora for co-occurrence of seed words with other opinion words. Similar approaches analyze texts based on word statistics or formal ontologies that capture semantic associations between concepts.

Regarding applications of SA, there seems to be a well-established tradition in macroeconomic predictions about markets, businesses, brands or products [12]. There are many studies that search for correlations between positivity of texts in social media and financial data records, using the former to model and predict the latter. Companies are also highly motivated to investigate the reasons why customers accept or reject a new product. In this context, understanding customer needs and preferences through opinion mining is highly important and most big companies incorporate it as part of their mission. Online product reviews are becoming important to the point that various websites are soliciting user feedback about restaurants, hotels or other commodities. A common but less obvious application of SA is its use as a subcomponent in other quickly growing technologies like recommender systems. One can easily imagine utilizing positivity of user reviews about certain items to

create clusters of users that might become potential clients. The same information can be utilized to augment and improve item profiling as well. Other potential applications of SA include question answering systems, stock market forecasting, political surveys, criminology etc.

2.4 Machine and Deep Learning Expansion

2.4.1 Early Mileston Developments

The recent success of machine learning and especially deep neural networks in various industries is producing fascinating results. Andrew Ng who is considered as one of the pioneers in the field has predicted a revolutionary effect of deep learning in today's society, similar to that of electricity about 100 years ago.[4] The first milestone was probably the neuron prototype designed back in 1943 by McCulloch and Pitts [16]. They combined a weighted linear function with what they called "threshold logic" to model the working of the human brain. Inside the philosophical debate of whether a machine can think or not, Alan Turing evaded the question by proposing "The Imitation Game" (known as Turing Test), a series of criteria for assessing if a machine could be enough intelligent to fool a person into thinking it is actually a human [17]. While the debate still goes on, the pragmatic need to sort out humans from computers in the Web brought out automatic and public versions of Turing Test commonly known as CAPTCHA.[5]

The progress in AI would not become visible to the general public until Frank Rosenblatt invented the first artificial neural network called "Perceptron" back in 1957 [18]. Inspired by the human visual system and designed for image recognition, this newborn machine created a lot of enthusiasm and resurrected the old debate about the limits of AI. Technically a "Perceptron" is a linear function of input values, limited in representing linear decision boundaries for learning operations like AND, OR and NOT, but not XOR. The first to make this observation were Minski and Papert in 1969 who also proved that is was theoretically impossible for the "Perceptron" to learn XOR function [19]. Consequently, the so-called first AI winter came on and progress was sluggish until the beginning of the 1980s. The most important

[4] http://fortune.com/2016/10/05/ai-artificial-intelligence-deep-learning-employers/
[5] Completely Automated Public Turing test to tell Computers and Humans Apart

development that followed was a paper from David Rumelhart, Geoff Hinton and Ronald Williams, entitled "Learning representations by back-propagation errors" [20]. In that paper, they showed the simple procedure (Backprop algorithm) that could be used to train neural networks with many hidden layers which unlike the "Perceptron" could learn nonlinear functions. Almost at the same time, Yann LeCun made the first practical demonstration of a convolutional neural network trained with back-propagation for recognizing handwritten digits. Unfortunately, neural networks could still not scale to larger problems and the broken expectations together with the collapse of Lisp machine market[6] resulted in a second winter (during the 1990s).

2.4.2 From Logic to Data

While neural networks were left untouched, there was still time for other developments. Ensemble learning came out as a novel branch of machine learning techniques that utilize multiple learners to solve the same problem with higher predictive accuracy. The various techniques usually differ in how they select the training data and the way they aggregate the decisions of each classifier. Bootstrap aggregating known as *Bagging* trains ensemble models on randomly picked subsets of the training set and makes them vote with equal weight. Random forest, for example, is a combination of decision trees with bagging that provides a robust classification on problems with huge numbers of features. *Boosting* on the other hand, incrementally learns various classifiers adding them to a final model. When each classifier is added, the data are reweighted giving importance to training instances that previous learners misclassified. In works like [21], Adaboost which is the most common boosting implementation is reported to yield better improvements than bagging.

Roughly at the same time, Support Vector Machine (SVM) classifier was invented and proven to be highly effective [22]. It maps the input vectors into a high dimensional feature space and constructs a linear decision surface to ensure high generalization ability and global optimality. Furthermore, the notion of soft margins extends its applicability to non-separable training data. A few years later, the kernel trick generalized usability of SVMs beyond linear functions by transforming the input space into a feature space [23]. All these advances in machine learning helped to shift the focus from logic-driven, deductive solutions to data-driven, statistical

[6] https://danluu.com/symbolics-lisp-machines/

ones. The power of computers started to be used for analyzing big quantities of data and inferring conclusions from the obtained results. Moreover, popularization of the Web helped to create bigger experimental datasets and boosted research in disciplines like *information retrieval*, *text analysis*, *text mining*, etc. Bag-Of-Words (BOW) representation was commonly used for representing texts in many studies. It was typically combined with various classification algorithms like k-nearest neighbors, naïve Bayes, decision trees or SVM. In [24] we find one of the first empirical comparisons of such algorithms in text categorization tasks. According to its experimental results, SVM appears the dominant classification algorithm in text categorization tasks. That finding emphasizes the fact that strong points of SVM match text document characteristics such as high dimensionality of input space (big vocabulary size), the few irrelevant features (words) and the sparsity of document-term matrix.

2.4.3 Current Hype of Deep Learning

The exponential increase of user-generated content in the emerging Web 2.0 social media of the early 2000s contributed to the proliferation of even bigger datasets of labeled images or texts. ImageNet,[7] a huge dataset of millions of labeled images from different categories was created and made available to support research in computer vision and image recognition. At the same time, faster and cheaper hardware (especially GPUs for generic computations) became available and could be quickly and economically deployed in the form of cloud services. This extra supply in computing infrastructure stimulated proliferation of startups that created intelligent and innovative services. Interest in neural network research resurrected and was rebranded as Deep Learning (DL). Groundbreaking network designs like Long Short-Term Memory (LSTM) which was proposed in 1997 and improved in 2000 came out. Generative neural networks that can work in both supervised and unsupervised contexts like *deep belief networks* or *deep Boltzmann machines* followed in 2006 and 2009. More recent designs like *generative adversarial networks* presented in 2014, significantly reduced computational costs. Furthermore, novel regularization techniques like Dropout or Batch Normalization helped for the mitigation of problems like overfitting and accelerated training by reducing covariate shift. That kind of problems had hindered usability of deep neural architectures for many years. Consequently, improved and deeper architectures of convolutional

[7] http://image-net.org

neural networks like *VGG-19*,[8] *AlexNet*,[9] or *Inception*[10] were developed and made available for public use. Those big models have won many international image recognition competitions like ILSVRC.[11] Together with open source software platforms like *Torch* (2002), *Theano* (2008) or *Tensorflow* (2015), they have democratized AI creating the hype of today.

2.4.4 From n-grams to Neural Language Models

An important contribution of neural networks in NLP has to do with the transition from the traditional statistical language models to the more advanced neural ones. A language model is a probability function that is able to describe important statistical characteristics of word sequence distributions in a natural language. It typically enables us to make predictions about the next (or previous) word appearing in a sequence of given words. This capability is successfully implemented in various applications of natural language processing like speech recognition, automatic language translation, spell checking, etc. In a n-gram model for example, the probability $P(w_1, \ \dots \ , w_m)$ of word sequence $w_1, \ \dots \ , w_m$ is computed as:

$$P(w_1, \ \dots \ , w_m) = \prod_{i=1}^{m} P(w_i | w_1, \ \dots \ , w_{i-1}) \approx \prod_{i=1}^{m} P(w_i | w_{i-(n-1)}, \ \dots \ , w_{i-1}) \quad (2.3)$$

As we can see, the above approximation assumes that the probability of observing the i^{th} word w_i in context of the proceeding $i-1$ words can be calculated as the probability of observing it in the shorter context history of the proceeding $n-1$ words. This property (known as *Markov* or *memoryless* property) is a characteristic of those stochastic processes in which the conditional probability distribution of future states depends upon the present state only. The most common n-gram models are *unigram* ($n = 1$), *bigram* ($n = 2$) and *trigram* ($n = 3$). The conditional probability of those models is obtained from the frequency of n-gram counts[12] as:

$$P(w_i | w_{i-(n-1)}, \ \dots \ , w_{i-1}) = \frac{count(w_{i-(n-1)}, \ \dots \ , w_{i-1}, w_i)}{count(w_{i-(n-1)}, \ \dots \ , w_{i-1})} \quad (2.4)$$

[8] http://www.robots.ox.ac.uk/~vgg/research/very_deep/
[9] https://en.wikipedia.org/wiki/AlexNet
[10] https://github.com/google/inception
[11] http://www.image-net.org/challenges/LSVRC
[12] In practice, smoothing algorithms are used to give some probability weight to unseen n-grams.

In contrast, neural language models utilize continuous space word embeddings (also known as distributed word feature vectors) to make language predictions. One of the motivations for developing such models was the need to fight the *course of dimensionality* problem that results when models are trained on large text collections with a big vocabulary (number of unique words) size V. The total number of possible word sequences increases exponentially with V causing severe data sparsity. Neural language models avoid the *course of dimensionality* problem by representing text words in a distributed way. They are conceived as probabilistic classifiers that learn a probability distribution over vocabulary V given some linguistic context that is typically a fixed-size window of previous k words. Neural language models are thus trained to predict:

$$P(w_m|w_{m-k},\ \ldots\ ,w_{m-1})\ \ \forall m\ \in\ V$$

Bengio *et al.* in [5] presented one of the first attempts to train and evaluate a distributed representation of words. Authors proposed to learn the probability function $P(w_m|w_1,\ \ldots\ ,w_{m-1})$ by decomposing it in the following two parts:

1. A mapping C from any element i of V to a real vector $C(i) \in \mathbb{R}^d$. This basically represents the distributed feature vectors (word embeddings) of d dimensions associated with each word of the vocabulary.

2. A function g that maps an input sequence of distributed context word vectors $C(w_{t-k+1},\ \ldots\ ,C(w_{m-1}))$ to a probability over words in V for next world w_m. Function g creates as output a vector whose i-th element estimates the probability $P(w_m = i|w_1,\ \ldots\ ,w_{m-1})$.

This way, according to their approach:

$$P(w_m = i|w_1,\ \ldots\ ,\ w_{m-1}) = g(i, C(w_m - 1),\ \ldots\ ,\ C(w_m - n + 1)) \tag{2.5}$$

As a result, the number of free parameters scales only linearly (not exponentially as in n-gram models) with vocabulary size V. Authors implement their solution by means of a Multi-Layer Perceptron (MLP) network and compare against a state-of-the-art trigram model. They report 10 – 24 percent higher performance at the cost of a significantly longer training time. Yet computation requirements scale linearly with the number of conditioning variables. Their work paved a path of significant improvements in language models, transiting from discrete and sparse

representations (n-grams) to continuous and dense feature vectors that are highly compliant with the emerging neural network technologies. In the epoch of faster GPUs and continuously improving neural architectures, many other researchers followed the same path, proposing neural models with better generalization abilities or improved computational efficiency.

2.4.5 Word Embeddings for Text Representation

One of the first popular methods for generating word embeddings, known as C&W (for Collobert and Weston) was presented in [25]. Authors make use of a CNN architecture to generate word embeddings from a Wikipedia text corpus. They report significant improvements in various NLP tasks such as part-of-speech tagging, named entity recognition, semantic role-labeling etc. A few years later, Continuous Bag-of-Words (CBOW), and Skip-Gram were presented in [26]. Authors utilize shallow networks for easier training of models that predict a word based on the context (CBOW) or predict the context words of a given word (Skip-Gram). Both methods are considerably improved (in terms of training time) in [27] where negative sampling and subsampling of frequent words are also presented. At this point, interesting properties of word embeddings like the exhibition of syntactic and semantic regularities between words became more evident. For example, word analogies (in this case regarding gender) appear in algebraic regularities of the form:

$$v(king) - v(male) \approx v(queen) - v(female)$$

where v represents the learned d-dimensional feature vector of the corresponding word. Datasets with word analogy questions were created and started to be used for evaluating the quality of word embeddings.

Glove introduced in [28] represents another recent and popular method for generating word embeddings. Authors train texts using word-word co-occurrence counts and global corpus statistics. At the same time, they preserve the linear structure of CBOW and Skip-Gram. According to authors' results, Glove scales very well on huge text collections and outruns similar methods on many tasks, including word analogies. In Section 5.3 we present and discuss our own experiments, comparing Glove and Skip-Gram on different SA tasks. Regarding applicability in NLP or SA, BOW representation remains popular despite the data sparsity problem

it suffers from. This popularity is mainly because of its simplicity and performance, especially when used in combination with SVM classifiers on texts of relatively small vocabularies. Word embeddings on the other hand, while still being more computation intensive have better generalization abilities and are immune to high dimensionality problem even on texts of large vocabularies. The tendency towards bigger training datasets and continuous speed-ups of neural networks is expected to favor distributed text representations via word embeddings.

2.4.6 Deep Learning Applications and Achievements

The growing applicability of deep neural networks, besides shadowing earlier machine learning techniques in traditional tasks, also opened up new perspectives in more difficult scenarios. Speech recognition and automatic machine translation are two application domains where LSTMs are excelling. Healthcare is probably the most important domain for humanity. Deep learning image recognition systems are already analyzing X-rays better than radiologists do. Drug discovery, melanoma screening or brain cancer detection are appealing applications as well. Genomics or gene editing is also a highly complex task that is getting advantage from deep neural networks. In the realm of cybersecurity, pattern recognition of newer viruses or network threats is being trusted to neural network models. Other applications include optimizing space mission efforts where an Italian team of scientists is currently applying neural networks [29].

At the same time, we have witnessed spectacular achievements involving humans competing against machines. In 2011 for example, Watson,[13] the supercomputer of IBM defeated the two most reputable champions of Jeopardy, a quiz show where players are introduced with knowledge indications and are expected to respond with a question. That game involves complex skills like memorization, intuition, agility, etc. Furthermore, AlphaGo program of Google beat in 2016 a professional human Go player. The program renamed AlphaZero was later generalized to play chess and other mind games as well.

The natural question that comes to our minds is what makes deep neural networks better players than traditional machine learning approaches (e.g., SVM) in so many tasks. In fact, an obvious advantage of today's neural network architec-

[13] https://www.ibm.com/watson/

tures is their ability to automatically extract and select features. This eliminates the need for hand-crafting features out of data, speeding up model creation and deployment. Convolution layers, for example, are famous for their ability to find the most representative features in images (e.g., ears, noses or eyes in face recognition). When different convolution layers are stacked one after the other, they transform features received from previous the layers into more complex and detailed features that are eventually passed to the classifier (usually the last few layers). This very successful paradigm of using deep feature extraction and selection layers combined with a simple classifier has been the basis of the image award-winning architectures mentioned in Section 2.4.3.

2.5 Hunger for Labeled Data

"Data is the new oil." We have been reading or hearing this intriguing phrase by so many information technology leaders, public speakers or even politicians in the last decade. Clive Humbly, a mathematician and data scientist is usually credited as the first one to have coined it.[14] The basic idea is to emphasize the essential role that data and information have in society. Actually, data has been around for hundreds of years. The difference today is in the quantity of data that users provide daily by utilizing free online services like Google searches, Twitter or Facebook posts and comments, etc. The other important difference is our ability to process it and extract great value from it using advanced AI techniques and technologies. This ability is giving such an immense advantage to technology giants, making economy experts discuss antitrust regulations, similar to those imposed to oil companies at the beginning of 20^{th} century.[15]

Same as internal combustion engines invented about 150 years ago, AI techniques can be considered as "prediction engines" of today. AI is "fueled" from labeled training data in the same way as combustion engines are fueled by refined oil. However, there is one point where this interesting analogy breaks. It is usually harder to find crude oil than to refine it into gas. The same thing is not true about data. It is so easy today to find it in immense quantities. Yet it is harder and expensive to label or "refine" it, producing the necessary fuel for our prediction engines. Deep

[14] http://ana.blogs.com/maestros/2006/11/data_is_the_new.html
[15] http://www.cbc.ca/news/technology/data-is-the-new-oil-1.4259677

neural networks of today are incredibly complex and utilize millions of parameters to generate their "magical" predictions. They are thus data hungry, requiring in many cases hundred thousands or millions of labeled training samples. Asking Subject Matter Experts (SME) to hand-label the required data is not feasible anymore as it would take many months and a lot of money. Labeled data have consequently become the development bottleneck for real-world AI applications. There are still some indirect or partial solutions to this problem:

Crowdsourcing services Platforms like Amazon Mechanical Turk[16] enable hiring of non-expert workers that can fulfill massive tasks. They also provide high-quality tools for a precise and efficient labeling process. Although crowd workers will certainly not perform same as SMEs, labeling quality might be acceptable for many applications.

Online applications Fancy and entertaining games or other applications usually lure many users in short online interactions. One possibility would be to obtain classification or labeling of the data by users that randomly interact with such applications.

Public datasets Obtaining labeled data from existing free datasets could be the fastest and most economical way for solving the problem. Crawling tools or abilities can also help for harvesting large quantities of data from websites. However, copyright issues may need to be carefully considered as many public data are distributed with non-commercial licenses.

Model tunning If there are enough labeled data from public datasets but their quality is suspicious, one possibility could be to tune the model on progressively better data. This way the model is first trained on the bigger quantity of public data. Next, it is retrained (and thus tuned) on smaller, professionally labeled data.

Transfer learning Sometimes there might be tons of professionally labeled data for a similar but not identical task that can be utilized. There are also tools that could be used to automatically determine which elements of the source data can be repurposed for the new task.

[16] https://www.mturk.com/

Data programming proposed in [30] is a novel and more systematic paradigm for obtaining large quantities of labeled data efficiently. It is based on higher-level supervision over unlabeled data from SMEs (e.g., in form of heuristic rules) for generating noisy training samples programmatically via labeling functions. Authors propose learning the accuracies of the labeling functions and their correlation structure in the form of a generative model for automatically denoising the generated data. They test their solution on *relation mention extraction* task applied in texts of news, genomics, pharmacogenomics and diseases, reporting very promising results. Data programming might thus become an important specialization of data science in the near future.

Chapter 3

Emotions in Music

"Some sort of emotional experience is probably the main reason behind most people's engagement with music."

– P. Juslin and J. Sloboda, *Handbook of Music and Emotion*, 2001

Music listening is a universal experience that has been highly influenced by technology. The transition from personal music player devices to online streaming platforms changed music industry significantly in the last fifteen years. These platforms boosted personalization of music listening experience by recommending music to users based on their listening history and profile. Retrieval and recommendations of music are based on title, genre, artist or other metadata, as well as on mood, an important attribute of music. These advances promoted research in areas like *Music Information Retrieval* or *Music Emotion Recognition*. The later is a form of sentiment analysis in music domain where the text is only a partial source of attributes (features). It uses analogous supervised or unsupervised methods and same constructs like lexicons.

We start this chapter by presenting some fundamental concepts that concern music and emotions. In Section 3.1.2 we introduce some popular music emotion models proposed by psychologist. Then in Section 3.1.3, various feature types used for emotion identification in songs are described. Section 3.2 presents some crowdsourcing methods for massive collection of song emotion labels and discusses their applicability. In Section 3.3 we describe the steps and methodology we followed

for creating MoodyLyrics4Q and MoodyLyricsPN, two relatively big datasets of song emotions. In the end, Section 3.4 explores the possibility of using a lexicon-based sentiment analysis method as a generative function of song emotion labels.

3.1 Music and Emotions

3.1.1 Music Emotions: Concepts and Definitions

In the epoch of online networking and forums, music listening and appreciation has become social and collective. Personal CD players and collections of album disks kept in drawers are part of history. Subscriptions in streaming Web and mobile platforms like *SoundCloud*, *Pandora Radio*, *AllMusic*, *Last.fm* or *Spotify* took their place. The first big shift came in when Apple introduced iTunes and iPod in 2001. They enabled users to purchase, download and store digital music files in the fancy and easy to use iPod, instead of going to the store for buying the preferred album CDs. The eminent transition from local to cloud storage and the proliferation of smart mobile devices created the right conditions for transiting to the above-mentioned subscription platforms that offer listeners streaming of the favorite tracks instead of buying whole albums. The immediate consequence was a steady decrease in album sales. Data (songs) are now stored in the cloud and users can create playlists or music preference profiles, find songs, like and share them or even suggest them to other peers. They also receive music recommendations based on their profiles or previous preferences. Search and retrieval of songs is based on typical metadata like title, artist or genre, as well as on emotions they convey.

All these developments were made possible by (and also promoted) significant advances in Music Information Retrieval (MIR) and Music Emotion Classification (MEC) research. Wheres MIR is the application of information retrieval to find music or songs, MEC can be seen as sentiment analysis applied in music domain. Both are based on data mining and AI (machine learning and/or deep learning) techniques, utilizing various types of music metadata and features. The problem here is that we are dealing with *music* and *emotions*, two abstract and complex concepts that are study subjects of other disciplines like psychology, philosophy or sociology. There are certainly many definitions of music and emotions from the perspective of such disciplines. In general, music can be described as *an art of sound in time that*

expresses ideas and emotions in significant forms through the elements of rhythm, melody, harmony, and color.[1] From our perspective, music or emotion definitions are not that important. A more intriguing question is "Why do people listen to music?". Authors in [31] find (by means of statistical evidence) and report that the most important motivations for daily listening to music are:

- **Self-awareness** – *Music listening is highly associated with self-related thinking, emotions and sentiments, absorption and escapism, etc.*
- **Social relatedness** – *Music involves people in socializing and affiliation, helping certain individuals to reveal that they belong to particular social groups, connect to their peers, etc.*
- **Mood regulation** – *Music can help to get in a positive mood, to stimulate the physiological arousal, to amuse or entertain, etc.*

Their findings show a strong correlation between music listening and self-regulation of mood. People tend to listen to specific kinds of music when they are in a particular emotional state. Music expresses and induces *emotions* and influences *mood*, two concepts that are even more abstract and difficult to stipulate. In psychological studies like [32] we find the following definitions:

- **Affect** – *A neurophysiology state consciously accessible as a primitive, non-reflective feeling, most evident in mood and emotion but always available to consciousness.*
- **Emotional episode** – *A complex set of interrelated sub-events related with a specific object.*
- **Mood** – *The appropriate designation for affective states that are about nothing specific or about everything, about the world in general.*

Something we can spot from these confusing definitions is the fact that moods usually last longer in time than emotions. This has also been stressed in the literature. Nevertheless, from our perspective, the two terms are quite similar. As a result, throughout this thesis, they are used interchangeably.

[1] http://www.dictionary.com/browse/music

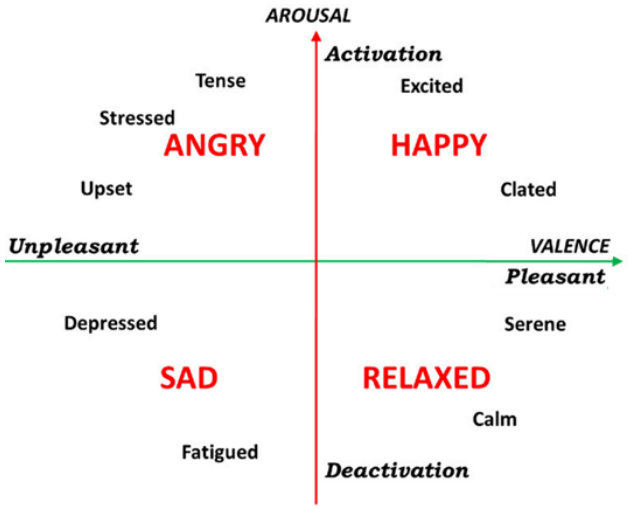

Fig. 3.1 Model of Russell

Table 3.1 Mirex emotion clusters

Cluster	Emotion Adjectives
Cluster 1	passionate, rousing, confident, boisterous, rowdy
Cluster 2	rollicking, cheerful, fun, sweet, amiable/good natured
Cluster 3	literate, poignant, wistful, bittersweet, autumnal, brooding
Cluster 4	humorous, silly, campy, quirky, whimsical, witty, wry
Cluster 5	aggressive, fiery, tense/anxious, intense, volatile, visceral

3.1.2 Models of Music Emotions

It is not easy to elaborate or describe music emotions. Actually, there are big variations in taxonomies and the terminology that is used. For example, when searching for music by mood in *AllMusic* website, the user is confronted with about 570 descriptors.[2] Reducing this excessive complexity in a manageable set of categories is an essential prerequisite. Psychologists have developed models or taxonomies of music emotions that are very important for alleviating the problem. Two types of music emotion models can be found in the literature: categorical and dimensional. Categorical taxonomies describe music emotions using short text labels or descriptors. Those labels that are enough close semantically (synonyms) are clustered together to form a music emotion category. Contrary, labels representing contrasting emotions should appear in different and distant categories. Dimensional

[2] https://www.allmusic.com/moods

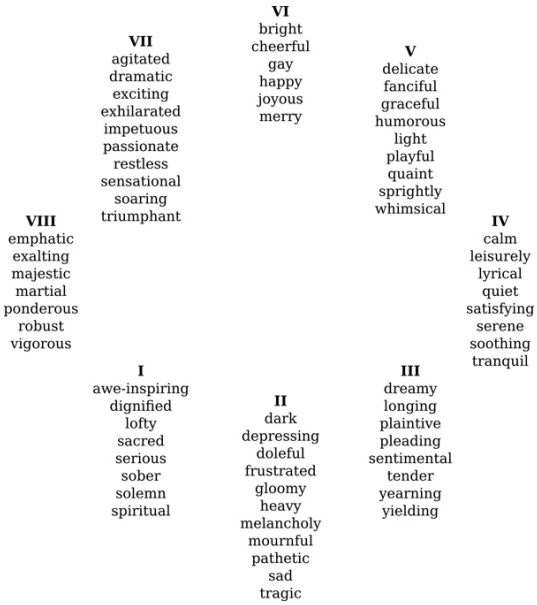

Fig. 3.2 Model of Hevner

models, on the other hand, represent emotions using numeric values of certain parameters (dimensions) in a continuous space. Valence and arousal are two typical dimensions that are commonly used in different dimensional models. No common agreement exists about what type of models are best in various situations. There are however some of them that have gained wide popularity in the research literature.

One of the earliest studies in this field was conducted by Hevner in 1936 [33]. She presented a categorical model of 66 emotion labels clustered in eight categories as shown in Figure 3.2. Obviously, there is high synonymy between labels of same class and dissimilarity (sometimes antonymy) between labels of different classes. The model of Hevner has not been used widely in its original form. Nevertheless, it is considered as an important starting point for constructing other music emotion representation models. A more recent categorical model proposed in [34] organizes music emotion labels in five classes as shown in Table 3.1. Despite the semantic overlapping between clusters 2 and 4 reported in [35], that model has been widely used in several MEC tasks. One of the most popular dimensional models was proposed by Russell in 1980 [7]. It represents music emotions as points in a two-dimensional plane of valence and arousal dimensions as shown in Figure 3.1. *Valence* (pleasant - unpleasant) represents how much an emotion is perceived as *positive* or *negative* whereas *Arousal* (sleepy - aroused) indicates how strongly the emotion is

felt. Russell's model is simple, intuitive and highly used in research literature. There are also other models that add more dimensions to represent music emotions.

3.1.3 Sentiment Analysis of Songs

Sentiment analysis of songs is about utilizing data mining or AI techniques in combination with different features to correctly classify songs in mood categories based on the most typical emotion types they express. Identifying emotions in songs is however complex and difficult. The main trouble is the subjective nature of music perception. Songs are perceived in various ways from different subjects. Another difficulty comes from the heterogeneous form of features that are found in songs. Besides audio which is the most important component, there is also text (lyrics) and other types of attributes like instrumentation, genre, epoch, etc. The different approaches are usually identified by these features they mostly process. In the literature, the most popular feature types are:

- **Tags** – Social tags such as "mellow", "bittersweet", "cool", "90s" and many more, may be useful for mood, genre or instrument recognition tasks.
- **Lyrics** – Sentiment analysis of song lyrics can also be highly convenient for emotion identification in musical tracks.
- **Audio** – Techniques that work with sound were the earliest explored and are based on acoustic features like cepstral coefficients, energy distribution, etc.
- **Hybrid** – For higher accuracy some researchers mix various features in certain ways and construct multimodal solutions.

In the context of music listening, a tag is just a free text descriptor provided by any user for describing a musical object such as a song, album or artist. Some typical examples are "rock", "melodic", "bass", etc. Today's music portals collect many of them for free on a daily basis. The power of these tags lies in the detailed and descriptive information they carry. Tags are semantically rich and easy to work with. In the literature, social tags are frequently analyzed to build and test folksonomies of music emotions. Authors of [36] for example, propose a method they call Affective Circumplex Transformation (ACT) for transforming the planar model of Russell into a space of *valence*, *arousal* and *tension*. This new space

enables a better representation for emotionality of music tags and tracks. Also in [37], authors exploit *Last.fm* tags of emotions for creating a simple representation space of three clusters only. Their emotion space seems oversimplified and has not gained popularity. Nevertheless, their approach can be considered as a valuable guideline for similar works. Despite their advantages, social tags have drawbacks as well. The absence of a common vocabulary of tags creates ambiguity and problems such as polysemy. Furthermore, user tags do frequently contain irrelevant feedback (noise) and thus require careful preprocessing.

Same as tags, song lyrics represent another source of high-level features. They are easily processed and usually found for free (contrary to audio that is mostly copyrighted). Studies utilizing song lyrics usually fall into two categories: lexicon-based and corpus-based. The former try to predict emotionality by mapping lyrics words with their synonyms in affect lexicons. ANEW (Affective Norms for English Words) presented in [38] is one of such affect lexicons that has been widely used. It provides a set of normalized emotional scores for the 1040 English words it contains. Those words were rated in terms of *valence*, *arousal* and *dominance* from many human subjects who participated in the psycholinguistic experiments. Dominance is an emotion dimension that represents the scale of ascendancy (vs. submissiveness) a word induces. Authors in [39] make use of ANEW terms to find the emotional category of the intro and refrain parts of song lyrics. They assume that intro and refrain are the most emotionally significant parts of song lyrics. Furthermore, in [40] authors create and use ANCW, the Chinese version of ANEW. They compute valence and arousal of each word appearing in song lyrics and afterwards the aggregate values of each sentence. For integrating values of all sentences and deriving the emotion category of entire song, fuzzy clustering is utilized.

Corpus-based approaches utilize collections of mood-annotated lyrics to train models. In this respect they represent pure supervised learning solutions. One such study is [41] where authors show that word oriented metrics like term frequency or tf-idf provide important insights for automatic mood classification of lyrics. Feeding such features in traditional classifiers they train a model that is able to predict emotionality of unlabeled songs. Despite being free and easy to work with, lyrics are not available for some types of music like instrumental, classical etc. In contrast with that, audio is the universal component in all types of music. It is also very rich in various feature types. Authors in [42] experiment with loudness, pitch, tempo, tonality, harmonics, key, and rhythm. Support vector regression is used

to map features with emotion categories and 94.55% accuracy is reported. There are still difficulties when working with audio. Expertise in musical concepts and signal processing is required at some scale. There are also performance limitations related to the low-level audio features. Their semantic gap with the high-level user perception reduces accuracy and applicability. To mitigate the cons of the above feature types, some studies like [43] or [44] employ feature aggregation strategies. It is typical for example to combine tags with audio or lyrics. Sometimes other types of metadata like artist, genre or instrumentation are mixed in.

3.2 Crowdsourcing Song Emotions

The recent tendency towards intelligent models that learn from data does not exclude music domain. As explained in the previous section, many researchers prefer supervised learning strategies for emotion recognition in songs. However, one difficulty they commonly face (and we faced) is the lack of song datasets with emotion labels correctly assigned. Manual annotation of emotions to musical tracks is time-consuming, costly and labor intensive. Crowdsourcing is a recent network-powered work paradigm that may solve this problem. As a term, it was first coined by Jeff Howe in [45]. He describes cases in which this distributed labor approach has proven highly successful not only for lowering production costs but also for finding innovative R&D solutions. The fundamental elements that make crowdsourcing work well are independence and variety of opinions, work decentralization, and opinion aggregation [46]. The interested parties (employers, organizations, institutions, researchers, etc.) publish job requests or unsolved problems in online platforms that serve as marketplace networks.

Some of these platforms (e.g., InnoCentive, NineSigma, iStockphoto or YourEncore) target specialized or talented subjects, especially for addressing R&D creative problems in specific areas. Other platforms like Amazon Mechanical Turk address simple and repetitive tasks that any subject with Internet access can elaborate. In MTurk participants are paid by the publisher of those tasks after successfully completing them. MTurk is thus highly suitable for activities which require massive social involvement like emotion annotation of a high number of tracks. Other forms of crowdsourcing campaigns are conceived as challenges with bountiful rewards like Netflix $1M Prize. Back in 2006, Netflix requested an algorithm for movie recom-

mendations with error rate 10% lower than state-of-the-art and offered the money to the team that would propose it first. That challenge had a positive public impact, boosting research in the field of recommender systems which are now part of almost every commercial or advertising web platform. The suitability of crowdsourcing alternatives for experimental microtasks such as statistical surveys or data collection has attracted interest from researchers, including those that work in music emotion recognition. Some of the most explored alternatives for collecting emotion labels of songs are:

MTurk As mentioned above, this marketplace is probably the most popular for tasks of massive involvement. Many researchers use it for gathering feedback about emotionality of songs. Authors of [47] for example, involve MTurk workers for collecting tags of various types like mood, instrument, genre, etc. After analyzing the data, they conclude that different intervals of the same song are usually described differently from users. There are also authors that use MTurk not only for creating datasets, but also compare labeling quality with that of other methods to assess the viability of MTurk. In [48] for example, they compare annotations crowdsourced from MTurk with those obtained from MoodSwings, a collaborative game they developed. Both annotations follow an emotional model of four categories derived from the planar model of Russell. Based on their results, authors report accordance between MoodSwings and MTurk data, concluding that the later is an applicable method for song annotation. Furthermore in [49], MTurk labels are contrasted with the ones collected from MIREX[3] campaign. Authors conclude that agreement rates are satisfactory and MTurk crowdsourcing can serve as a practical and inexpensive alternative for creating music mood ground truth datasets.

Online games Fancy and amusing online applications like games may represent an interesting option for crowdsourcing opinions. Online users are certainly more inclined to play games than answer survey questions. MajorMiner[4] described in [50] was one of the first online games specifically designed to gather opinions about emotions of certain songs. Users first listen to ten-second clips and then select the most representative tags from a predefined list. Authors compare tags collected from their game with those obtained

[3] http://www.music-ir.org/mirex/wiki/MIREX_HOME
[4] http://majorminer.org/info/intro

from *Last.fm* music portal. They conclude that MajorMiner tags are less noisy and can serve for creating emotion datasets of songs. TagATune is another collaborative game developed to crowdsource music clip labels [51]. Here players are involved in a rich audio experience and coupled with a partner for tagging tunes agreeably. Authors pretend that TagATune is more effective than MajorMiner because of its entertaining features. MoodSwings mentioned above was specifically developed to collect mood tags of songs. It is more effective in tag collection as it makes users provide ratings on a per-second basis. User ratings are then converted in labels using the valence-arousal planar model of emotions.

Social tags From the different music listening platforms, *Last.fm* is probably the most popular among academics. This is because of the open API it provides for collecting and analyzing tags, metadata and musical preferences of its users. Consequently, *Last.fm* user tags appear as an important research resource in many academic works. In general, users tend to provide tags about songs and other online objects for several reasons. Creation of assistance for future searches, expression of opinions and social exposure are some of the most important [52]. One of the first studies that examined type distribution of song tags is [53]. According to this study, 68% of tags are related to song genre, 12% to locale, 5% are mood tags, 4% of them are about instrumentation tags and 4% express opinion. In [4] on the other hand, we find one of the first studies that specifically examined mood social tags. Authors report an unbalanced distribution of emotion tag vocabulary. They also infer that many labels are interrelated or reveal different views of a common emotion category. In [54] authors utilized *AllMusic* tags to create a ground truth dataset of song emotions. They first used tags and their norms in ANEW to categorize each song in one of the four valence-arousal quadrants of Russell's model. Afterwards, three persons validated and improved annotation quality manually. The resulting dataset has 771 songs and their corresponding emotion category.

Other Research papers explore many other strategies for collecting opinions about songs like online web services, traditional surveys etc. An example is Songle: a web service with music visualizations presented in [55]. It enables users to play music and associate each played track with corresponding visualizations of beat structure, vocal melody, chords and other characteristics that are

sometimes erroneous. Users that spot errors in visualizations can provide corrections which are shared for improving the experience of future users.

The high number of works in the literature that are exploring crowdsourcing options emphasizes the growing importance of the network-powered crowd intelligence. A more detailed discussion about the crowdsourcing alternatives discussed in this section can be found in [56].

3.3 Creating Song Datasets from Social Tags

In the previous section, we discussed different alternatives for collecting emotion labels about songs. Here we present the creation steps of two relatively big song emotion datasets, MoodyLyrics4Q and MoodyLyricsPN. Firstly, various existing datasets together with their limitations are described. Afterwards, we illustrate the systematic process for dataset creation.

3.3.1 Existing Song Emotion Datasets

The need to experiment with emotionally annotated songs has motivated researchers to utilize various strategies for creating music ground truth datasets. Such datasets should possess the following characteristics:

1. *Contain as many songs as possible (e.g., more than 1000)*
2. *Annotated following a well-known model of emotions*
3. *Have polarized annotations to be usable as ground truth*
4. *Publicly released for cross-interpretation of experiments*

The above characteristics are conflicting and hardly achieved together. Due to the subjective nature of music appraisal, complete cross-agreement between different subjects involved as annotators is hardly achieved. Hiring several music professionals for manual annotation would certainly produce high-quality data. However, it would also be time-consuming and probably very costly. Actually, organizations like *Pandora* employ music experts for annotating tracks with relevant emotion words.

Nevertheless, they are not willing to share their datasets for public use. Many researchers have explored crowdsourcing alternatives discussed in the previous section for the sole purpose of constructing labeled song datasets. Authors in [57] crawl *Last.fm* tags and use them for constructing a big dataset of 5296 songs dispersed in 18 emotion categories (synonymous tags). Annotation is automatically performed using a binary method (tagged, not tagged) for each song and emotional category. This dataset could be convenient for various types of music emotion recognition experiments. Nevertheless, it has not been released to the public. Also in [54], authors created a public and polarized dataset of songs using *AllMusic* tags and following the popular valence-arousal model of Russell. However, that dataset is relatively small, consisting of 771 tracks only.

Another music dataset is described in [58]. Authors involved many MTurk workers to annotate songs according to valence-arousal planar model. They claim that each clip has been labeled by at least ten workers which guarantees high polarity and data annotation quality. The final dataset contains 744 entries and is publicly released as a whole (clips and features). Also in [59], a traditional survey based on questions to paid participants is used to obtain emotional labels. The resulting dataset is public and consists of 500 (small) western songs. A multimodal dataset containing text and audio features of 100 songs (very small) is presented in [60]. Authors collect emotion labels from MTurk workers and provide the dataset upon request for academic use only. Several other research works create close datasets that are used to evaluate methods or algorithms they propose. Though hard to believe, we could not find any experimentation dataset that fulfills all four requisites listed above. In the following sections, we describe the steps we followed for creating two such datasets ourselves. MoodyLyrics4Q is a dataset of 2,000 songs, fully compliant with the four requisites listed in the previous section. We also created MoodyLyricsPN, a bigger collection of 5000 songs labeled as *positive* or *negative* only (violation of second requisite).

3.3.2 Folksonomy of Emotion Tags

For the annotation of the songs, we decided to crawl social tags from *Last.fm*. To comply with the second requisite of the previous section, we had to find a commonly used emotional model. Apart from the few popular models of psychologists discussed in Section 3.1.2, there are also folksonomies of music emotions built in

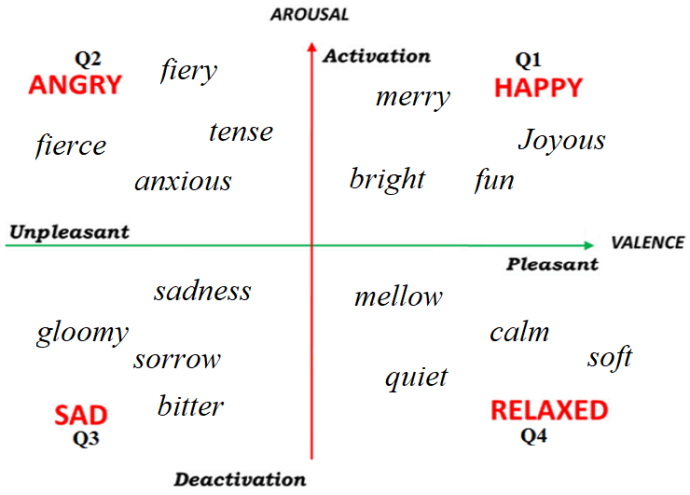

Fig. 3.3 Adopted model of tag emotion categories

research works starting from social tags about songs. In [61] for example, they perform clustering on *AllMusic* mood tags and aggregate them in four classes that are analogous to the four quadrants of Russell. A similar work utilized *Last.fm* emotion tags [6]. Authors apply unsupervised clustering and Expected Maximization algorithm to document-tag matrix. They report four as the optimal number of emotion tag clusters. Moreover, their four clusters (*happy*, *angry*, *sad*, *relaxed*) are again analogous to the four quadrants of Russell's model. All these research results confirm that categorical models of song emotions that are derived from social tags do comply with the theoretical models of psychologists and are applicable for sentiment analysis of songs. They also convinced us that among the various models of emotions, the categorical version of Russell's model with one emotional category for each quadrant (*happy* for Q1, *angry* for Q2, *sad* for Q3 and *relaxed* for Q4) is the most simple, intuitive, widely recognized and practical for our purpose. A graphical illustration of the model is shown in Figure 3.3.

For organizing tags, we constructed a folksonomy of terms that is very similar to the one of [6]. First, we retrieved 150 mood terms from relevant research papers and the mood terms from *AllMusic*[5] portal. A preliminary selection process was conducted manually to keep in only terms that clearly fall in one of the four categories of our model and filter out ambiguous ones. We consulted ANEW valence and arousal norms of each word for objectivity in selection. The quality of a folksonomy of terms can be measured by the average intra-cluster (as high as possible) and

[5] https://www.allmusic.com/moods

Table 3.2 Four clusters of tag terms

Q1-Happy	Q2-Angry	Q3-Sad	Q4-Relaxed
happy	angry	sad	relaxed
happiness	aggressive	bittersweet	tender
bright	fierce	bitter	soothing
joyous	outrageous	sadness	mellow
cheerful	rebellious	depressing	gentle
fun	anxious	tragic	peaceful
humorous	fiery	gloomy	soft
merry	tense	miserable	calm
exciting	anger	funeral	quiet
silly	hostile	sorrow	delicate

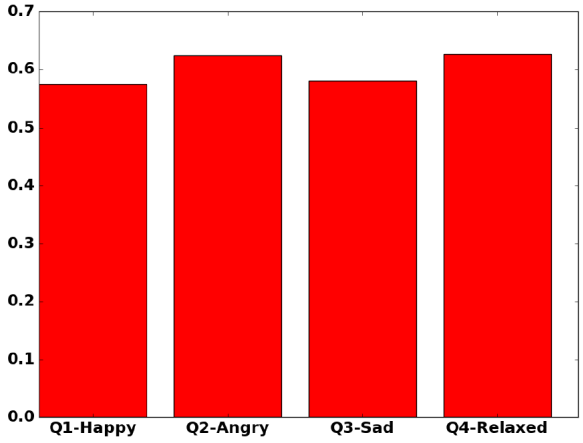

Fig. 3.4 Intra-cluster similarity of tags

inter-cluster (as high as possible) similarities. For this reason, we utilized word embeddings trained from a corpus of 2 billion tweets with Glove method.[6] Word embeddings are popular for their ability to capture semantic similarities between words (see Section 2.4.5). The average intra-cluster similarities were optimized by trying a high number of tag combinations inside each of the four clusters. The optimal configuration resulted the one shown in Table 3.2 which comprises the ten most suitable mood tags in each cluster. Intra-cluster similarities are presented in Figure 3.4. Inter-cluster similarities, on the other hand, are shown in Figure 3.5. As we can see from that figure, the less similar (or more dissimilar) clusters are Q1-Q3 and Q2-Q4 (the diagonals in the plane) which differ in both valence and arousal.

[6] http://nlp.stanford.edu/data/glove.twitter.27B.zip

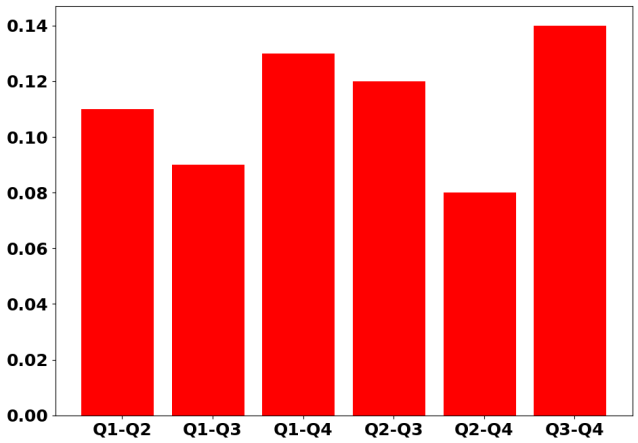

Fig. 3.5 Inter-cluster similarity of tags

3.3.3 Data Processing and Annotation

To produce a large final set of labeled songs (first requisite) we imported all tracks of *Million Song Dataset* presented in [62]. It is one of the biggest song collections, created to test the scalability of algorithms to commercial sizes. We mixed in the records of Playlist dataset as well. This is a smaller collection (75,262 tracks) of more recent songs [63]. At this point, a total of 1018596 tracks was reached. Data processing went on removing duplicate tracks. Afterwards, we crawled all tags of each track utilising *Last.fm API*.[7] Songs with no tags were removed and statistical analysis of tags was performed. The most frequent tag was *rock* appearing 139295 times, followed by *pop* with 79083 occurrences. We also analyzed tag type frequencies. Genre tags were the most common with 36% of the total, followed by opinion (16.2%) and mood (14.4%) tags.

Among mood tags, *mellow* was the most frequent with 26,890 occurrences, followed by *funk* (16324) and *fun* (14777). The word cloud of mood tags is shown in Figure 3.6. There was an obvious bias towards positive emotion tags. This is probably because people are more inclined to give feedback when they listen to positive songs. Popularity bias may be another reason. After concluding the analysis of tag statistics, we moved on removing every tag that was not about mood or other tags that were ambiguous (e.g., we could not know if tag *love* means the user loves that song or he/she thinks it is about love). At the end of this phase, we reached to 288708 tracks. Further details about data processing steps and tag statistics can

[7] https://www.last.fm/api/show/track.getTags

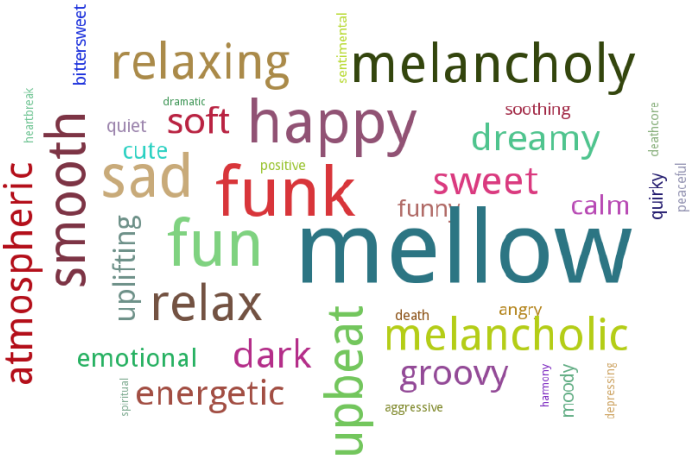

Fig. 3.6 Word frequency cloud of mood tags

be found in [64]. Next, we identified and counted emotion tags of each cluster appearing in the remaining tracks. Four counters (one per emotion cluster) were obtained for every track. To reach to a polarized collection of songs (third requisite) we used a tight annotation scheme. A track is set to quadrant Qx if it fulfills one of the following conditions:

- has 4 or more tags of Qx and no tags of any other quadrant
- has 6 up to 8 tags of Qx and at most 1 tag of any other quadrant
- has 9 up to 13 tags of Qx and at most 2 tags of any other quadrant
- has 14 or more tags of Qx and at most 3 tags of any other quadrant

Songs with fewer than four tags or those not fulfilling any of the above conditions were discarded. This scheme guarantees that even in the worst case scenario (song tag distribution), any song set to Qx quadrant has more than 75% of all its received tags being part of that quadrant. What remained was a collection of 1986 happy or $Q1$, 574 angry or $Q2$, 783 sad or $Q3$ and 1732 relaxed or $Q4$ songs for a total of 5075 (2,000 after balancing).

Datasets with *Positive vs. Negative* representation are clearly oversimplified and do not reveal much about song emotionality. However such kind of datasets could be used for various experimental purposes. We merged $Q1$ with $Q4$ (*happy* with *relaxed*) considering them as *positive*, and $Q2$ with $Q3$ (*angry* with *sad*) for the

Table 3.3 Confusion matrix between A771 and ML4Q datasets

A771 \ ML4Q	Happy	Angry	Sad	Relaxed
Happy	**97.43**	0.85	0	1.7
Angry	0.85	**98.29**	0.85	0
Sad	0	0.85	**97.43**	1.7
Relaxed	1.7	0	1.7	**96.58**

negative category. The corresponding tags of each cluster were recombined as well. As binary discrimination is easier, an even tighter annotation scheme was enforced. A track is considered to belong to Qx (*positive* or *negative*) only if:

- it has 5 or more tags of *Qx* and no tags of the other category
- it has 8 up to 11 tags of *Qx* and at most 1 tag of the other category
- has 12 up to 16 tags of *Qx* and at most 2 tags of the other category
- has 16 or more tags of *Qx* and at most 3 tags of the other category

This scheme guarantees that even in the worst case scenario (song tag distribution), any song labeled as *positive* or *negative* has more than 85% of all its received tags being part of that category. We got a collection of 2589 *negative* and 5940 *positive* songs, for a total of 8529 (5,000 after balancing). Apparently, the resulting datasets are imbalanced towards positive songs, same as the corresponding emotion tags they were derived from. To have an idea about the quality of the first labeling scheme that was used, we compared our labels of the first dataset (ML4Q) with those of another one considered as ground-truth. The most appropriate for our purpose was the dataset (here A771) described in [54]. It consists of 771 songs labbeled according to the planar model of Russell, same as we did. Authors used *AllMusic* tags for the process and involved three persons to validate the annotation quality. The problem is however the size of this dataset. From the 771 songs it contains, only 117 were part of our initial collection of 5075 labeled tracks.

In Table 3.3 we show the confusion matrix between labels of our dataset and those of A771 for each category. As we can see, the overall agreement between the two datasets is 97.28%. Despite the fact that this result is based on a small portion of the records, it seems to be high enough to confirm the validity of our method. Both

datasets presented here can be freely downloaded from our group website.[8] Lyrics or metadata of their songs can be easily retrieved from online music websites. Audio is usually copyrighted and hard to find. Researchers who have access to audio of songs can experiment with sound features as well.

3.4 Music Data Programming via Lexicons

The two datasets we created may be big enough to feed traditional machine learning algorithms. However, they are still small for deep neural networks. Actually, it is difficult to collect data in music domain as songs are usually copyrighted. Alternative implementations of data programming introduced in Section 2.5 might be good options for constructing bigger datasets of emotional categories. In this section we present the results of some experiments we conducted with a text emotion identification method that was used as a generative function of mood labels. We also observed the quality of generated labels by comparing them with a benchmark dataset. The basic method for text sentiment identification is described in [65] where authors illustrate its use for computing overall positivity of large-scale texts such as song lyrics, blog posts, etc. It is based on utilization of valence norms for each word found in ANEW lexicon. The norm of each word appearing in the text under analysis is summed and then the total is divided by the total number of content words to get the average. Authors utilize this simple and fast technique for estimating overall positivity in song lyrics of different epochs. To construct a dataset of four emotion categories we can use both valence and arousal norms of ANEW and compute their totals for each song text with the following equations:

$$v_{lyric} = \sum_{i=1}^{n} v_i f_i \; / \; \sum_{i=1}^{n} f_i \qquad a_{lyric} = \sum_{i=1}^{n} a_i f_i \; / \; \sum_{i=1}^{n} f_i \tag{3.1}$$

Here v_{lyric} and a_{lyric} represent valence and arousal of all words in the text that also appear in ANEW. Also, f_i is their frequency in the text wheres v_i and a_i represent ANEW valence and arousal norms of each text word. Because the values are real numbers from 1 to 9, we adapted valence-arousal planar model as shown in Figure 3.7.

[8] http://softeng.polito.it/erion/

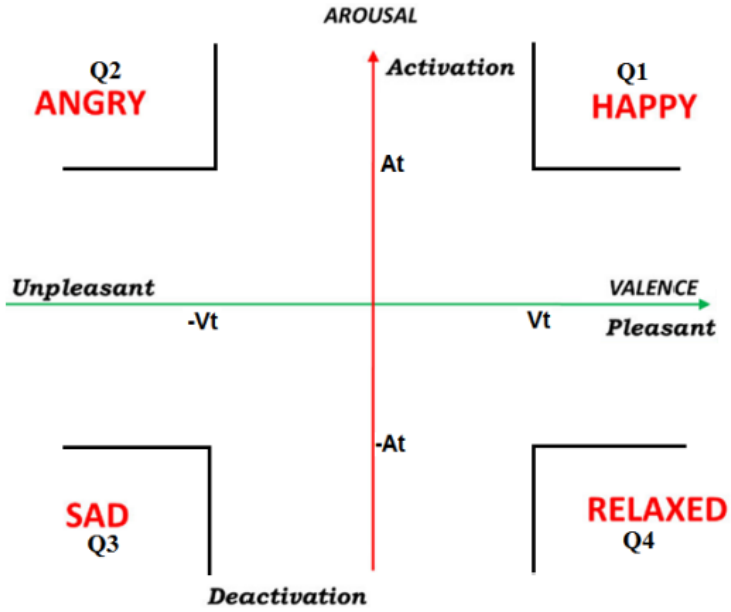

Fig. 3.7 Planar model for emotion categories of texts

Obviously, the 1 – 9 interval was transformed to get zero-centered values ranging from -4 to 4 and then use the sign value as the discriminator. Aggregate v_{lyric} and a_{lyric} values of a text computed with Equation 3.1 provide a point in the plane that falls in one of the quadrants. The corresponding emotion category is assigned to the text. To avoid misclassification of the points appearing near the origin (values close to 0), threshold valence and arousal (Vt and At) values are used. The rectangular zone $[(Vt,At),(-Vt,At),(-Vt,-At),(Vt,-At)]$ is considered as "unknown". One of the four labels is assigned to each text only if its point falls in one of the quadrants indicated in Figure 3.7. For example, a text is labeled as "sad" only if $v_{lyric} < -Vt$ and $a_{lyric} < -At$. If we are interested in text positivity only, we can use just the first formula of Equation 3.1 to compute v_{lyric}. This value represents a point which falls somewhere in valence axis of Figure 3.8. If it is enough displaced (black bars in the figure) the text takes the corresponding polarity. A problem with the above method is that it does not work well for texts containing just a few words that are part of ANEW. Given that song lyrics usually contain slang or rare words, we faced the need to extend ANEW. A much bigger and generic English lexicon is WordNet which contains at least 166,000 *(word, sense)* pairs [66]. Words in WordNet have synonymy relations which other and word senses are sets of synonyms called synsets. WordNet-Affect, on the other hand, is a highly reduced subproduct of Wordnet that contains emotion terms [67]. To overcome the size problem of ANEW, we decided

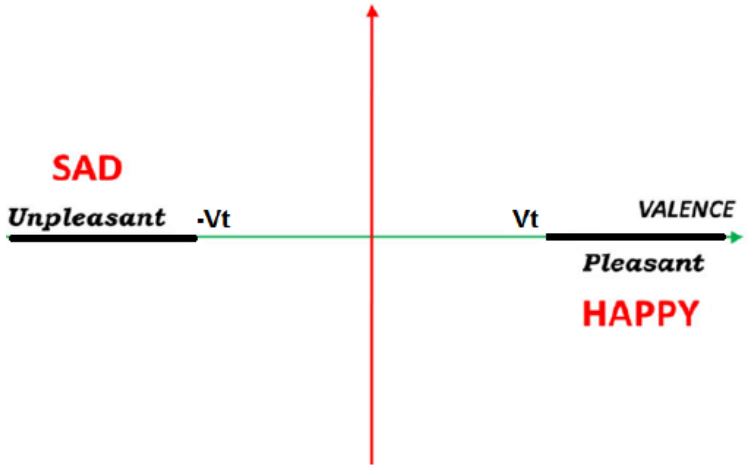

Fig. 3.8 Planar model for emotion polarity of texts

to combine the three lexicons in the following way. For each ANEW word, we checked WordNet synsets that include that word and extended it with the resulting synonyms. All imported words from WordNet took valence and arousal values of the ANEW source word. Afterwards, we kept only those words that belong to synsets of WordNet-Affect labeled as *Mood*, *Sensation* or *Emotion*. All added words of other synsets were removed. This way we reached a set of 2162 words which is more than double size of ANEW. In [68] authors extend ANEW in a similar way to experiment with heterogeneous text features.

We evaluated labeling quality of the method applying it to an existing dataset of emotionally labeled songs. Once again, we used the dataset described in [54] as external ground-truth. We applied the method described above on each song text generating the new labels which were compared with those of the dataset. First, $Vt = 0.25$ and $At = 0.25$ were used and the agreement was low. Increasing Vt and At increments polarization of generated labels but also reduces their quantity. This is because more lyrics start to fall inside the "unknown" zone of the plane. At this point, the goal was to explore many Vt and At combinations for maximixing accuracy of the generated labels with respect to those of the ground-truth dataset. Vt and At were increased by 0.01 on each comparison. After many trials, we stopped at $At = 0.34$ and $Vt = 0.34$, reaching a maximal conformity of 74.1% from 220 labeled lyrics. Further increases of Vt and At values significantly reduced number of comparable lyrics and computed accuracy started to go down. More details and statistics about dataset lyrics and the automatic annotations we obtained can be found in [69].

Table 3.4 Confusion matrix of lexicon-generated song labels

True \ Pred	Happy	Angry	Sad	Relaxed
Happy	**68.68**	3.63	2.72	25
Angry	5.9	**80.45**	13.63	0
Sad	7.27	15.9	**74.54**	2.27
Relaxed	18.18	0	9.09	**72.72**

Table 3.4 presents confusion matrix between the generative method we used and the ground-truth dataset. The overall accuracy of 74.1% is probably not high enough for considering the method as applicable. Various reasons could be the cause of this. First of all, the method itself is "crude". It simply sums valence and arousal norms. Some terms (e.g., verbs) could be emotionally more important than other terms. Furthermore, meaning and emotionality of words are highly dependent on the context in which they are used. Unfortunately, ANEW norms of words are static numbers that do not count for that context. Another problem could be the way we extended ANEW. In conclusion, we do not consider this method as an applicable generative function for emotion labels of texts. Nevertheless, it might be useful if combined with other high-level heuristics devised from music and emotion experts.

Chapter 4

Mood-Aware Music Recommenders

"Information overload is a symptom of our desire to not focus on what's important. It is a choice."

– Brian Solis, digital analyst

In the era of pervasive computing and "everything online" culture, people have an essential need for automatic filtering tools to alleviate the information overload problem and the distraction it induces which is stressing. Search engines and recommender systems are such tools that have become very popular. The latter were particularly promoted by the increasing Internet commerce. They utilize several filtering strategies as well as various types of data to predict items that should be the most useful for the users. Hybrid recommender systems try to make better user preference predictions by combining two or more basic filtering techniques whereas context-aware recommenders utilize contextual data for achieving the same goal. Songs are one type of items that most people consume consistently on a daily basis, especially in the context of car driving.

This chapter presents survey results about recent research trends in hybrid and context-aware recommender systems. Furthermore, we describe a contextual mood-based recommendation system for music suggestions to car drivers that aims to enhance their driving experience, comfort, and cautiousness. Section 4.1 introduces the basic recommendation techniques and describes some public datasets that can be utilized for experimentations. Section 4.2 provides an even more detailed discussion

about hybrid and context-aware recommender systems as well as their applicability. Finally, in Section 4.3, design steps and module details of the contextual music recommender are presented.

4.1 Recommender Systems

4.1.1 Introduction and Early History

People have historically counted on their peers or experts for suggestions or recommendations about what products to buy, what places to visit, what songs to listen, etc. Things have changed in the last two decades with the proliferation of the Internet and Web access. Most people today use search engines and information found on websites for such suggestions. The huge and increasing amount of data available on the Web, combined with the massive daily-generated user content have created the problematic phenomenon of information overload. Regardless of the quote at the head of this chapter, *information overload* is more formally defined as *"a situation in which you receive too much information at one time and cannot think about it in a clear way."*[1] This problem induces stress and restricts our capability to review specifications of the objects for choosing the most convenient one from the many alternatives. To address the problem, computer science and technology have reacted appropriately and automatic information filtering tools have been developed. Recommender Systems (RS) represent a category of such tools invented in the 90s to provide recommendations or suggestions of interesting items to users [70]. Nowadays, RSs appear everywhere in the Web for assisting users in finding different items or services. They are also an important instrument for businesses, advertising products and increasing sales.

In their dawn (early 90s), RSs were mostly studies of research disciplines like human-computer interaction or information retrieval. One of the earliest that appeared was Tapestry, a manual Collaboration Filtering (CF) mail system [71]. The first computerized versions (GroupLens, Bellcore, and Ringo) of the mid-90s also implemented collaborative filtering strategy [72, 73]. GroupLens was a CF engine designed to find and suggest news. Bellcore presented in [73] was a video recom-

[1] https://dictionary.cambridge.org/dictionary/english/information-overload

mendation algorithm embedded in the Mosaic[2] browser interface. Ringo, on the other hand, utilized preference similarities of users to suggest them personalized music. There were also other implementations such as NewsFeeder or InfoFinder for news and documents. They used Content-Based Filtering (CBF) and item features to generate their recommendations [74, 75]. Knowledge-Based Filtering (KBF) or hybrid (combining different strategies) recommenders followed shortly, completing the recommender system mosaic of today.

4.1.2 Basic Recommendation Techniques

Technically, RSs are information filtering engines that try to predict the rating or the preference value that users would give to certain items and then suggest them the best one (or top n). Suppose we have a set of users U and a set of items (e.g., movies) I. If the cardinality of I is high, we normally expect each user $u \in U$ to have watched only certain movies $i \in I$ and given ratings (e.g., 1 to 5) r_{ui} to few of them. In this scenario, we have a $U \times I$ matrix that is mostly sparse, with only a subset of r_{ui} ratings available. The job of the recommender is thus to predict the unknown ratings and fill the matrix by clustering together similar users and items. This simple recommendation approach is known as collaborative filtering. In the case of movies, the recommender predicts what rating would each user give to movies he/she has not watched yet. Afterwards, the movie with the highest predicted rating (or top n after ranking) is recommended to each user. The CF recommender described above uses only three elements or data types: users, items, and ratings. In different types of RSs, other data types are involved or more knowledge is required. The various recommendation strategies that have been proposed differ in the data (or knowledge) and filtering algorithms they combine. In this context, four main RS categories are usually identified: collaborative, content-based, knowledge-based and hybrid filtering [70]. A brief description of each category is presented below:

Collaborative filtering CF recommenders assume that users with similar preferences in the past will keep having similar preferences in the future as well. As briefly mentioned above, ratings or other forms of user feedback are used to identify and cluster common tastes among user groups and then provide suggestions based on intra-user similarities [76]. In this way, users "collaborate"

[2] A popular Web browser of the 1990s, discontinued in 1997.

with each other by "exchanging" their item preferences. A common problem of CF is data sparsity that happens when very few ratings are available and $U \times I$ matrix is extremely sparse. Another common problem is cold-start, a situation with new users or items that have no r_{ui} ratings at all.

Content-based filtering CBF is an approach that usually requires more data (especially about items) than CF. Here, item features are analyzed to identify and cluster together items with similar characteristics. CBF assumes that users who liked items with certain attributes in the past will prefer items with same attributes in the future as well. This type of recommender is highly dependent on (and limited by) the extracted features of recommended items. CBF suffers from cold-start problem, same as CF.

Knowledge-based filtering In KBFs, knowledge about user requirements and item characteristics is utilized to infer the type of items that match user preferences and suggest accordingly [77]. They are more appropriate in scenarios when little or no interaction between users and the system exists. In these cases, users have not provided ratings about items. As a result, CF or CBF cannot be used. For example, when people buy houses no ratings about their previous house preferences are available. In this cases, users enter their item (house) requirements in the knowledge base and the system confronts them with item characteristics to find the best matches. The most important weakness of KBFs is the difficulty to maintain and update the knowledge base.

Hybrid filtering This is a more complex approach that mixes together two or more of the above techniques to alleviate their weaknesses. The most commonly adapted hybrid strategy is the combination of CF with CBF to combat data sparsity problems, increase recommendation accuracy, etc.

Context-Aware Recommender Systems (CARS) represent another complex and advanced filtering strategy. They exploit contextual information (e.g., time, location, etc.) to generate adequate and useful suggestions. Sometimes these RSs are considered as a distinctive category and in other cases, they are described as a special type of hybrid recommender. Section 4.2 discusses both context-aware and hybrid RS types in more details.

4.1.3 Experimentation Datasets

The growing popularity of recommender systems built to direct and assist users online poses the need for systematic and rigorous evaluation of their characteristics to assure user satisfaction. Accuracy, diversity, and novelty are among the most common quality criteria of recommenders that are assessed. One of the most viable and yet effective methods for RS experimentation and evaluation is based on utilizing datasets with feedback data from real users to compare newly developed algorithms or methods with existing ones in the given settings. To help researchers experiments with RSs, in [78] and [79] we describe properties of the most popular datasets available, the repositories they can be retrieved from and various cloud-based recommender systems. These datasets usually contain user feedback about amusing items like movies, books, music, etc. Most of the datasets were built after 2004. The oldest we found was Chicago Entree,[3] a collection of restaurant preferences that dates back to 1996. MovieTwittings[4] is the newest (2013 and on), containing movie preferences expressed in tweets.

We observed that most of the datasets are made up of explicit item ratings provided by users. They are thus highly suitable for evaluating CF recommenders or user similarity measures. Nevertheless, there are still collections of book or music features that are highly appropriate for assessing content-based RSs. Few datasets we found contain subjective user reviews in form of comments. They are thus better suited for sentiment analysis experiments. Regarding access and availability, most of the datasets can be freely retrieved and utilized for non-commercial purposes. Some of them can be obtained upon request to the owner/publisher. Few datasets are closed, restricted or retired from public access. Regarding the format of the data, in most of the cases, they come as simple texts. In some cases though, the data are organized in .csv, .sql or .mdb formats. Prior to using the datasets, researchers are encouraged to make data quality controls to ensure their research requirements are met. A more comprehensive discussion about RS evaluation techniques and practices can be found at [80].

[3] http://archive.ics.uci.edu/ml/datasets/Entree+Chicago+Recommendation+Data
[4] https://github.com/sidooms/MovieTweetings

4.2 Hybrid and Context-Aware Recommender Systems

The debut of Amazon in online commerce late in the 90s boosted interest and academic research in RSs. During that period, hybrid and other complex RS types came out. The very first hybrid recommender was probably Fab, a filtering system that was used to suggest websites [81]. Fab combined CF for finding similar users with CBF to gather websites of similar content. Many other hybrid RSs like [82] that followed explored other combinations. In Sections 4.2.1 and 4.2.2 we present the results of a systematic literature review we conducted on hybrid RSs. Also, in Section 4.2.3 we describe context-aware recommendation strategy. This later was used in Section 4.3 to create a music recommender in the context of car driving.

4.2.1 Hybrid Recommenders: Review Methodology

For the survey on hybrid RSs, we followed the guidelines for systematic literature reviews defined by Kitchenham and Charters in [83]. The following research questions were addressed:

RQ1 *What studies addressing hybrid recommender systems are the most relevant?*

RQ2 *What problems and challenges are faced by the researchers in this field?*

RQ3 *What technique combinations are explored and implemented in hybrid RSs?*

RQ4 *What hybridization classes are used, based on the taxonomy of Burke?*

RQ5 *In what domains are hybrid recommenders applied?*

RQ6 *What methodologies are used for the evaluation and which metrics they utilize?*

RQ7 *Which directions are most promising for future research?*

In RQ1 we observe the relevant studies and try to see any pattern with respect to publication type (e.g., journal vs. conference), publication date, etc. RQ2 identifies the most common problems that hybrid RSs address. RQ3 examines popular technique combinations and associated problems each of them tries to solve. In RQ4 we examine the possible ways in which different techniques can be combined with respect to the systematic taxonomy proposed by Burke [84]. The author examined a

plethora of existing hybrid RSs and created a complete taxonomy of seven hybrid recommender classes we briefly describe below:

Weighted This type of hybrid RSs are very simple and intuitive. They calculate utility scores of recommended items by aggregating output scores of different recommendation strategies utilizing weighted linear functions.

Feature combination In this class of hybrid RSs, the output of one recommender is considered as additional feature data that enters as input to the other (main) recommender. This latter generates the final item suggestions.

Cascade These hybrid RSs that combine two or more strategies makes up another hybridization class. The first recommender in the cascade generates a coarse ranking list and the second strategy refines that list. Cascades are order-sensitive, which means that a CF-CBF cascade is different from a CBF-CF one. It is certainly possible to have cascades of three or even more basic modules chained together.

Switching These recommenders switch between the composing techniques in accordance with certain criteria. As a simple example, we can consider a CF-CBF that mostly uses CF and occasionally switches to CBF (using item features) when CF does not have enough data about users.

Feature augmentation This hybrid RS type uses one technique to produce item predictions or listings that are further processed by the second technique. As an example, we can consider an association rule engine generating item similarities that are fed as augmented features in a second recommender.

Meta-level These hybrid RSs utilize the entire model produced by a first technique as input for the second one. A content-based recommender, for example, may be used to create item representation models that may be entered to a second CF for better item similarity matching.

Mixed These hybrids use different RSs in parallel and select the best predictions of each of them to create the final recommendation list. They represent the simplest form of hybridization and are suitable in cases when it is possible to use a high number (e.g., more than three) of RSs independently.

Experimentation and application domains are examined in RQ5. RQ6 addresses metrics and methodologies that are used for evaluating hybrid RSs and finally, RQ7 summarizes promising research directions. As primary sources for scientific studies, we picked five scientific digital libraries shown in Appendix A, Table A.1. Meanwhile, a set of keywords including basic terms like “Hybrid”, “Recommender” and “Systems” was defined. Later, we added synonyms and organized terms to form the search string shown in Listing 4.1. The whole string was applied in the search engines of the digital libraries and 9673 preliminary research papers were retrieved. The set of inclusion / exclusion criteria listed in Table A.2 of Appendix A were defined for an objective selection of the final papers from the preliminary ones.

```
("Hybrid" OR "Hybridization" OR "Mixed") AND
("Recommender" OR "Recommendation") AND ("System" OR
"Software" OR "Technique" OR "Technology" OR "Engine"
 OR "Approach")
```

Listing 4.1 The search string for finding studies in digital libraries

Utilizing inclusion/exclusion criteria and a coarse inspection based on abstract and metadata, we reached to a set of 240 most relevant papers. Next, we carried out an even more detailed analysis, examining besides abstract, content parts of each paper as well. In the end, we reached to the final set of 76 included papers that are listed in Table A.3 of Appendix A. We also performed a quality assessment of the final included papers. For a systematic evaluation, we defined six quality questions listed in Table A.4 of Appendix A. Each of them was given a certain weight for highlighting the importance of that question in the overall paper assessment. Quality evaluation process consisted in responding with “yes”, “partly” or “no” to each of the six questions. Finally, the overall quality score of each study was computed using the following formula:

$$score = \sum_{i=1}^{6} w_i * v_i / 6 \tag{4.1}$$

w_i is the weight of question *i (0.5, 1, 1.5)*
v_i is the vote for question *i (0, 0.5, 1)*

As part of data extraction phase, both paper attributes (e.g., title, authors, year, etc.) and content data were collected. The data extraction form we used is shown in

in Appendix A, Table A.5. All extracted information was stored in Nvivo,[5] a data analysis software that automates identification and labeling of initial text segments from the selected studies. For the thematic synthesis, Cruzes and Dyba methodology was followed [85]. To organize and aggregate extracted information, that methodology utilized the concept of codes which are labeled segments of text. Codes were later merged into themes for grouping the selected papers. Each research question was mapped with the corresponding themes and extracted data were summarized in categories that were reported as results of the survey. More details and statistics about each step we followed can be found in [86].

4.2.2 Hybrid Recommenders: Review Results

We discuss in this section the obtained results of the systematic literature review on hybrid recommenders and answer each research question listed in the previous section. Regarding quality of the selected studies, we observed that journal papers tend to have a slightly higher quality score. Regarding publication year of studies, more than 76% were published after 2010. This is an indication of a high and increasing interest in RS research, same as reported in similar surveys like [87] or [88]. Regarding research problems (RQ2), the most frequently addressed was cold-start which mainly affects CF recommenders. It was followed by data sparsity that can affect any kind of RS. Both problems are attacked with a variety of data mining or matrix manipulation technique combinations or aggregation of extra user data and item features. Increasing accuracy or scalability and providing higher diversity in recommended items were other typical addressed problems.

A high variety of basic data mining or machine learning techniques and algorithms are serving as centric parts of hybrid recommenders. K-nearest neighbors is the most popular, especially as part of collaborative filtering implementations. Clustering algorithms with *K-means* the most popular are also highly utilized, especially in the preliminary phases when similar items or users are identified. In most of the cases, two recommendation strategies are mixed together, with CF-CBF being the predominant combination. The goal is to alleviate problems like data sparsity and cold-start from which both CF and CBF suffer a lot. With respect to hybridization classes (RQ4), *weighted* hybrids are the most popular, followed by

[5] http://www.qsrinternational.com/products.aspx

feature combination and *cascade* RSs. In many cases CF and CBF are put together through a weighting function. *Mixed* hybrids are the least studied and implemented.

Regarding RQ5 and application domains, movies recommenders are still the most common. This is partly because of the many experimentation movie datasets (e.g., those described in Section 4.1.3) that are publicly available. Moreover, Netflix $1M prize did certainly promote research and implementation of movie RSs in some way. Education and especially e-learning represent another common and interesting application domain. MOOCs (Massive Open Online Course) are gaining a lot of popularity. Also, education materials on the Web have been increasing dramatically in the last decade. Evaluation of RSs is not an easy task [89]. According to our findings, most of the studies perform evaluations by comparing their hybrid RSs with similar baseline recommenders. Datasets and various metrics such as *mean average error* and *root mean square error*, or information retrieval metrics like *precision*, *recall* and *F1* score are used for this process. Accuracy is still the most commonly assessed characteristic, followed by diversity of the recommendations. A highly desired characteristic that is reported as future direction (RQ7) is to have hybrid RSs that suggest items of different and changing domains (cross-domain recommenders). Another possibility is the increase of data utilization by parallelizing algorithms with *MapReduce* model as suggested in [90]. Other common future works that are reported include increasing personalization of recommendations, reducing their computational cost, etc.

4.2.3 Context-Aware Recommenders

Context-aware recommenders represent a family of RSs that are based on the notion of context. Recommendations and decision making, in general, are inherently related to contextual data. For example, current activity as the context has a significant influence on one's musical choices. Nobody expects the same type of music recommendations in different situations like working out, car driving, studying or going to sleep. Similarly, a restaurant menu recommended for a quick lonely lunch should differ from that of a business dinner with colleagues. Context as a concept has been defined in various forms, especially characterized by location and nearby people or objects. In [91] we find a complete and formal definition from Anid K. Dey: *"Context is any information that can be used to characterize the situation of an entity. An entity is a person, place, or object considered relevant to the interaction*

between a user and an application, including users and applications themselves." The information mentioned in the above definition is usually about (or comes from) various contextual factors like time, location, actual activity or occasion (above examples). They are aggregated with user and item information by means of the recommendation function. More formally, a CARS can be modeled as a function $f : U \times I \times C \rightarrow \mathbb{R}$, where the real-value item ratings are generated by aggregating context factors C with users U and items I [92]. From this formalization, we can distinguish three components: the input data (U, I, and C), the recommendation algorithm or function f and the output (recommendation list). Contextual factors as part of input data are sometimes not easily obtained and aggregated. This is especially true when they continuously change in time.

In fact, a highly desired characteristic from RSs is exactly the ability to adapt to quickly shifting user interests as a result of context change. The pervasive utilization of mobile devices has simplified the ability to obtain certain factors like time of day or location. However, other context objects such as people, occasion or goal remain difficult to interpret. Contextual data can be applied to the recommendation process in different phases and components. We can thus identify three types of CARS:

Contextual pre-filtering This is the term used to describe CARS in which contextual factors are used in data input phase. They guide input data selection or processing. This approach is usually simple and can be applied to other recommendation strategies to improve their performance.

Contextual modeling In this case, context is applied in the recommendation function as part of the rating prediction model. As a result, the function becomes highly complex. For this reason, this method cannot be applied to existing recommendation strategies.

Contextual post-filtering In this case, contextual factors are applied to output data (recommendation list). They are ignored in input data selection and ratings are initially predicted via traditional approaches. Afterwards, the initial set of ratings is corrected using the contextual factors. The correction may be performed through a filter or a rearrangement.

Performance of these contextualization approaches is highly dependent on the application. Usually prefiltering is simpler than the other two, and thus more suitable for non-critical performance requirements.

4.3 Mood-based On-Car Music Recommendations

Since the first in-car radios were introduced back in the 1930s,[6] music listening has been the favorite activity for most people while driving their cars. According to [93], roughly 70% of car drivers do it habitually. Various psychological studies report relations between background music and concentration, comfort or driving performance, providing evidence that music behaves as a stimulator that can have both positive and negative effects on mood and driving [94, 95]. They opened up several research possibilities that attempt to create relaxing car conditions for optimal car driving by means of proper music recommendations. In this section, we present the design of a mood-based recommender in the context of car driving. It tunes song recommendations using different sources of contextual data like driver's heart rate dynamics, his/her musical preferences, driving style obtained from telemetry data as well as location and time. The ultimate goal is to enhance driving comfort and safety by means of a proper music induction.

4.3.1 System Prerequisites

Important correlations between driver's mood and his/her driving patterns are highlighted in different studies. In [96] for example, authors reveal influence that negative moods like depression or anger have in driving cautiousness. Furthermore, in [94], they show correlations between an angry emotional state and aggressive driving style. Regulative efforts by music stimulation such as those of [97] have resulted somehow effective for a gradual shift in the mood of the driver. Several prerequisites are essential for building a car-based music mood recommender. Same as in Section 3.3, we had to pick up a model for representing emotion categories of songs. The only external feedback about song emotions that we could access was that of social tags. For this reason, we once again utilized the tag folksonomy of Table 3.2 and the emotion model of Figure 3.3. They are both practically convenient and highly compatible with the psychological model of Russell (Section 3.3.2).

Impact of music in one's mood is assessed by psychologists using a standard method called Musical Mood Induction Process (MMIP) which consists in replaying mood-eliciting tracks to participants. Authors in [98] describe various MMIP

[6] https://www.caranddriver.com/features/the-history-of-car-radios

methods such as behavioral measures, self-reports or physiological measures. For this project, we trusted on physiological data such as heart rate dynamics. Similarly, in [99], authors recognize emotions based on skin conductivity, skin temperature, and heart rate. Other contextual factors are driver's mood state and his/her current driving style. One approach is to include all contextual factors in the recommendation function (contextual modeling), obtaining a multidimensional model from the Cartesian product of the most relevant attributes [100]. In the case of our project the dimensions could be:

$$\begin{aligned} \textbf{\textit{Users}} &\subseteq \textit{UserName} \times \textit{Age} \times \textit{Gender} \times \textit{Profession} \\ \textbf{\textit{Items}} &\subseteq \textit{SongTitle} \times \textit{Artist} \times \textit{Genre} \times \textit{MoodLabel} \\ \textbf{\textit{Contx}} &\subseteq \textit{HeartRate} \times \textit{Place} \times \textit{Time} \times \textit{DriveStyle} \end{aligned}$$

Contextual parameters can be retrieved in different ways. Location and time, for example, are easily retrieved from the GPS and car dashboard respectively. For the heart rate, there are different cheap sensors like *Empatica* that can be used. Driving style can be inferred from car telemetric data by means of OBD-II technology which provides diagnostic information about the car. Nowadays, a plethora of OBD-II adapters are made available in the market. They include APIs to mobile applications and are easily integrated with the quickly growing on-car infotainment dashboards which are connected to the Internet [101].

4.3.2 System Architecture

Figure 4.1 shows the entire parts of the system connected together. The central module is the music recommender which is interconnected with all other parts. Its role is to generate the appropriate playlist of tracks that will be suggested to the driver. The recommender takes in various types of data such as contextual factors (e.g., time or location), driving style patterns, driver's mood state, and emotionally labeled songs. User mood recognition module obtains driver's mood data (heart rate dynamics) from wearable sensors (Empatica). To recognize mood state of the driver, the system presented in [95] is implemented. That system is based on cardiovascular dynamics (heart rate variability) observations on short-time emotional stimuli. Authors have used as emotional model the Circumplex Model of Affect which is very similar to the one we adopted. Two levels (high and low) of *arousal*

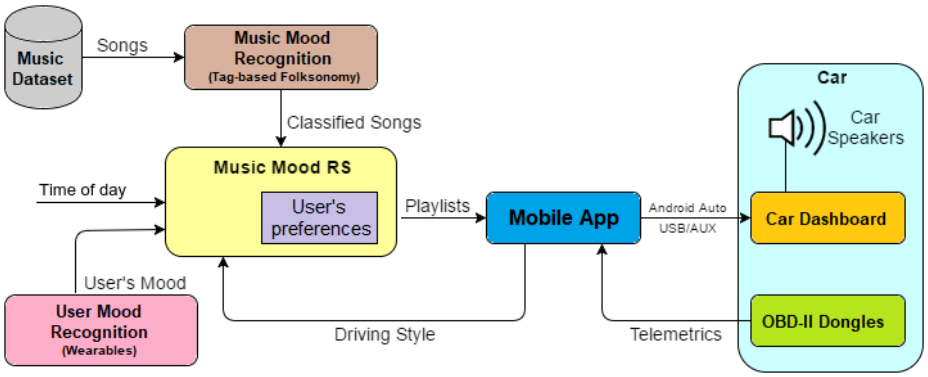

Fig. 4.1 Holistic view of the system

and *valence* that correspond to to one of the four emotional categories are recognized and transmitted to the main module (the recommender).

OBD-II module generates car telemetry data from which driving style patterns are extracted. Driver's aggressive patterns are identified by computing the jerk (the first-order derivative of acceleration) and considering car acceleration profile. To discriminate between calm and aggressive driving styles, a heuristic threshold of jerk is utilized. We considered as jerk threshold the one provided by [102] that was derived as average driving jerk value on a number of driving cycles of typical scenarios. As a result we obtain *aggressive* style when the actual jerk is greater than the threshold. Driving style goes as a flag to the mobile application and is used inside the recommender to affirm or dissent the mood state of the wearable sensor. Usually, and aggressive driving is associated with high levels of arousal and/or an angry emotional state. In the top left part of the scheme, we also see the music mood recognition module. It is responsible for the emotional annotation of songs retrieved from public datasets such as Million Song Dataset.[7] Social tags collected from *Last.fm* as well as the folksonomy of Table 3.2 are used for this process. Each song receives a label that may be *happy*, *angry*, *tender* or *sad* based on the planar model shown in Figure 3.3. More details about the different modules of the system can be found in [103].

[7] https://labrosa.ee.columbia.edu/millionsong/

4.3.3 Recommender and Mobile Application

After obtaining all contextual data, the recommender has to generate the appropriate playlist for the driver. Besides the modules (and data) described above, there is also another feature: time of day. We assume that there is no need for extra arousal during the day and consider "tender" (relaxing) as the default target mood category. Contrary, during a night drive it is better to avoid sleepy state of the driver by recommending *happy* (more aroused) music. The goal of the system is to maintain a relaxed state of the driver with song recommendations of his/her taste. The recommender takes in also the driving style which can be *aggressive* or *normal*. When the user is already in relaxed mood and driving style is normal, priority is given to past musical preferences. Otherwise, relaxing music is displayed. The recommendation list goes to the mobile application. This is an Android mobile

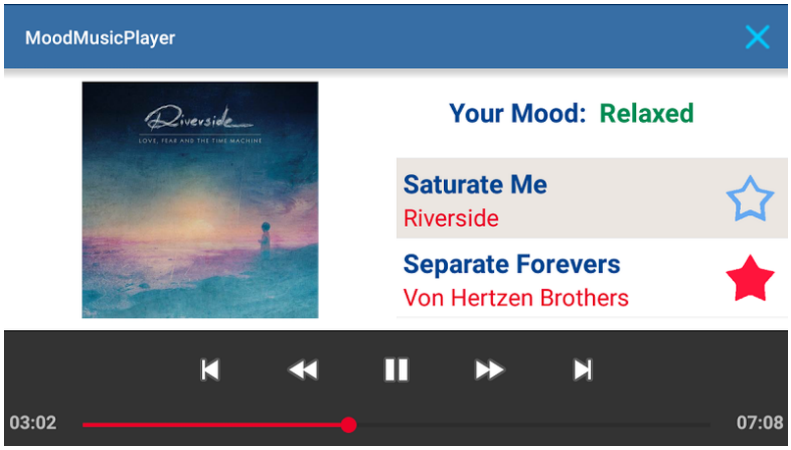

Fig. 4.2 Iterface of song recommendations

application that provides a simple user interface and enables media playback. Since it is supposed to be used in the car environment, it was programmed to be compatible with the Android Auto platform for the in-car streaming. The phone needs to be connected to the car dashboard via AUX/USB cable for music playback through car speakers. The user interface of the application is designed not to cause distractions to the driver. There is also a button that enables the user to express appreciation for the recommended songs of the list. Application interface is presented in Figure 4.3. The user is free to select which song to play from the recommended list (Figure 4.2). If no selection is performed the top-ranked song starts automatically.

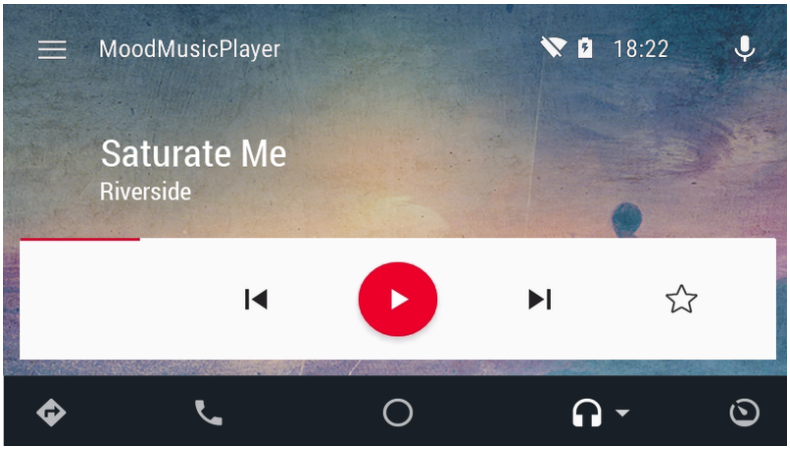

Fig. 4.3 Interface of mobile application

Chapter 5

Distributed Word Representations

"The beginning of wisdom is the definition of terms."
– Socrates

Distributed word feature representations known as word embeddings are generated training shallow neural architectures with huge text bundles for learning word relation predictions. Several neural architectures have been designed for that purpose. Some of them are quite efficient and produce vectors that are able to retain semantic and syntactic similarities between words, making them applicable in tasks like topic modeling or sentiment analysis. Performance of those feature vectors depends on various factors like training method and parameters, size and vocabulary of source texts, the thematic relevance of application domain with that of source texts, etc. Experimental observations reveal interesting relations between the influencing factors and performance of word embeddings on each task. For example, sentiment analysis of song lyrics and movie reviews seems to be more sensitive to corpus size than to other factors like the thematic relevance of texts.

This chapter presents obtained results from various comparative experiments on sentiment analysis tasks with word embeddings as dense text feature representations. Section 5.1 introduces local (traditional) and distributed text representation models, highlighting their advantages and disadvantages. Details of the most popular word embedding generation neural architectures are described in Section 5.2. Finally, Section 5.3 presents and further discusses the empirical results that were reached.

5.1 Word Representation Models

Distributed word representations generated from neural language models are replacing Bag-Of-Words (BOW) representation in various text analysis applications. BOW has been traditionally recognized for its simplicity and efficiency. It was first proposed in [104] where the authors discuss the possibility of describing languages via distributional structures (e.g., in terms of co-occurring parts). Sometimes Set-Of-Words (SOW) representation is used, where each word of vocabulary V is counted only once and its presence or absence is encoded and used as a feature. Both BOW and SOW are considered as discrete representations where each word is encoded with a binary or frequency number. Other vectorization and scoring methods like *term frequency-inverse document frequency* (*tf-idf*) are also popular. In fact, BOW combined with *tf-idf* have been successfully applied in many text classification studies, especially in combination with support vector machine used as classifier [24]. SOW is also an example of localist representations, in the sense that it allocates a unit of memorization for every word in all documents that word appears in.

The main problem with BOW and SOW is their poor scalability with respect to vocabulary size V. Every word is encoded in a sparse vector of a V-dimensional space. As vocabulary V can grow to hundred thousands of words, data sparsity becomes a serious issue. Another problem is the very high feature dimensionality (again with respect to large V) that results, leading to overfitting (the infamous *curse of dimensionality* problem). Furthermore, BOW representation is not able to conserve order of text words. For example, the phrase "excellent and not expensive service" has same representation with the phrase "expensive and not excellent service". The former expresses a *positive* opinion whereas the latter a *negative* one. This problem causes performance degradation on sentiment polarity analysis tasks. From the linguistic point of view, BOW is unable to retain semantic relations of words. For example, words "boy" and "girl" are semantically related (gender, human beings) wheres their corresponding vectors are orthogonal.

Word embeddings trained from neural networks on large text corpora were invented to solve the above problems. They are examples of continuous space representations where every word is encoded to a D-dimensional (typically 100 – 300) vector of real (continuous) values. They are also called distributed in the sense that every word vector is stored only once and shared in all documents containing that word. The main difference with BOW is the fact that D is fixed and independent

of V. As a result, word embeddings offer dense data representations of reduced dimensionality even when vocabulary size is very big. Moreover, studies like [26] or [28] that present Skip-Gram and Glove methods, also confirm that word embeddings trained from large text corpora are able to preserve syntactic and semantic word relations. They test this property by means of word analogy tasks and report very good results. It is, however, important to note that word feature quality depends on training data, number of word samples and size of vectors. It takes a lot of computation time to obtain high-quality representations. The following sections describe in details some of the most popular training methods that are available today.

5.2 Popular Word Vector Generation Methods

5.2.1 Continuous Bag of Words

CBOW architecture proposed in [26] is a simplification upon the feed-forward neural model of [5]. The hidden layer that introduces non-linearity is removed and the input window of Q words is projected into a P-sized projection layer. Q future words are used as well and the objective is to correctly predict the middle word. They use a log-linear classifier with a binary tree representation for the vocabulary. This way the number of units in the output layer drops from V to $log_2(V)$. As a result, the total training complexity of CBOW becomes:

$$C = E \times T \times (Q \times P + P \times log_2(V)) \tag{5.1}$$

where E is the number of training epochs and T is the total number of tokens appearing in the training text bundle. Throughout this thesis, the term “token” is used to indicate raw words that do usually repeat themselves within a text document or text bundle. Unique words that form the vocabulary of that text structure are called “vocabulary words” or simply “words”. The architecture of CBOW is schematically presented in Figure 5.1 (a) and shows the use of distributed context words to predict the current word.

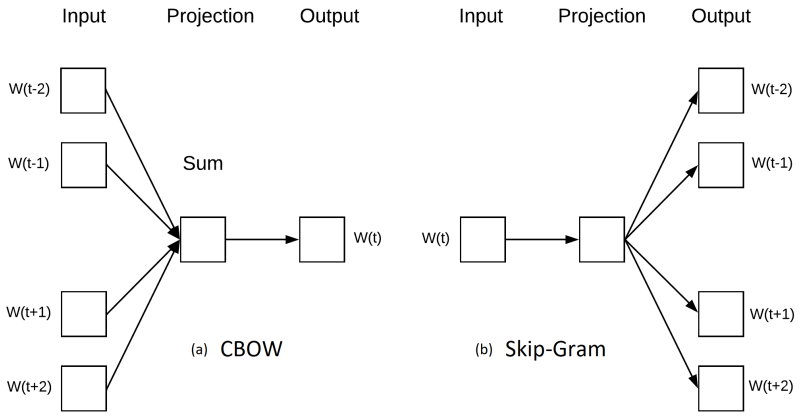

Fig. 5.1 CBOW and Skip-Gram neural architectures

5.2.2 Skip-Gram

Skip-Gram architecture shown in Figure 5.1 (b) is similar to that of CBOW. However, it starts from the current word and predicts context words appearing near it [26]. A log-liner classifier with projection layer takes the current word as input and predicts nearby words that appear before and after (inside a window) the current word. The training complexity of this architecture is

$$C = E \times T \times (W \times (P + P \times log_2(V))) \tag{5.2}$$

where W is the word window size. Authors report that enlarging the size of the window enhances the quality of generated vectors but also increases computation cost as suggested from Equation 5.2. For benchmarking vector quality of different architectures, they create the analogical reasoning task that consists of analogies like *Italy* : *Rome* == *France* : ___. These tasks are solved by finding the vector of a word x (in this case x is *Paris*) such that $vec(x)$ is closest in cosine distance to $vec(Italy) - vec(Rome) + vec(France)$. Besides semantic analogies, the task dataset also contains syntactic analogies as well (e.g., *walk* : *walking* == *run* : *running*) and is available online.[1] Authors generated a collection of word vectors trained on a huge Google News corpus of 100 billion tokens and released them for public use.[2]

[1] https://code.google.com/archive/p/word2vec/source/default/source
[2] https://code.google.com/archive/p/word2vec/

5.2.3 Glove

Glove (GLObal VEctors) described in [28] is a log-bilinear regression model that generates word vectors based on global co-occurrences of words. Authors argue that probability ratios of word co-occurrences can be used to unveil aspects of word meanings. For example, they observe that $P(solid|ice)/P(solid|steam)$ is about 8.9 whereas $P(gas|ice)/P(gas|steam)$ is only 0.085. This is something we logically expect, since "solid" is semantically more related with "ice" than it is with "steam". Similarly, "gas" is closer to "steam" than it is to "ice". Based on these premises, they build a weighted least square regression model that learns word vectors by means of word-word co-occurrence statistics. The calculations on real text corpora show a complexity of $O(|T|^{0.8})$ for the model, where T is again the total number of tokens appearing in the train texts. To evaluate the quality of word vectors, authors create various big datasets of varying contents.[3] They also utilize the word analogy task described above and report that Glove performs slightly better than other baselines such as CBOW or Skip-Gram, especially with larger training corpora. Moreover, its scalability provides substantial improvements from further increase of training corpus size.

5.2.4 Paragraph Vectors

A common limitation of all above word vector generation methods in text mining applications is the fact that they produce fixed-length vectors for words but not for variable-length texts like sentences or paragraphs. Many text datasets contain documents that have different lengths. As a result, they must be preliminarily clipped and/or padded to a fixed length. To overcome this limitation, authors in [105] propose *Paragraph Vector*, a method for learning fixed-length continuously distributed representations of variable-length text excerpts such as sentences, paragraphs or entire documents. Same as in CBOW, paragraph vectors contribute to the prediction of the next word (following few words in a fixed window) using the context words sampled from the paragraph. Paragraph and word vectors inside each paragraph are concatenated to predict the next word. As soon as the word and paragraph vectors are obtained from training texts (training phase), vector prediction for unseen paragraphs is performed (inference phase). Authors report significant improvements

[3] https://nlp.stanford.edu/projects/glove/

when comparing with BOW representation on supervised sentiment analysis of short texts like sentences. For longer documents like movie reviews, accuracy gains of paragraph vectors are lower.

5.3 Performance of Word Embeddings on Sentiment Analysis Tasks

The purpose of the conducted experiments with word embeddings was to observe the role that various factors like training method, size of training corpus and thematic relevance between training texts and analyzed documents might have on sentiment analysis prediction accuracy. To this end, three research questions were posed:

RQ1 *Is there any observable difference in performance between Skip-Gram and Glove word embeddings?*

RQ2 *What is the role of training corpus size in the performance of the generated word embeddings on sentiment analysis tasks?*

RQ3 *How does thematic relevance of training texts influence behavior of word embeddings on sentiment analysis of different text types?*

The following sections present the experimental resulst and the concluding remarks that answer these questions.

5.3.1 Word embedding models and corpora

In [106] we contrasted word embedding models trained with text corpora of various attributes. Some of those models are publicly available and two of them were trained by us (Text8Corpus and MoodyCorpus). The full list with some basic attributes is presented in Table 5.1. Wikipedia Gigaword combines Wikipedia 2014 dump with Gigaword 5,[4] obtaining a total of six billion tokens. Authors of [28] created it to evaluate Glove performance. There are 400,000 unique words inside, trained from context windows of ten words to the left and ten other words to the right. The

[4] https://catalog.ldc.upenn.edu/LDC2011T07

Table 5.1 List of word embedding corpora

Corpus Name	Training	Dim	Size	Voc	URL
Wiki Gigaword 300	Glove	300	6B	400K	link
Wiki Gigaword 200	Glove	200	6B	400K	link
Wiki Gigaword 100	Glove	100	6B	400K	link
Wiki Gigaword 50	Glove	50	6B	400K	link
Wiki Dependency	Skip-Gram	300	1B	174K	link
Google News	Skip-Gram	300	100B	3M	link
Common Crawl 840	Glove	300	840B	2.2M	link
Common Crawl 42	Glove	300	42B	1.9M	link
Twitter Tweets 200	Glove	200	27B	1.2M	link
Twitter Tweets 100	Glove	100	27B	1.2M	link
Twitter Tweets 50	Glove	50	27B	1.2M	link
Twitter Tweets 25	Glove	25	27B	1.2M	link
Text8Corpus	Skip-Gram	200	17M	25K	link
MoodyCorpus	Skip-Gram	200	90M	43K	link

four versions that were derived differ in vector sizes only having vectors of 50, 100, 200 and 300 dimensions. Wikipedia Dependency is a bundle of one billion tokens crawled from Wikipedia. It was trained using a slightly modified version of Skip-Gram presented in [107]. Authors have experimented with several syntactic contexts of words. They filtered out every word with a frequency lower than one hundred, reaching to a vocabulary size of 175,000 words and 900,000 syntactic contexts. Authors report that the additional contexts help in producing word vectors that exhibit better word analogies. Google News is one of the largest and richest text sets that are available. It was trained on 100 billion tokens and contains three million distinct words and phrases. [26]. Skip-Gram was used with context windows of five words, generating vectors of 300 dimensions. Reduced versions of the corpus were also used in [108] for validating the efficiency and training complexity of CBOW and Skip-Gram methods.

Common Crawl 840 is even bigger than Google News. It was trained using Glove on 840 billion tokens, resulting in 2.2 million unique word vectors. It comprises texts of Common Crawl,[5] a company that builds and maintains public datasets crawled from the Web. Common Crawl 42 is a smaller version trained on 42 billion tokens and has a vocabulary size of 1.9 million words. Both Common Crawl 840 and Common Crawl 42 contain word vectors of 300 dimensions. The collection of

[5] http://commoncrawl.org

Twitter tweets was also trained with Glove. The training bundle was made up of texts from two billion tweets, totaling in 27 billion tokens. A total of 1.2 million unique word vectors were generated. The four collections of embeddings that were produced contain vectors of 25, 50, 100 and 200 dimensions each. Text8Corpus was used to observe the role of corpus size in the quality of generated embeddings. It is a smaller text collection consisting of 17 million tokens and 25,000 words only. Text8Corpus was trained with Skip-Gram method and various parameters. The last model is MoodyCorpus,[6] a text bundle of song lyrics created following the work in [69]. Its biggest part is the collection of Million Song Dataset.[7] Songs texts of different genre and epoch were added to have more diverse texts. Regarding text preprocessing, punctuation, numbers or other symbols, as well as text in brackets were cleared out and everything was lowercased. The output consists of 90 million tokens and 43,000 unique words.

5.3.2 Sentiment Analysis Tasks

This section describes the sentiment analysis experiments with song lyrics and movie reviews that were conducted to observe patterns relating training method, corpus or vocabulary size and thematic relevance of texts with the performance of generated word vectors. The first set of experiments were conducted on sentiment polarity analysis of song lyrics. The dataset of songs described in [54] was utilized for one set of experiments. The original version contains 771 lyrics classified by three human experts. Here, a balanced version of 314 *positive* and 314 *negative* songs (A628) is used instead. The second set of lyrics experiments was conducted using MoodyLyrics (here ML3K), a collection of 3,000 lyrics that was created in [64]. It contains songs of different epochs, dating from the sixties to the current days.

The second experimentation task consists in identifying the polarity of movie review documents. Seminal work on movie reviews has been conducted by Pang and Lee in [109] and [110]. They published sentiment polarity dataset which contains 2,000 movie reviews labeled as positive or negative. They also constructed and released subjectivity dataset consisting of 5,331 subjective and 5,331 objective sentences using the same method. The work of Pang and Lee created the road-map for a series of similar studies which apply various techniques to the problem. Use

[6] http://softeng.polito.it/erion/MoodyCorpus.zip
[7] https://labrosa.ee.columbia.edu/millionsong/

of neural networks and word embeddings for analyzing movie reviews appeared on more recent works like [111]. In that paper, authors explore RNNs (Recurrent Neural Networks), and CNNs (Convolutional Neural Networks). Another very important work that was conducted is [112] where authors create a large dataset of 50,000 movie reviews crawled from Amazon IMDB. That dataset has become very popular among researchers, serving as a benchmark for various studies such as [113], [114], etc. Here we used a subsets of 10,000 (MR10K) reviews and the full set of 50,000 (MR50K) movie reviews for the experiments.

Every text of the four datasets (A628, ML3K, MR10K, and MR50K) was first cleaned and tokenized. The dataset of each experimentation set is loaded and a set of unique words is created. The 14 models of word embeddings are also loaded and a 15^{th} model (*self_w2v*) is trained with Skip-Gram using the corpus of the current set. Every line of the pretrained models is analyzed, splitting apart the word and the corresponding vector and building {word: vec} dictionary that is later used for producing the classification features. To have low variance in the obtained accuracy scores, we decided to use random forests with many (150) estimators as classifiers on each experiment. Classification models were prepared using tf-idf vectorizer which has been successfully applied in many relevant works like [115] or [116]. It has been shown to perform better than other schemes like boolean scoring (presence or absence of certain words) or term frequency alone [117]. Tf-idf was applied to both words (for semantic relevance) and their corresponding vectors (for syntactic or contextual relevance). Average 5-fold cross-validation scores are calculated and reported for each of the models.

5.3.3 Results and Conclusions

Figures 5.2 and 5.3 show 5-fold cross-validation accuracy results on the two song lyrics datasets. Obviously, the top three models are *crawl_840*, *twitter_50*, and *self_w2v*. On A628 (smallest dataset) it is the biggest model (*crawl_840*) that leads, followed by *twitter_50*. *Self_w2v* is positioned at the bottom of the list, severely penalized by the size of the A628 dataset it is trained on. The situation completely shifts in ML3K (larger dataset) where *self_w2v* comes at the top of the list, followed by *twitter_50*. We also see that other models like *google_news*, *wikigiga* and *dep_based* are positioned in the middle of the list. The worst models are those trained on Text8Corpus and MoodyCorpus bundles, with accuracy scores between

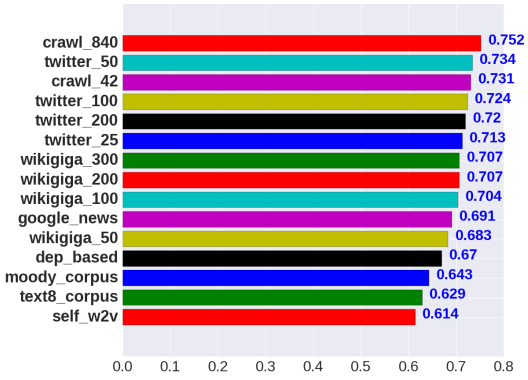

Fig. 5.2 Lyric accuracies on A628

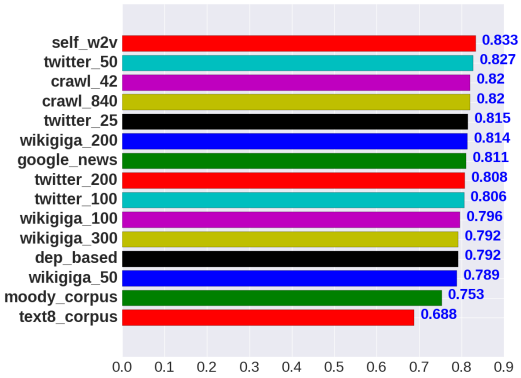

Fig. 5.3 Lyric accuracies on ML3K

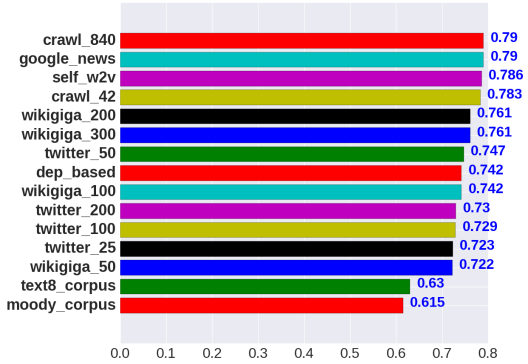

Fig. 5.4 Review accuracies on MR10K

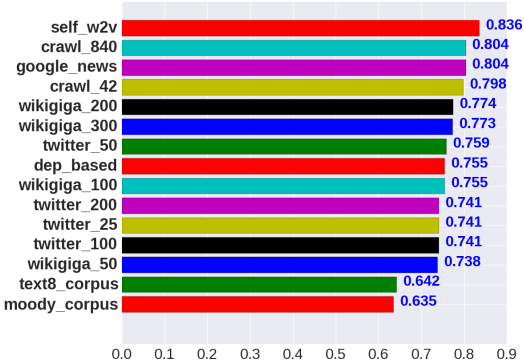

Fig. 5.5 Review accuracies on MR50K

0.62 and 0.75. In fact, *self_w2v* was generated from the texts of each experimentation dataset and thus depends on the size of that dataset. Its leap from the bottom to the top of the list with accuracy scores rising from 0.61 to 0.83 suggests that there is a strong correlation between corpus size and model accuracy. Top accuracy scores obtained here are similar to those reported in analogous works like [118] where a dataset of 1032 songs is utilized.

Accuracy results of the experiments on move reviews are presented in Figures 5.4 and 5.5. We once again see that *crawl_840* performs very well. Also, *google_news* is positioned among the top. Twitter models, on the other hand, are located in the middle of the list. We see that *self_w2v* grows considerably from the small to the bigger dataset, same as in the experiments with lyrics. In MR50K it has a discrete margin of more than 3% from the 2^{nd} position. Once again, *wikigiga* models are positioned in the middle of the list and the worst models are Text8Corpus and MoodyCorpus. They perform badly on this task with a top accuracy of no more than

Table 5.2 Google News compared with Common Crawl

Test	avg	t	p	Verdict
A628	0.691	5.683	0.00047	Hm_0 rejected
ML3K	0.811	4.913	0.00119	Hm_0 rejected
MR10K	0.79	1.462	0.182	Hm_0 not rejected
MR50K	0.804	1.547	0.161	Hm_0 not rejected

0.64 and a deficiency of 10% from the closest model. There are also other studies that analyse movie reviews. Some of them have obtained higher scores than those reported here. Authors in [112] for example, apply a probabilistic model able to capture semantic similarity between words and report an accuracy of 0.88. Also in [114], they combine bag-of-3-grams with a CNN of three layers and achieve an accuracy of 0.92. In [113] authors utilize a very similar experimentation setup with the one that was used here. They feed word vector features to a random forest classifier and achieve 0.84 accuracy on movie reviews.

To answer the three research questions, we had to check the statistical significance of the above results. Regarding RQ1 and training method, we set the following null hypothesis:

$\mathbf{Hm_0}$ *There is no significant difference between accuracy scores of word embeddings trained with Glove and those trained with Skip-Gram.*

For a fair comparison, we picked up results of Common Crawl 840 and Google News corpora. These two bundles were both created from web and news texts which are not of a particular topic. We performed t-test analysis, comparing the values of *google_news* model with those of *crawl_840* in the four experiments. An $\alpha = 0.05$ was chosen as the level of significance. They have similar vocabulary sizes and differ mainly in the method they were trained with. Statistical results are shown in Table 5.2. The *avg* values are basically the 5-fold cross-validation results reported in the figures above. As we can see, in the first two experiments (songs), t values are much greater than p values. Also, obtained p values are much smaller than the pre-chosen value of α (0.05). As a result, we have evidence to reject the null hypothesis and confirm that indeed *crawl_840* performs better than *google_news*. On the other two experiments, we have a totally different picture. Both models got same average scores and have similar t and p values, with the latter being considerably

greater than α. As a result, we do not see significant difference between them on movie reviews. In conclusion, we can say that word embeddings trained with Glove slightly outrun those trained with Skip-Gram on polarity analysis of song lyrics but perform the same on movie reviews.

Table 5.3 Properties of self_w2v models

Trial	Dataset	Dim	Size	Voc	Score
1	AM628	200	156699	8756	0.614
2	ML3K	200	1028891	17890	0.833
3	MR10K	300	2343641	53437	0.786
4	MR50K	300	11772959	104203	0.836

Regarding RQ2 and the effect of training corpus size, the results seem more convincing. Biggest models like *crawl_840* appeared among the best in every set of experiments whereas MoodyCorpus and Text8Corpus that are the smallest were always positoned at the bottom of the list. Moreover, *self_w2v* performed very well in the second experiment of each task where it was trained on the bigger datasets. The properties of *self_w2v* on each experiment are summarized in Table 5.3. Despite these arguments, we still conducted a statistical examination, formulating and checking the following null hypothesis:

Hs_0 *There is no significant difference between accuracy scores of word embeddings trained on text corpora of different sizes.*

The fairest comparison we could make here is the one between *crawl_42* and *crawl_840* models. They were both trained using Glove on Common Crawl texts. The only difference is in the size of corpora they were derived from. Statistical results are shown in Table 5.4. Experimental results on A628, MR10K, and MR50K

Table 5.4 crawl_42 compared with crawl_840

Test	avg	t	p	Verdict
A628	0.731	4.97	0.0011	Hs_0 rejected
ML3K	0.82	0.86	0.414	Hs_0 not rejected
MR10K	0.783	2.56	0.033	Hs_0 rejected
MR50K	0.798	2.96	0.027	Hs_0 rejected

indicate that the accuracy difference between the two models is significant and the null hypothesis can be rejected. Same is not true about results on ML3K. All in all, we can confirm that training corpus size has a strong influence on the performance of word embeddigns. Furthermore, it comes out that the choice between pretrained vectors and vectors generated from the training dataset itself also depends on the size of the latter. When the training dataset corpus is big enough, using it to generating word vectors is the best option. If it is small, it might be better to source word vectors from available pretrained collections.

Regarding RQ3 and thematic relevance, we also see that pretrained models behave differently on the two tasks. Twitter corpora performed better on lyrics. They are large and rich in vocabulary with texts of an informal and sentimental language. This language is very similar to the one that is found on song lyrics, with *love* being the predominant term (word cloud in [69], p. 5). Common Crawl and Google News that are trained with more informative texts performed best on movie review analysis. These results indicate that topic of training texts may have an influence on the generated word features. Nevertheless, it was not possible here to have a rigorous and fair analysis of it. Obviously, a more extensive experimental work that excludes the effect of other factors is required.

Chapter 6

Sentiment Analysis via Convolution Neural Networks

"You can use the power of words to bury meaning or to excavate it."
– Rebecca Solnit, *Men Explain Things to Me*, 2014

Dense vector representations of words are well interpreted from neural network layers. Convolutional and max-pooling neural networks, for example, are capable of capturing relations between words and sentiment categories of the document. They exhibit very good performance on text mining tasks when combined together. Before building sentiment analysis models based on neural networks, it is essential to know how we can adapt the large set of network hyperparameters to the varying sizes and document lengths of the training data. To address this issue, we performed several experiments where simple neural networks of stacked convolutional and max-pooling layers are trained with different dataset sizes and document lengths.

This chapter is organized as follows. Section 6.1 introduces convolutional neural networks and relevant studies that successfully apply them for text mining tasks. Training datasets, as well as data statistics and text preprocessing steps, are described in Section 6.2. Section 6.3 presents the neural network structure we experiment with, made up of stacked convolutional and max-pooling layers, as well as the hyperparameter setup alternatives. Finally, Section 6.4 summarizes and further discusses the obtained relations between data properties and performance.

6.1 Neural Network Types for Text Analysis

Neural networks are excelling in various complex tasks such as speech recognition, object detection, machine translations or sentiment analysis. Intelligent personal assistants, self-driven cars or chatbots passing Turing Test are examples of the deep learning hype. This success is mostly attributed to the ability of deep neural networks to generalize well when fed with tons of data. Convolutional Neural Networks (CNN) are particularly effective in mimicking the functioning of the visual cortex in the human brain, becoming great players on computer vision applications. The basic structure of CNNs was proposed by LeCun *et al.* in [119] and was utilized for recognizing images of handwritten digits. After a decade of lethargy (the second AI winter), they came back in the late 2000s, becoming the crucial part of several advanced image analysis architectures.

Figure 6.1 shows a neural network based on convolutional and max-pooling layers for classification of digit images. The convolution operation applies the filter over the bigger matrix of pixels moving forward at a certain stride and producing feature maps. This operation extracts features by preserving the spacial relations between them. Later on, a max-pooling operation is applied on the generated feature maps to downsample (reduce) their dimensionality by hopefully selecting the most relevant ones. It is illustrated in Figure 6.2. Afterwards, all feature maps are flattened (reducing dimensionality from 3 to 2) and pushed to the dense layer that serves as a simple classifier. The neural network of Figure 6.1 can be applied on text analysis

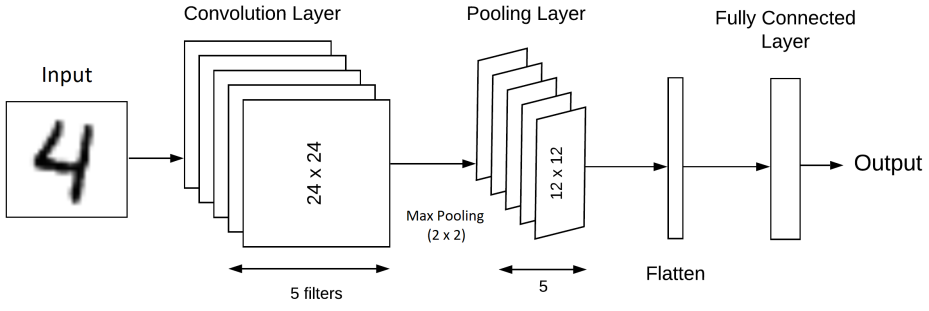

Fig. 6.1 Simple convolution-pooling network for image recognition

as well. The only difference is in the input data and target categories. In general, if we have a document of n words with vectors of d dimensions each, we can apply on it a filter (usually m filters) $w \in \mathbb{R}^{d \times k}$ on all windows of k consecutive words. The output is a feature map $f = [f_1, f_2, ..., f_{n-k+1}]$ that is passed to the max-pooling

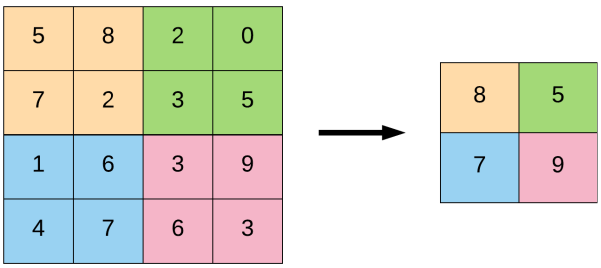

Fig. 6.2 Illustration of a 2 *x* 2 max-pooling operation

layer. If simple max-pooling is used, than we get only one $f' = max(f)$. Otherwise, if we apply regional max-pooling with regions of size R, the output we get will be a set of p max features $f' = [f'_1, f'_2, ..., f'_p]$, where $p = ceil(\frac{n-k+1}{R})$ is the number of the resulting regions.

Recurrent Neural Networks (RNN) represent another network family that has gained significant popularity in the recent years. They perform especially well with data that exhibit continuity in time, like words coming one after another as a sequence. This family of neural networks makes use of feedback loops that behave like "memory" for maintaining the already seen data. Long Short-Term Memory (LSTM) networks introduced in [120] are probably the most popular and highly utilized version of RNNs. However, the common problem of RNNs is their limited memorization ability. In practice, they can look back to few steps and are thus not able to represent well long text sequences. Furthermore, RNNs are slower to train compared with CNNs which are simpler and faster. Examples of sentiment analysis studies utilizing RNNs are [121] or [122] where they make use of gated networks. Also, similar studies like [123] employ various combinations of CNNs with RNNs to achieve the same goal.

Application of CNNs for text analysis was initially impeded by problems like small available text datasets, high dimensionality of word features etc. One of the first works that successfully used them for sentiment analysis was proposed by Kim in [124]. A basic convolutional network was applied on short sentences to extract and select word features and a simple dense layer was used as a classifier. The author reported state-of-the-art accuracy results with little computation load on datasets containing from 3375 to 11855 documents. Other studies like [125] and more recently [126] used deeper architectures of 9 – 29 layers, starting from characters and building up word patterns utilized as classification features. They report competitive

results on large datasets with hundred thousands of text documents. However, on smaller datasets, their networks are weaker than shallow counterparts which start from words as basic language elements. The debate of shallow word-level versus deeper char-level CNN-based networks was further disputed in [127]. Authors try shallow word-level CNNs on same big datasets used in [125] and [126]. They report that shallow word-level CNNs are not only more accurate but also compute faster than the deeper char-level CNNs. Obviously text words do comprise a semantic value that is lost when analysis starts from characters. The only drawback of word-level CNNs is their high number of parameters required for the word representations, which counts for higher storage requirements. Here (and in the next chapter) we favor and experiment with simpler word-level CNNs.

It is important to note that applying convolutional and pooling neural layers for analyzing sentiment polarity of texts is not easy. For example, suppose we are using the simple architecture of Figure 6.1 with length (depth) $L = 2$ made up of only one convolutional and one max-pooling layer. In this simplified scenario, we still have to pick many parameter values such as number of filters m, filter (kernel) length k, filter stride s, pooling region size R, dropout, L_1 and L_2 regularization norms to avoid overfitting as well as other parameters or functions used to train the network. This high number of parameters makes it hard to find the optimal performance setup. Moreover, performance and network parameters are strongly related with input data metrics such as the size of the training set, length of the document etc. For this reason, in [128] we conducted multiple experiments that relate data properties and network parameters with top classification accuracy. The goal was to observe any patterns that could be useful for simplifying optimal network creation by providing template parameter setups, answering the following questions:

RQ1 *What is the relation between pool region size (R) and training text length with respect to optimal accuracy score?*

RQ2 *What are the effects of convolutional kernel size (k) and network width (W) on accuracy score?*

RQ3 *What is the relation between network depth (L) and training dataset size with respect to optimal accuracy score?*

In the forthcoming sections, we describe the steps that were followed, obtained experimental results and some concluding remarks addressing the three research questions listed above.

6.2 Data Processing and Statistics

6.2.1 Experimental Datasets

For the experiments, public datasets of various sizes and document lengths were chosen. They comprise text excerpts of different thematic contents such as song lyrics, movie reviews, smartphone reviews etc.

Mlpn This dataset is the collection of song lyrics from MoodyLyricsPN dataset described in Section 3.3. There are 2,500 *positive* and 2,500 *negative* labeled songs in it. All lyrics were crawled from online music portals and are used for experimental purposes only.

Sent Sentence polarity dataset was one of the first experimental text collection, created by Pang and Lee back in 2005 [129]. There are 5331 *positive* and 5331 *negative* texts extracted from IMDB archive of movies. The reviews are short and consist of one sentence and about 10 – 20 words only.

Imdb This is the IMDB dataset of movie reviews described in [130]. It contains 50,000 movie reviews that were manually labeled as *positive* or *negative* and has been used as ground truth in many sentiment analysis studies. The dataset is also available in various libraries, preprocessed and ready for use.

Phon This dataset contains user reviews of unlocked smartphones sold in Amazon. Users usually provide text comments, a 1–5 star rating or both for products they purchase. The dataset contains entries with both star rating and text description. All reviews with a 3-star rating were cleared out to better separate 1-star and 2-star reviews which were considered as *negative*, from 4-star and 5-star reviews that were considered as *positive*. A final number of 232,546 reviews was reached.

Yelp This is the biggest dataset that was used. It was created by Zhang *et al.* in [131] and contains 598,000 Yelp user reviews about businesses such as restaurants,

hotels, etc. All reviews were balanced and labeled with the corresponding emotional polarity.

It is important to note that the first three datasets were created using systematic methods involving human judgment for the labeling process. They can be thus used as ground truth for cross-interpretation of results. The same thing is not certainly true about the last two which are labeled considering user star ratings only. In fact, there is no proven guarantee that few stars do necessarily denote *negative* language and many stars a *positive* one. All affect datasets created by Zhang *et al.* in [125] (here we use Yelp only) are generated in that way. They are being used in several studies (e.g., [132] or [127]) which do not acknowledge that limitation. While it is highly desirable to have big labeled datasets for experimentation, creating them based on simplistic heuristics like star ratings only is probably not enough and appropriate data quality examinations are required. In fact, lack of big and professionally labeled text datasets was one of the reasons why we chose to focus on shallow word-level network architectures.

Table 6.1 Document length statistics for each dataset

Dataset	Docs	MinLen	AvgLen	MaxLen	UsedLen
Song Lyrics	5K	23	227	2733	450
Sentence Polarity	10K	1	17	46	30
Movie Reviews	50K	5	204	2174	400
Phone Reviews	232K	3	47	4607	100
Yelp Reviews	598K	1	122	963	270

6.2.2 Preprocessing and Statistics

The usual text preprocessing steps were performed on each document of the five datasets. Firstly, any remaining html tags (many texts were crawled from websites) were removed. Smiley symboles like :D, :-), :), :(, :-(, :P, were kept in, as they are very effective for emotion identification. Stopwords are usually discarded as they contain little or no semantic value. We cleared out the subset {“the”, “these”, “those”, “this”, “of”, “at”, “that”, “a”, “for”, “an”, “as”, “by”}. Presence of short form residues such as “ll”, “d”, “s”, “t”, “m” and negation forms like “couldn”, “don”, “hadn” or “didn” can completely shift the emotional polarity of a phrase or

sentence. They were thus kept in. Finally, any remaining "junky" characters were removed and everything was lowercased.

Having every document in its final form, we further observed the length distributions summarized in Table 6.1. As we can see, song text lengths fall between 23 and 2733 words, averaging at 227. Smartphone and movie review lengths are highly dispersed. The former range from 3 to 4607 with an average of 47. The later fall between 5 to 2174 averaging at 204. Regarding Yelp reviews, we see that they are less dispersed spanning between 1 and 963 words. The dataset of sentence polarities is the most uniform, with document lengths from 1 to 46. A more detailed observation of length distributions revealed that most documents are quite short. For example, in review datasets, there are very few documents longer than 500 words. For this reason, as well as to decrease computation load of the experiments, we clipped the few long documents and padded the shorter ones with zeros to reach uniform lengths. Actually, no more than 10% of documents were clipped. The last column of Table 6.1 reports experimentation lengths of each dataset.

6.3 Experimental Setup

6.3.1 Representation of Words

As explained in Section 5.1, word embeddings or distributed word representations in general, are gradually replacing the bag-of-words model in many applications. They are dense and well suited for the common neural network structures that are used today. The ability to retain syntactic and especially semantic word relations makes them appropriate for representing documents on text mining tasks. For example, we can assume that for every possible word in our dataset documents, there is a word vector of d dimensions generated with one of the methods described in Section 5.2. Then we can represent each of those documents as a matrix of numbers. The document representation becomes very similar to a matrix of pixels that represents an image. If we have $d = 5$ (in practice 100 – 300), the representation of a short sentence like "your shirt looks nice" will be as shown in Figure 6.3. One possibility for obtaining word embeddings is to initialize them with random values and then utilize the neural network for tuning their values based on the appearance of the words in the training dataset. This method can be applied when the supervised

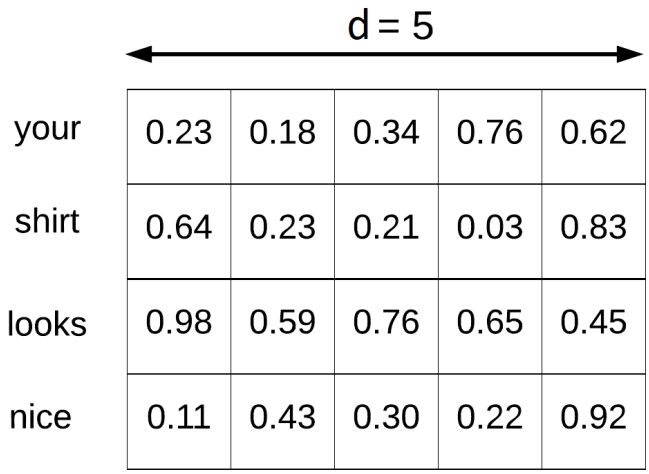

Fig. 6.3 Matrix representation of a short sentence

training dataset is enough large. As suggested in [106], when relatively small labeled text sets are available, sourcing pretrained word vectors that were generated from big text corpora gives better results. The first three datasets of Table 6.1 are relatively small. For this reason, we decided to utilize the word embeddings of 300 dimensions that are available in GoogleNews[1] collection. They were created from a text bundle of news documents containing 100 billion tokens and three million unique words and phrases. Relevant studies like [133] or [124] report excellent results when using them on similar tasks.

6.3.2 Data-driven Experimentation Networks

The basic network structure chosen for our experiments is presented in Figure 6.4. The embedding layer is actually not trainable. It just uses the corresponding word vector sourced from GoogleNews collection, for every word appearing in the documents. Next come the convolutional layers that work as feature extractors. There is one stack with W of them that are used in parallel, with filter size $k = 1$ for extracting word features, $k = 2$ for 2-gram features, $k = 3$ for 3-grams and so on. In fact, the added value of 2-gram and 3-gram features in performance have been pointed out in various studies like [134]. Each convolutional layer applies 70 filters to capture relations between feature maps and the *positive* and *negative* document categories. Regarding activation function of convolutions, $relu(x) = max(0,x)$, $tanh(x) = \frac{e^{2x}-1}{e^{2x}+1}$

[1] https://code.google.com/p/word2vec/

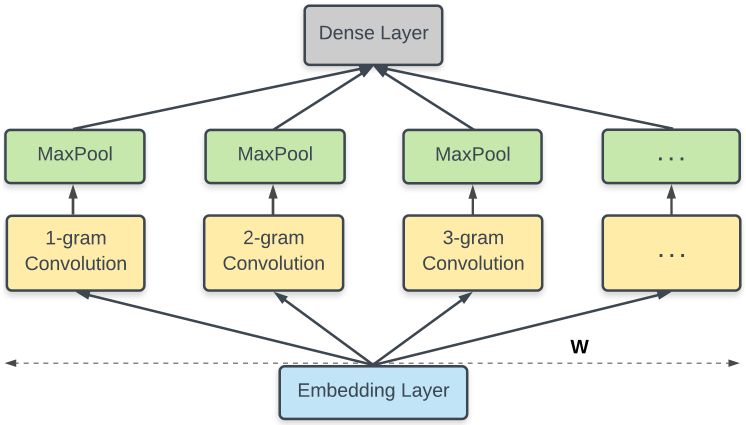

Fig. 6.4 Basic neural network structure

and $softsign(x) = \frac{x}{1+|x|}$ were explored. The best in most of trials was *relu* which is the one we used. Convolutional layers are directly followed by max-pooling layers that subsample data, selecting the most salient features. For retaining local information of word combinations, regional max-pooling with varying region size is used. Studies like [135] and [126] show that it outperforms other pooling methods such as k-max or average pooling. Generated feature maps of length $p = ceil(\frac{n-k+1}{R})$ are recombined together (flattened) and fed to a dense layer that serves as classifier. Length (depth) of the network L is the total number of convolutional and max-pooling layer stacks. As we can see, the network of Figure 6.4 has length $L = 2$. For the classification, a single dense layer of 80 nodes was used in each experiment. Overfitting was mitigated using 0.1 L_2 regularization and 0.35 dropout norms. Finally, to compute loss and optimize network training, binary cross-entropy and Adam optimizer were employed.

In Figure 6.5 we can see a planar projection of the training data length and size for each of the five datasets that were utilised. To find the optimal network structure and observe the role of dataset attributes in performance, alternative versions of the scheme in Figure 6.4 (one stack of convolution and max-pooling layers) were considered. The parameters of these network structures are driven by training data characteristics. For example, different values of the region size R were used for adapting the network to long and short documents of the datasets. Usually bigger datasets (e.g., the last two of Table 6.1) are better interpreted using deeper neural networks with more training parameters. As a result, a deeper network structure is obtained by duplicating the convolutional and max-pooling stacks in the same order.

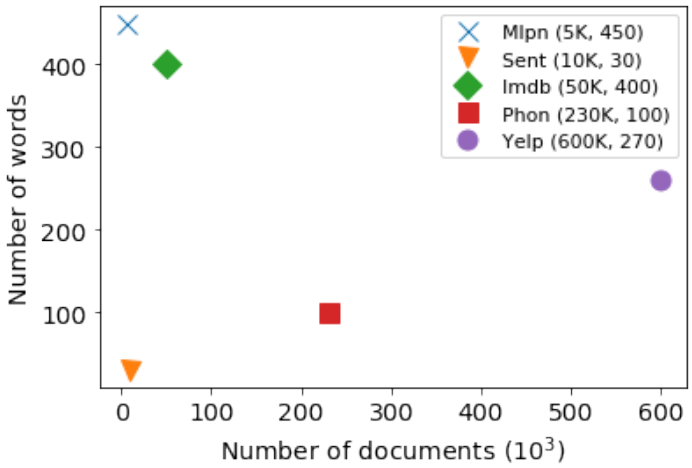

Fig. 6.5 Size-length distribution of training datasets

The rest of the networks that were tried are similar, changing only in max-pooling region sizes. During each experiment, a 70/10/20 percent data split for training, development, and testing was used respectively. The following section presents the obtained accuracy scores.

6.4 Results and Discussion

Top neural networks and their accuracy score on each dataset are summarized in Table 6.2. Simple network structures with just one or two consecutive convolutional and max-pooling stacks produce very good results. Top score on phone reviews (Phon) seems excellent. Contrary, the one obtained on song lyrics (Mlpn) seems somehow disappointing. Unfortunately, we still do not have a comparison basis for these two datasets. On sentence polarity dataset (Sent), a peak score of 79.89% was obtained. The many studies that have experimented with this dataset report accuracy scores from 76% to 82%. The very best result we found in the literature is 83.1%. It was reported by Zhao *et al.* in [136]. In that study, they proposed a self-adaptive sentence model that forms a hierarchy of representations from words to sentences and then to entire document. Their model is implemented using gating networks. On movie reviews (Imdb), an accuracy of 90.68% was reached. Top score on literature is 92.23% and was reported by Johnson and Zhang in [137]. Same authors report a

peak score of 97.36% on Yelp reviews (top score reached here is 94.95) in [132]. In both studies, they utilized deep and highly complex neural network models.

Table 6.2 Accuracies of top five network structures

Network	Mlpn	Sent	Imdb	Phon	Yelp
Conv-Pool (R=4, L=2)	72.24	**79.89**	87.98	95.31	92.32
Conv-Pool (R=25, L=2)	**75.63**	74.46	90.12	95.15	93.51
Conv-Pool (R=16, L=4)	73.34	75.08	89.87	**96.57**	94.86
Conv-Pool (R=25, L=4)	75.44	74.22	**90.68**	95.64	93.84
Conv-Pool (R=27, L=6)	71.88	74.04	89.11	95.21	**94.95**

It is important to note that the goal of the experiments presented in this chapter was not to obtain record-breaking results. The top-performing results we compared with were obtained from complex neural networks with millions of parameters. Contrary, the last model of Table 6.2 (the most complex one) has fewer than 200 thousand parameters. From that table, we can notice interesting patterns that can answer the three research questions posed at the end of Section 6.1. Regarding RQ1, we see that the three datasets of longer documents (Imdb, Mlpn, and Yelp), perform better on networks with bigger max-pooling regions. Their top scores are reached with aggregate downsampling coefficients $R = 5\ x\ 5 = 25$, $R = 25$ and $R = 3\ x\ 3\ x\ 3 = 27$ respectively. Contrary, Sent and Phon datasets that contain shorter texts reach their peak scores on networks with smaller values of R. In fact, R is the parameter that dictates length p of the final feature maps ($p = ceil(\frac{n-k+1}{R})$). According to the results, highest accuracy scores are always reached with p values that are within $7-15$.

Regarding the role of filter size (RQ2), convolutions of $k = 1, k = 2$ and $k = 3$ were essential for optimal scoring. Omitting one of them reduced accuracy. On the other hand, wider networks with convolutions of filters $k = 4$ or $k = 5$ did not perform any better. For this reason we stopped at $W = 3$. Regarding RQ3 and network depth, we see that vertical expansion with $L = 4$ improves results for the two bigger datasets Phon and Imdb. They play better with the structures of four convolutional-pooling stacks. Moreover, Yelp which is the biggest dataset reaches optimal performance on a deeper network of six stacks of convolutional and max-pooling layers. Aiming towards simplification and low training times, we did not try deeper networks ($L > 6$) for further improvements. Sent and Mlpn that are the

smallest datasets, reach their peak scores on the simpler networks of two stacks only ($L = 2$). This is something intuitive since deeper networks are more data hungry and tend to overfit small datasets. Obviously, other layer combinations and network architectures should be explored to obtain top-notch results. In the next chapter, we direct our efforts precisely on this issue.

Chapter 7

A Neural Network Architecture for Sentiment Analysis Prototyping

"If you torture the data long enough, it will confess."
– Ronald Coase, Nobel Prize in economics

As described in the previous chapter, using neural networks for feature extraction, feature selection, and sentiment polarity prediction of texts produces very good results. However, finding the best network architecture and hyperparameter setup can be tedious because of the many design alternatives and hyperparameter values that need to be tuned. This is a general problem that has been faced by many researchers that use neural networks for various tasks. Image recognition and computer vision research communities have reacted by creating prepackaged deep neural architectures that are mostly based on convolutional layers. Those architectures are able to correctly classify objects from thousands of categories when trained with millions of object images. Considering that text datasets are quite smaller, it makes sense to tackle the complexity problem encapsulating network design parameters and hyperparameters in architectures of few layers.

This chapter presents and proposes NgramCNN, a shallow neural architecture for simplifying sentiment analysis model prototyping. Section 7.1 presents similar studies successfully implementing same concepts for analyzing images. There are also studies that propose simple neural networks made up of convolution and

recursive network combinations. Basic architectural components such as word representation layer, as well as convolutional and max-pooling layers are described in details in Section 7.2. Section 7.3 shows text preprocessing steps and experiments conducted to evaluate the performance of NgramCNN models. Finally, Section 7.4 further discusses the obtained results.

7.1 Popular Neural Network Architectures

Deep learning is today the buzzword technology for many tasks and domains like image recognition, machine translation, speech recognition or sentiment analysis. Prepackaged neural network architectures are very easy to use with little domain knowledge, producing top results. Furthermore, performance scales fairly well with increasing data availability and computation speed that have been growing steadily in the recent years. The basic structure of Convolutional Neural Networks (CNN) has been utilized to form dozens of advanced architectures that are producing record-breaking results in the yearly ImageNet challenge [138]. Images of thousands of categories are correctly identified via neural architectures like *VGG-19*, *AlexNet*, *Inception* and more. These advanced image classification models have been included in libraries that are freely available in Python or other programming languages. Furthermore, they are continuously updated with various architectural optimizations introduced in the newer versions. A simple idea would be to utilize the above off-the-shelf complex networks on sentiment analysis tasks as well. However, this is hardly possible, given the high complexity of computer vision neural networks. They do usually have millions of trainable parameters and are thus trained with datasets containing millions of images. Sentiment analysis datasets are usually smaller.

As a result, similar but simpler architectures are proposed in literature studies for text analysis tasks as well. In [139] for example, author proposes a complex architecture called Language Inception Model, inspired by *Inception* of Google. The basic structures of his architecture are convolution layers of different filter lengths that operate in parallel over text phrases to capture n-gram features. To preserve the value of word orders, he also applies a Long Short-Term Memory (LSTM) network in parallel with the Inception-like structure. The generated features from the two parallel structures are merged together and pushed to the fully connected layer that serves as the classifier. The author uses IMDB large movie review dataset

and network structures like unigram, bigram or trigram RNNs to evaluate his architecture. He reports that the simpler bigram RNN performs slightly better than his bigger Inception-like architecture. Also in [132], authors propose a pyramid-shaped, word-level architecture of convolutional layers that increases network depth while decreasing computation time per layer. The latter effect is achieved by keeping a fixed number of feature maps and applying 2-stride downsampling from one layer to the next. The max-pooling layers appear only after certain blocks of consecutive convolutions. They report leading results in five of the large datasets created in [125].

In [140] we find a similar neural network architecture called CharSCNN that is designed to exploit information of characters, words and sentences at the same time. It is created to work particularly well with short and noisy texts such as tweets. Authors claim that character embeddings are able to capture important morphological and shape information of words. For this reason, each character is encoded in a fixed-length character embedding vector and an embedding matrix is created. To generate word-level embeddings, authors use Skip-Gram method with windows size 9. Character-level and word-level embeddings are produced from an English Wikipedia collection of 1.75 billion tokens and joined together. A convolution layer is then applied to produce local features and a max-pooling layer finally creates the feature vectors of the entire sentence. CharSCNN is tested on Stanford Sentiment Treebank corpus of sentences and Stanford Twitter Sentiment corpus of tweets, achieving state-of-the-art results in both.

All above architectures differ in complexity, effectiveness, and number of parameters. However, their fundamental logic is the same: using deep and complex neural network constructs for extracting and selecting features in combination with one or few simple feed-forward layers for classification. In [141], authors support the same idea: generic features extracted from complex CNNs are very powerful and versatile. Their learned representations result high effective even in analogous tasks. Following the same logic, in this chapter, we probe architectures of CNNs that can package the complexity in feature extraction and selection layers and then use a single dense layer as a classifier. The goal is to find simple network configurations that are fast to train and can be easily used as templates, yet providing competing results on sentiment analysis of medium-sized text datasets. We build on the results of the previous chapter, using the optimal convolution filter size and max-pooling region size parameters that were found. Same representation of text words was used as well. Performance of different architectures with hierarchical convolutional and

max-pooling layer combinations are examined. The following sections describe their design, conducted experiments, and the corresponding results.

7.2 Neural Architecture Design Alternatives

7.2.1 NgramCNN Basic Architecture

As shown in Figure 7.1, the basic architecture we designed (NgramCNN) is an extension and generalization of the neural networks explored in the previous section. It starts from the vector representations of words which can be static (sourced from pretrained bundles) or trained on experimenting dataset. Afterwards, it applies W parallel convolutions of growing kernel lengths ($k = 1, \ldots, W$). These operations extract word and phrase combination features out of the text documents. Parameter W (width) can be different, depending on the task and data. Max-pooling layers that follow, select the maximal value from each feature map region of length R, downsampling the feature sets. The reduced features do usually contain the most salient samples. As shown in the previous chapter, the values of R in the different max-pooling layers can be adjusted based on number of words (n) in the training documents. A series of similar parallel convolutional and max-pooling layers go on, forming a structure of L stacks. Same as with R, the value of L can be adapted to specific use cases and the available training data. Finally, output features of the last max-pooling layer are concatenated and pushed to the feed-forward classification layer. More than one classification layer can be used, though.

7.2.2 NgramCNN Pyramid Architecture

A slightly different architecture is the one presented in Figure 7.2. Embedding layer, the first stack of convolutions, last stack of max-pooling layers and the classification layer are exactly the same. The difference is only in the downsampling stacks. Here we use striding convolutions with stride $s > 1$ instead of regional max-pooling with region length R for the intermediate stacks. Max-pooling is still used in the final feature selection stack. If a fixed value of s (e.g., $s = 2$) is used in all downsampling convolution layers, this architecture becomes pyramid-shaped and very similar with the one in [132]. The feature set length is equally reduced in each stack of

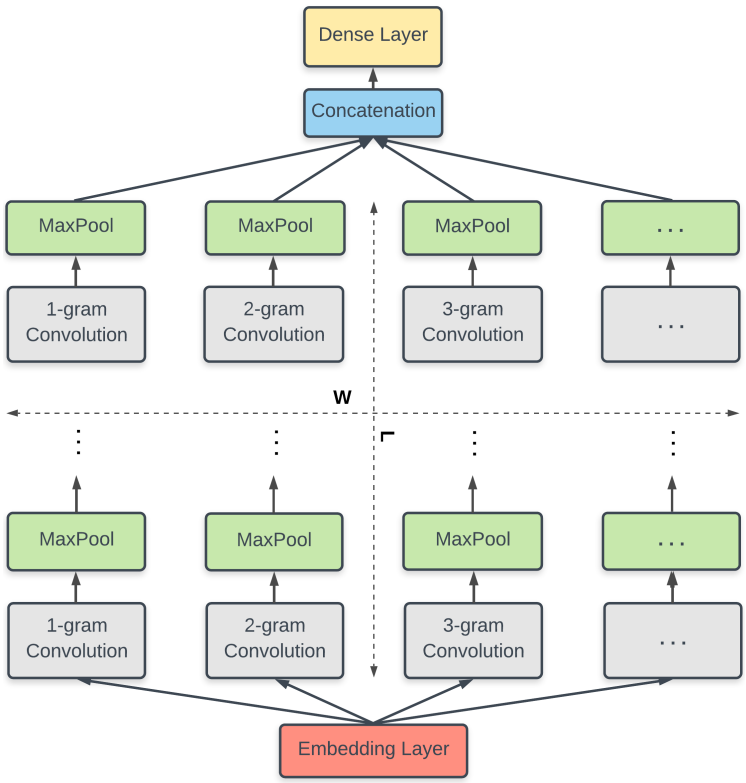

Fig. 7.1 NgramCNN architecture scheme

convolutions till the final max-pooling stack. When compared with the basic version of NgramCNN, there are a couple of differences to note between striding convolution layers and max-pooling layers. One obvious advantage of striding convolutions is their ability to preserve positional information in feature sets. Max-pooling on the other hand, forgets everything about the spatial structure of the data, selecting features that are probably the most valuable for classification. There is still a problem with convolutions regarding training time. They are slower than max-pooling as they have parameters for updating weights. Training algorithms (e.g., *Adam*) need to store information for the striding convolution but not for max-pooling. A study addressing this issue is [142] where authors conduct several object-recognition experiments. They conclude that convolutions with increased stride can completely replace max-pooling layers, simplifying neural network structures without performance loss. Nevertheless, their results apply to image analysis only and do not count for other types of tasks like sentiment analysis, topic recognition etc.

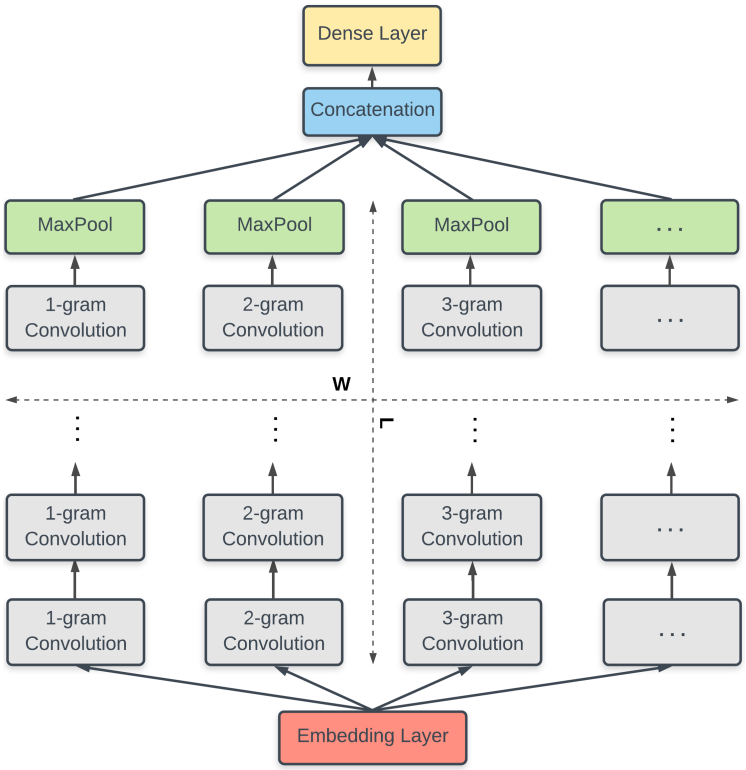

Fig. 7.2 NgramCNN with downsampling convolutions

7.2.3 NgramCNN Fluctuating Architecture

Our third alternative architecture is shown in Figure 7.3. It is significantly different from the first two. The only common parts are embedding (first) and classification (last) layers. Feature selection and extracting layers are organized in a fluctuating form, with features expanding and contracting after each convolutional and max-pooling stack of layers. The first stack of convolutions is same as in the other two architectures. W layers of convolutions with increasing filter sizes are performed in parallel. All generated feature maps are concatenated and pushed to a single max-pooling layer of region size R which significantly reduces (contracts) their size. The second round of convolutions is applied to the new features, expanding their size. The process goes on with another max-pooling layer (stack) for subsampling, repeating feature contraction and expansion in a total of L stacks. Once again, W, L, and R parameters can be chosen to adapt to attributes of available training data.

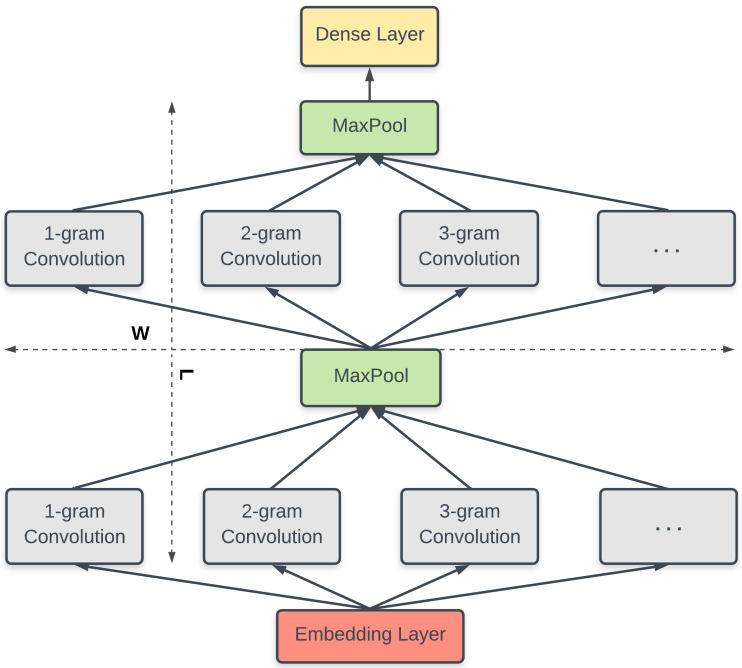

Fig. 7.3 NgramCNN with fluctuating features

7.3 Experimental Settings and Baselines

In this section, we describe various network parameter decisions we made in the conducted experiments where we compare the above architectures. These experiments extend over those presented in [143] (evaluation of basic NgramCNN) to consider the two alternative architectures as well. Other neural network models presented in similar text mining studies are also described and used as comparison baselines.

7.3.1 Common Network Settings

A common setup of basic parameters was chosen for all three network architectures presented in the previous section. Based on the results in that chapter, we fixed W to 3, excluding convolution layers of longer filters from every network stack. Regarding network length (or depth), structures of four-layer stacks were best in most of the previous experiments. As a result, a fixed $L = 4$ was chosen to keep the networks simple and quick to train. Regarding datasets, we reuse four of the five of those described in Section 6.2.1, excluding Mlpn only. This latter dataset has a small size, is overfitted by most of the models and has not been used in similar studies to

compare with. Same data preprocessing steps as those of Section 6.2.2 were applied. Region sizes R of max-pooling layers were also chosen based on experimental results of the previous chapter. Since hierarchical pooling is used, the product of region sizes in the two stacks must equal the optimal value reported in Section 6.4 for each dataset. Hence we chose $R = 2,\ R = 5,\ R = 4$ and $R = 5$ for Sent, Imdb, Phon and Yelp datasets respectively. In the case of NgramCNN pyramid architecture, the above values correspond to convolution stride (s) of the second layer as well. Regarding the representation of text documents, pretrained word embeddings sourced from GoogleNews bundle were used once again, same as in Section 6.3.1.

7.3.2 Baseline Models

Using the settings and parameters described above, we contrast the three NgramCNN architectures with each other as well as against similar neural network models proposed in relevant studies. One of the comparison models (SingleLSTM) is made up of a single LSTM layer positioned above the embedding layer. It is followed by the feed-forward classification layer. Another baseline model was presented in [122] (BLSTM-POOL). Here they use a bidirectional recurrent network (left and right LSTM) to capture word contexts. A max-pooling layer that follows is used for automatic selection of top features for classification. Authors report very good results on topic identification task and lower but still good results on sentiment analysis of movie reviews. A more complex baseline model was introduced in [123] (BLSTM-2DCNN). It builds upon the bidirectional structure of [122], adding two-dimensional convolution and pooling layers applied on word-feature windows. The model is exercised on various datasets and tasks and achieves top scores in sentiment analysis of short sentences.

We also compared against a simple model made up of few convolution layers in a single stack (SingleCNN). This model is quite similar to that of Kim in [124]. Here, convolution layers of filter size 3, 4, and 5 operate on word embeddings and the feature maps they produce are concatenated together. Max-pooling and a dropout layer for regularization follow the convolutions and a dense layer is used for classification. Considering the diversity of the datasets and tasks we experiment on, it is worth comparing against simpler and more traditional classification models that work with bag-of-words text representation and check their performance. As a result, we also implemented Support Vector Machine and Logistic Regression

classifiers using tf-idf feature vectorizer. They were optimized with grid-searched regularization parameters.

7.4 Results and Discussion

This section presents and discusses classification performance scores obtained from the datasets. The three variants of NgramCNN are compared with each other and the rest of baseline models. The sensitivity of results with respect to various network and training parameters is also discussed.

7.4.1 Sentiment Polarity Classification Results

Each experiment was run on 70/10/20 percent splits for training, development, and testing respectively. Table 7.1 summarizes classification accuracy scores of the models on each of the four datasets. On sentiment polarity analysis (smallest dataset with shortest documents), NgramCNN models perform relatively badly. Optimized Logistic Regression and Support Vector Machine, together with SingleCNN network achieve very good results whereas LSTM-based models (SingleLSTM and BLSTM-POOL) perform worse. Best score of 82.32% was reached from BLSTM-2DCNN presented in [123] which integrates both recurrent and convolution networks. This result is slightly lower from the very top score of 83.1% reported in [136]. On movie reviews dataset, we see a different picture. The first two versions of NgramCNN score above 91%. Top accuracy score on this dataset is 92.23%, as reported in [137]. Fluctuating architecture is well behind, together with the two linear models and SingleCNN. LSTM-based models, on the other hand, achieve very poor results that are below 86%. Obviously, recurrent neural networks do not work well on text analysis tasks of long documents.

On smartphone reviews, we see similar results, with NgramCNN basic architecture peaking at 95.92%. Unfortunately, we did not find a literature score report on this dataset to compare with. The Pyramid architecture is very close whereas the Fluctuating version is again well behind. We also see that LSTM models perform relatively well (BLSTM-2DCNN), at least on short documents. Linear models, on the other hand, perform weakly. Yelp business reviews is the last dataset we experimented on. Once again, best results are achieved from NgramCNN Basic

which is again followed by the Pyramid version. Top literature score on this dataset is 97.36% (as reported in [132]) which is 2.5% higher than that of NgramCNN Basic. SingleCNN is also a good player, with an accuracy of 93.86%. The rest of the models, including Fluctuating architecture, are behind of more than 2%.

Table 7.1 Accuracy scores of NgramCNN variants and baselines

Network	Sent	Imdb	Phon	Yelp
NgCNN Basic	79.87	91.14	**95.92**	**94.95**
NgCNN Pyramid	79.52	**91.21**	95.7	94.83
NgCNN Fluctuate	77.41	89.32	93.45	92.27
Optimized LR	81.63	89.48	92.46	91.75
Optimized SVM	82.06	88.53	92.67	92.36
SingleLSTM	80.33	84.93	93.71	90.22
SingleCNN	81.79	89.84	94.25	93.86
BLSTM-POOL	80.96	85.54	94.33	91.19
BLSTM-2DCNN	**82.32**	85.70	95.52	91.48

7.4.2 Further Observations

Based on the results of Table 7.1, we see that basic version of NgramCNN is the best of the three design alternatives. Pyramid version that uses striding convolutions in the intermediate stacks, is quite similar in design and slightly worse in performance. To further assess the role of striding convolutions and max-pooling layers, we explored another version which uses no max-pooling layers at all. This version performed even worse with accuracy scores lower than 2%. Apparently, regional max-pooling does a good job as a feature selector and is also faster to train. Fluctuating version, on the other hand, scored badly on every dataset. It is obviously not a good network design choice. The most intuitive explanation for this is the way it mixes up the features of the different convolution filters. Treating them separately as in the first two architectures yields better classification results. It also comes out that models based on recurrent neural networks perform well on short documents but considerably worse on longer ones. This has to do with their internal design. Their memorization abilities are limited and they cannot preserve long-term word dependencies. Furthermore, these kinds of networks usually take longer to train. Due

to their simpler design, models based on CNNs were much faster, with training times comparable to those of the two linear models. These later performed comparably well on the smaller datasets in which they might be acceptable choices in cases when performance is not crucial.

Regarding network hyperparameters, the most important design decision such as depth and width of network or size of max-pooling regions were chosen based on results of the previous chapter, as explained in Section 7.3.1. It is important to note that NgramCNN architecture is highly flexible. Different design choices such as more convolutional and max-pooling stacks or longer filters can be explored for even higher performance if other datasets are available. Remaining parameters were chosen based on grid search results. The number of epochs until convergence was irregular and specific on each dataset. Sentences converged in three epochs, whereas smartphone and movie reviews required seven and four epochs respectively. Yelp dataset that was the biggest required nine epochs to converge. There was not much sensitivity with respect to batch size. A batch of 60 was usually optimal in most experiments. To conclude, *Softplus* and *Sigmoid* were equally good as activation functions for the output layer.

7.4.3 Concluding Remarks

In this chapter, we presented a couple of neural network architectures for sentiment analysis that are based on convolutional layers applied on top of pretrained word embedding representations of texts. They follow the same fundamental design of the top performing architectures on image recognition tasks: complexity in feature extraction and selection combined with simplicity in classification. Based on obtained results, this design appears successful in text analysis as well. The basic design of convolution and max-pooling stacks of layers resulted the best performing one. Even though it is not a record-breaking architecture in terms of classification accuracy, it is yet very fast and highly adaptable to various kinds of datasets. Many versions that differ in width, depth and pooling region sizes can be easily constructed and potentially produce state-of-the-art results if properly tuned. We thus believe that it can be very useful as a prototyping or baseline sentiment analysis model, especially for practitioners. One limitation of our experiments is the use of static word vectors for text representations. If bigger text datasets are available, tuning word vectors on training texts would require more computation time but probably

produce slightly better features. Another improvement of NgramCNN could be to employ sentiment analysis on different splits (phrases) of text documents and predict their emotional polarity independently. The overall polarity of the document could be assessed utilizing aggregation schemes like Dempster-Shafer Inference or Abductive Reasoning [144].

Chapter 8

Conclusions and Prospects

This thesis addressed different aspects regarding sentiment analysis of songs and various types of texts. The main concerns were the need for viable methods to obtain bigger labeled datasets, the adequacy of distributed word representations for sentiment analysis of texts and the complexity reduction of neural network models for easier and faster prototyping. Three main goals were set out: *(i)* exploring the viability of crowdsourcing methods like crawled social tags for constructing datasets of song emotions, *(ii)* probing the role of factors like training corpus size or generation method in the quality of word embeddings that are produced, *(iii)* designing a neural network architecture made up of convolution and max-pooling layers for encapsulating complexity and simplifying sentiment analysis model creation. In short, the main contributions of this thesis are:

- The validity proof of crowdsourcing methods as alternatives for creating labeled data collection by comparing emotional spaces of social tags with those of psychologists, presented in Chapter 3.
- The creation of two datasets of song emotions using *Last.fm* social tags and the exploration of lexicons as possible means for text data programming, also introduced in Chapter 3.
- A systematic literature survey about hybrid and context-based recommender systems, the results of which are summarized in Chapter 4.
- The design of an in-car music mood recommender system in collaboration with Telecom Italia JOL Mobilab, described in Chapter 4.

- The empirical identification of interesting relations between text size, training method or text topic and the generated word vectors (Chapter 5).
- A set of patterns relating neural network hyperparameters with the size of datasets and length of documents, with respect to optimization of text sentiment prediction revealed in Chapter 6.
- A neural network architecture made up of convolutional and max-pooling layers that encapsulates most of hyperparameter tuning complexity, simplifying the creation of models for sentiment analysis of texts (Chapter 7).

Literature observations, as well as conducted experiments, show that social tags and other forms of crowdsourcing are viable for collecting labels of data if correctly applied. Furthermore, the excellent compliance between user tag emotion spaces and those of psychological emotion models, makes us conclude that massive crowd tags are a trustworthy source of intelligence. In this context, the creation of MoodyLyrics4Q and MoodyLyricsPN was based on *Last.fm* affect tags. A variant of Russell's model with tags of four categories was used for representing emotion classes. MoodyLyrics4Q is the first music dataset that is both large and polarized, follows a popular emotion model, and is released for public use.

The systematic survey on hybrid and context-based recommender systems revealed current tendencies such as the growing interest in the field, the popularity of K-nearest neighbors and K-means algorithms, the preference of weighted filtering technique combinations over the more complex ones, frequent applications on movies, and the use of accuracy for evaluating the quality of recommendations. Regarding context-based recommenders, those implementing input data pre-filtering are the most simple and common. The recommender of songs described in Section 4.3 was designed to work in the context of car driving, using the intelligence of collected user tags about songs. The goal was to enhance comfort and safety of driving experience by adjusting music emotions to contextual factors like driving patterns and emotional state of the driver, time of day, etc.

The experimental work with word embeddings revealed relations between training data and methods with the quality of the generated word vectors. Among the most important insights, we can mention the superiority of Glove over Skip-Gram on analysis of song lyrics and the direct relation between corpus size and vocabulary richness with the performance of word vectors. Another important conclusion drawn

from those experiments is the fact that for classification based on small training datasets, it is better to source pretrained word vectors created from big text corpora.

Numerous experiments with convolutional and max-pooling neural networks, trained with datasets of different sizes and variable-length text documents, revealed certain patterns relating those data properties with network hyperparameters, the number of stacked layers or pooling size and the optimal classification performance. Among the most important tendencies that were observed, those relating document length with the size of generated feature maps $(7-15)$ and the number of layers with dataset size were particularly valuable. The experiments that were conducted covered text mining tasks such as sentiment polarity prediction of song lyrics, movie reviews, and product reviews.

The results of those experiments were used to address neural network complexity problem by designing an architecture of convolutional and regional max-pooling layers. Its role is to encapsulate feature extraction and selection complexity by having most of the hyperparameters preset and adjusting the few remaining ones in accordance with training data properties. Using NgramCNN architecture as a starting point, one can easily create models that differ in width, depth and pooling region sizes and can possibly generate competitive results. This way, model creation on tasks such as sentiment analysis or text mining in general, becomes easier for practitioners. The architecture can be applied to various dataset sizes and document lengths with little parameter tunning and provides state-of-the-art performance in text polarity prediction.

The democratization of deep learning and the trend towards data-driven intelligent solutions requires even more and bigger public datasets. The most popular benchmarking datasets that are currently used for sentiment analysis such as IMDB Movie Reviews[1] or Cornell Sentence Polarity[2] are limited in text polarity (*negative* vs. *positive*) recognition. Practical systems do often require datasets with hundred thousands of documents containing *neutral* text category as well. It is thus important to crawl opinion texts from websites and build bigger and better experimentation datasets. The high cost of labeling large amounts of documents can be significantly reduced utilizing crowdsourcing or data programming. The latter was not very successful with song lyrics (Section 3.4) when employing affect terms of lexicons

[1] http://ai.stanford.edu/~amaas/data/sentiment
[2] http://www.cs.cornell.edu/people/pabo/movie-review-data

for emotion category discrimination. Nevertheless, more advanced heuristics could solve this problem, especially for shorter documents such as product reviews.

With enough data in possession, other promising ideas involving deep neural networks can be explored. For example, automatic features of convolutional or other types of neural networks are not always optimal. Ensemble methods of different levels might produce improved features and results when applied in the context of deep neural networks. As a result, one possibility could be to aggregate multiple types of local and distributed word features (feature-level fusing). Different feature types do usually represent complementary data characteristics. Each of them reflects a different view of the original data. Sentiment analysis systems are thus expected to benefit from feature aggregations.

Another problem is the information loss that happens in the consecutive convolution and pooling operations. Recurrent neural networks are immune to this problem but do not work well with longer texts due to their inability to memorize long word sequences. A possibility could be to use recurrent-convolutional networks (proved to work optimally in [145], [123] and more) on short phrases of the longer document and then perform decision-level fusion for the overall prediction. Various aggregation schemes like Dempster-Shafer Inference or Abductive Reasoning can be explored in the decision step [144].

References

[1] Yves Bestgen. Building affective lexicons from specific corpora for automatic sentiment analysis. In *Proceedings of the International Conference on Language Resources and Evaluation, LREC 2008, 26 May - 1 June 2008, Marrakech, Morocco*, 2008.

[2] Nobuhiro Kaji and Masaru Kitsuregawa. Building lexicon for sentiment analysis from massive collection of HTML documents. In *Proceedings of the Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 1075–1083, 2007.

[3] Yu-Ching Lin, Yi-Hsuan Yang, and Homer H. Chen. Exploiting online music tags for music emotion classification. *ACM Trans. Multimedia Comput. Commun. Appl.*, 7S(1):26:1–26:16, November 2011.

[4] Xiao Hu and K. Stephen Downie. Exploring mood metatada: Relationship with genre, artist and usage metadata. In *Proceedings of the 8th International Conference on Music Information Retrieval*, September 2007.

[5] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. A neural probabilistic language model. *J. Mach. Learn. Res.*, 3:1137–1155, March 2003.

[6] Cyril Laurier, Mohamed Sordo, Joan Serrà, and Perfecto Herrera. Music mood representations from social tags. In Keiji Hirata, George Tzanetakis, and Kazuyoshi Yoshii, editors, *ISMIR*, pages 381–386. International Society for Music Information Retrieval, 2009.

[7] J.A. Russell. A circumplex model of affect. *Journal of personality and social psychology*, 39(6):1161–1178, 1980.

[8] Walaa Medhat, Ahmed Hassan, and Hoda Korashy. Sentiment analysis algorithms and applications: A survey. *Ain Shams Engineering Journal*, 5(4):1093–1113, 2014.

[9] Bo Pang, Lillian Lee, et al. Opinion mining and sentiment analysis. *Foundations and Trends® in Information Retrieval*, 2(1–2):1–135, 2008.

[10] Bing Liu. Sentiment analysis and opinion mining. *Synthesis lectures on human language technologies*, 5(1):1–167, 2012.

[11] Bing Liu and Lei Zhang. A survey of opinion mining and sentiment analysis. In *Mining text data*, pages 415–463. Springer, 2012.

[12] Kumar Ravi and Vadlamani Ravi. A survey on opinion mining and sentiment analysis: tasks, approaches and applications. *Knowledge-Based Systems*, 89:14–46, 2015.

[13] Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. Thumbs up?: Sentiment classification using machine learning techniques. In *Proceedings of the ACL-02 Conference on Empirical Methods in Natural Language Processing - Volume 10*, EMNLP '02, pages 79–86, Stroudsburg, PA, USA, 2002. Association for Computational Linguistics.

[14] Youngjoong Ko and Jungyun Seo. Automatic text categorization by unsupervised learning. In *Proceedings of the 18th Conference on Computational Linguistics - Volume 1*, COLING '00, pages 453–459, Stroudsburg, PA, USA, 2000. Association for Computational Linguistics.

[15] Peter D Turney. Thumbs up or thumbs down?: semantic orientation applied to unsupervised classification of reviews. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 417–424. Association for Computational Linguistics, 2002.

[16] Warren Mcculloch and Walter Pitts. A logical calculus of ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics*, 5:127–147, 1943.

[17] A. M. Turing. Computing machinery and intelligence. *Mind*, 59(236):433–460, 1950.

[18] Frank Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386, 1958.

[19] Marvin Minsky and Seymour Papert. *Perceptrons: An Introduction to Computational Geometry*. MIT Press, Cambridge, MA, USA, 1969.

[20] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning internal representations by error propagation. Technical report, California Univ San Diego La Jolla Inst for Cognitive Science, 1985.

[21] Yoav Freund and Robert E Schapire. Experiments with a new boosting algorithm. In Icml (Vol. 96, pp. 148-156), 1996.

[22] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

[23] Bernhard Scholkopf and Alexander J. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press, Cambridge, MA, USA, 2001.

[24] Thorsten Joachims. Text categorization with support vector machines: Learning with many relevant features. In *European conference on machine learning*, pages 137–142. Springer, 1998.

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

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

[27] Tomas Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States.*, pages 3111–3119, 2013.

[28] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. Glove: Global vectors for word representation. In Alessandro Moschitti, Bo Pang, and Walter Daelemans, editors, *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543. ACL, 2014.

[29] Lorenzo Feruglio, Sabrina Corpino, and Daniele Calvi. Neural networks for event detection: an interplanetary cubesat asteroid mission case study. In *AIAA SPACE & ASTRONAUTICS FORUM 2016*, pages 1–5, Washington D.C., 2016. American Institute of Aeronautics and Astronautics.

[30] Alexander J Ratner, Christopher M De Sa, Sen Wu, Daniel Selsam, and Christopher Ré. Data programming: Creating large training sets, quickly. In *Advances in Neural Information Processing Systems*, pages 3567–3575, 2016.

[31] Thomas Schäfer, Peter Sedlmeier, Christine Städtler, and David Huron. The psychological functions of music listening. *Frontiers in psychology*, 4(511):1–33, 2013.

[32] Panteleimon Ekkekakis. *Measurement in Sport and Exercise Psychology*, chapter Affect, Mood, and Emotion. Human Kinetics, 2012.

[33] Kate Hevner. Experimental studies of the elements of expression in music. *The American Journal of Psychology*, 48:246–268, 1936.

[34] Xiao Hu and J Stephen Downie. Exploring mood metadata: Relationships with genre, artist and usage metadata. In *ISMIR*, pages 67–72, 2007.

[35] Cyril Laurier and Perfecto Herrera. Audio music mood classification using support vector machine. pages 2–4, ISMIR 2007.

[36] Pasi Saari and Tuomas Eerola. Semantic computing of moods based on tags in social media of music. *IEEE Transactions on Knowledge and Data Engineering*, 26(10):2548–2560, 2014.

[37] Xiao Hu, Mert Bay, and J Stephen Downie. Creating a simplified music mood classification ground-truth set. pages 309–310, ISMIR 2007.

[38] Margaret M Bradley and Peter J Lang. Affective norms for english words (anew): Instruction manual and affective ratings. Technical report. The center for research in psychophysiology, University of Florida, 1999.

[39] Seungwon Oh, Minsoo Hahn, and Jinsul Kim. Music mood classification using intro and refrain parts of lyrics. In *Information Science and Applications (ICISA), 2013 International Conference on*, pages 1–3. IEEE, 2013.

[40] Yajie Hu, Xiaoou Chen, and Deshun Yang. Lyric-based song emotion detection with affective lexicon and fuzzy clustering method. In *Proceedings of ISMIR 2009*, pages 123–128, 2009.

[41] Menno Van Zaanen and Pieter Kanters. Automatic mood classification using tf* idf based on lyrics. pages 75–80, ISMIR 2010.

[42] Byeong jun Han, Seungmin Rho, Roger B. Dannenberg, and Eenjun Hwang. Smers: Music emotion recognition using support vector regression. In Keiji Hirata, George Tzanetakis, and Kazuyoshi Yoshii, editors, *ISMIR*, pages 651–656. International Society for Music Information Retrieval, 2009.

[43] Xiao Hu. *Improving music mood classification using lyrics, audio and social tags*. PhD thesis, Citeseer, 2010.

[44] Yi-Hsuan Yang, Yu-Ching Lin, Heng-Tze Cheng, I-Bin Liao, Yeh-Chin Ho, and Homer H. Chen. *Advances in Multimedia Information Processing - PCM 2008: 9th Pacific Rim Conference on Multimedia, Tainan, Taiwan, December 9-13, 2008. Proceedings*, chapter Toward Multi-modal Music Emotion Classification, pages 70–79. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008.

[45] Jeff Howe. The rise of crowdsourcing. Wired magazine, 14(6), pages 1–4, 2006.

[46] James Surowiecki. *The Wisdom of Crowds: Why the Many Are Smarter Than the Few and How Collective Wisdom Shapes Business, Economies, Societies and Nations*. Doubleday, May 2004.

[47] Michael I. Mandel, Douglas Eck, and Yoshua Bengio. Learning tags that vary within a song. In *Proceedings of the International Society for Music Information Retrieval conference*, pages 399–404, August 2010.

[48] Jacquelin A Speck, Erik M Schmidt, Brandon G Morton, and Youngmoo E Kim. A comparative study of collaborative vs. traditional musical mood annotation. pages 549–554, ISMIR 2011.

[49] Jin Ha Lee and Xiao Hu. Generating ground truth for music mood classification using mechanical turk. In *Proceedings of the 12th ACM/IEEE-CS Joint Conference on Digital Libraries*, JCDL '12, pages 129–138, New York, NY, USA, 2012. ACM.

[50] Michael I Mandel and Daniel PW Ellis. A web-based game for collecting music metadata. *Journal of New Music Research*, 37(2):151–165, 2008.

[51] E. L. M. Law, L. Von Ahn, R. B. Dannenberg, and M. Crawford. Tagatune: A Game for Music and Sound Annotation. In *International Conference on Music Information Retrieval (ISMIR'07)*, pages 361–364, 2007.

[52] Morgan Ames and Mor Naaman. Why we tag: motivations for annotation in mobile and online media. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, CHI '07, pages 971–980, New York, NY, USA, 2007. ACM.

[53] Paul Lamere. Social tagging and music information retrieval. *Journal of New Music Research*, 37(2):101–114, 2008.

[54] R. Malheiro, R. Panda, P. Gomes, and R. P. Paiva. Emotionally-relevant features for classification and regression of music lyrics. *IEEE Transactions on Affective Computing*, PP(99):1–1, 2016.

[55] Masataka Goto, Kazuyoshi Yoshii, Hiromasa Fujihara, Matthias Mauch, and Tomoyasu Nakano. Songle: A web service for active music listening improved by user contributions. In *Proceedings of the 12th International Society for Music Information Retrieval Conference, ISMIR 2011, Miami, Florida, USA, October 24-28, 2011*, pages 311–316, 2011.

[56] Erion Çano and Maurizio Morisio. Crowdsourcing emotions in music domain. *International Journal of Artificial Intelligence & Applications*, 8(4):25–40, 2017.

[57] Xiao Hu, J Stephen Downie, and Andreas F Ehmann. Lyric text mining in music mood classification. *American music*, 183(5,049):2–209, 2009.

[58] Mohammad Soleymani, Micheal N Caro, Erik M Schmidt, Cheng-Ya Sha, and Yi-Hsuan Yang. 1000 songs for emotional analysis of music. In *Proceedings of the 2nd ACM international workshop on Crowdsourcing for multimedia*, pages 1–6. ACM, 2013.

[59] D. Turnbull, L. Barrington, D. Torres, and G. Lanckriet. Semantic annotation and retrieval of music and sound effects. *IEEE Transactions on Audio, Speech, and Language Processing*, 16(2):467–476, Feb 2008.

[60] Rada Mihalcea and Carlo Strapparava. Lyrics, music, and emotions. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, EMNLP-CoNLL '12, pages 590–599, Stroudsburg, PA, USA, 2012. Association for Computational Linguistics.

[61] Kerstin Bischoff, Claudiu S. Firan, Raluca Paiu, Wolfgang Nejdl, Cyril Laurier, and Mohamed Sordo. Music mood and theme classification - a hybrid approach. In Keiji Hirata, George Tzanetakis, and Kazuyoshi Yoshii, editors, *ISMIR*, pages 657–662. International Society for Music Information Retrieval, 2009.

[62] Thierry Bertin-mahieux, Daniel P. W. Ellis, Brian Whitman, and Paul Lamere. The million song dataset. In *Proceedings of the 12th International Conference on Music Information Retrieval (ISMIR)*, 2011.

[63] Shuo Chen, Josh L. Moore, Douglas Turnbull, and Thorsten Joachims. Playlist prediction via metric embedding. In *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '12, pages 714–722, New York, NY, USA, 2012. ACM.

[64] Erion Çano and Maurizio Morisio. Music mood dataset creation based on last.fm tags. In *Computer Science & Information Technology (CS & IT)*, volume 7, pages 15–26. AIRCC Publishing Corporation, Vienna, Austria, May 2017.

[65] Peter Sheridan Dodds and Christopher M. Danforth. Measuring the happiness of large-scale written expression: Songs, blogs, and presidents. *Journal of Happiness Studies*, 11(4):441–456, Aug 2010.

[66] George A. Miller. Wordnet: A lexical database for english. *Commun. ACM*, 38(11):39–41, November 1995.

[67] Carlo Strapparava and Alessandro Valitutti. WordNet-Affect: An affective extension of WordNet. In *Proceedings of the 4th International Conference on Language Resources and Evaluation*, pages 1083–1086. ELRA, 2004.

[68] Xiao Hu and J. Stephen Downie. Improving mood classification in music digital libraries by combining lyrics and audio. In *Proceedings of the 10th Annual Joint Conference on Digital Libraries*, JCDL '10, pages 159–168, New York, NY, USA, 2010. ACM.

[69] Erion Çano and Maurizio Morisio. Moodylyrics: A sentiment annotated lyrics dataset. In *2017 International Conference on Intelligent Systems, Metaheuristics and Swarm Intelligence*, ACM, pages 381–386, Hong Kong, March 2017.

[70] Francesco Ricci, Lior Rokach, and Bracha Shapira. *Recommender Systems Handbook*, chapter Introduction to Recommender Systems Handbook, pages 1–35. Springer US, Boston, MA, 2011.

[71] David Goldberg, David Nichols, Brian M. Oki, and Douglas Terry. Using collaborative filtering to weave an information tapestry. *Commun. ACM*, 35(12):61–70, December 1992.

[72] Paul Resnick, Neophytos Iacovou, Mitesh Suchak, Peter Bergstrom, and John Riedl. Grouplens: An open architecture for collaborative filtering of netnews. In *Proceedings of the 1994 ACM Conference on Computer Supported Cooperative Work*, CSCW '94, pages 175–186, New York, NY, USA, 1994. ACM.

[73] Will Hill, Larry Stead, Mark Rosenstein, and George Furnas. Recommending and evaluating choices in a virtual community of use. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '95, pages 194–201, New York, NY, USA, 1995. ACM Press/Addison-Wesley Publishing Co.

[74] Ken Lang. NewsWeeder: learning to filter netnews. In *Proceedings of the 12th International Conference on Machine Learning*, pages 331–339. Morgan Kaufmann publishers Inc.: San Mateo, CA, USA, 1995.

[75] Bruce Krulwich and Chad Burkey. Learning user information interests through extraction of semantically significant phrases. In *Proceedings of the AAAI spring symposium on machine learning in information access*, pages 100–112. AAAI Press Menlo Park, 1996.

[76] Michael D. Ekstrand, John T. Riedl, and Joseph A. Konstan. Collaborative filtering recommender systems. *Found. Trends Hum.-Comput. Interact.*, 4(2):81–173, February 2011.

[77] Robin Burke. Knowledge-based recommender systems. In *ENCYCLOPEDIA OF LIBRARY AND INFORMATION SYSTEMS*, page 2000. Marcel Dekker, 2000.

[78] Erion Çano and Maurizio Morisio. Characterization of public datasets for recommender systems. In *Research and Technologies for Society and Industry Leveraging a better tomorrow (RTSI), 2015 IEEE 1st International Forum on*, pages 249–257, Torino, Italy, Sept 2015.

[79] Erion Çano. Cloud-based recommendation systems: Applications and solutions. 2015, https://iris.polito.it/handle/11583/2669861.

[80] Pearl Pu, Li Chen, and Rong Hu. Evaluating recommender systems from the user's perspective: survey of the state of the art. *User Modeling and User-Adapted Interaction*, 22(4):317–355, 2012.

[81] Marko Balabanović and Yoav Shoham. Fab: Content-based, collaborative recommendation. *Commun. ACM*, 40(3):66–72, March 1997.

[82] Badrul M. Sarwar, Joseph A. Konstan, Al Borchers, Jon Herlocker, Brad Miller, and John Riedl. Using filtering agents to improve prediction quality in the grouplens research collaborative filtering system. In *Proceedings of the 1998 ACM Conference on Computer Supported Cooperative Work*, CSCW '98, pages 345–354, New York, NY, USA, 1998. ACM.

[83] Barbara Kitchenham and Stuart Charters. Guidelines for performing Systematic Literature Reviews in Software Engineering. Technical Report EBSE 2007-001, Keele University and Durham University Joint Report, 2007.

[84] Robin Burke. Hybrid recommender systems: Survey and experiments. *User Modeling and User-Adapted Interaction*, 12(4):331–370, November 2002.

[85] D.S. Cruzes and Tore Dyba. Recommended steps for thematic synthesis in software engineering. In *Empirical Software Engineering and Measurement (ESEM), 2011 International Symposium on*, pages 275–284, Sept 2011.

[86] Erion Çano and Maurizio Morisio. Hybrid recommender systems: A systematic literature review. *INTELLIGENT DATA ANALYSIS*, 21(6):1487–1524, 2017.

[87] Deuk Hee Park, Hyea Kyeong Kim, Il Young Choi, and Jae Kyeong Kim. A literature review and classification of recommender systems research. *Expert Syst. Appl.*, 39(11):10059–10072, September 2012.

[88] J. Bobadilla, F. Ortega, A. Hernando, and A. GutiéRrez. Recommender systems survey. *Know.-Based Syst.*, 46:109–132, July 2013.

[89] Jonathan L. Herlocker, Joseph A. Konstan, Loren G. Terveen, and John T. Riedl. Evaluating collaborative filtering recommender systems. *ACM Trans. Inf. Syst.*, 22(1):5–53, January 2004.

[90] Ana Belén Barragáns-Martínez, Enrique Costa-Montenegro, Juan C Burguillo, Marta Rey-López, Fernando A Mikic-Fonte, and Ana Peleteiro. A hybrid content-based and item-based collaborative filtering approach to recommend tv programs enhanced with singular value decomposition. *Information Sciences*, 180(22):4290–4311, 2010.

[91] Anind K. Dey. Understanding and using context. *Personal Ubiquitous Comput.*, 5(1):4–7, January 2001.

[92] Gediminas Adomavicius and Alexander Tuzhilin. *Context-Aware Recommender Systems*, pages 217–253. Springer US, Boston, MA, 2011.

[93] N. Dibben and V.J. Williamson. An exploratory survey of in-vehicle music listening. *Psychology of Music*, 35(4):571, 2007.

[94] Sundé M. Nesbit, Judith C. Conger, and Anthony J. Conger. A quantitative review of the relationship between anger and aggressive driving. *Aggression and Violent Behavior*, 12(2):156 – 176, 2007.

[95] Gaetano Valenza, Luca Citi, Antonio Lanatá, Enzo Pasquale Scilingo, and Riccardo Barbieri. Revealing real-time emotional responses: a personalized assessment based on heartbeat dynamics. *Scientific reports*, 4, 2014.

[96] Ryan D. Garrity and Jack Demick. Relations among personality traits, mood states, and driving behaviors. *Journal of Adult Development*, 8(2):109–118, April 2001.

[97] Marjolein D. van der Zwaag, Joris H. Janssen, Clifford Nass, Joyce H.D.M. Westerink, Shrestha Chowdhury, and Dick de Waard. Using music to change mood while driving. *Ergonomics*, 56(10):1504–1514, 2013. PMID: 23998711.

[98] Daniel Västfjäll. Emotion induction through music: A review of the musical mood induction procedure. *Musicae Scientiae*, 5(1 suppl):173–211, 2002.

[99] Christian Peter, Eric Ebert, and Helmut Beikirch. *Affective Computing and Intelligent Interaction: First International Conference, ACII 2005, Beijing, China, October 22-24, 2005. Proceedings*, chapter A Wearable Multi-sensor System for Mobile Acquisition of Emotion-Related Physiological Data, pages 691–698. Springer Berlin Heidelberg, Berlin, Heidelberg, 2005.

[100] Gediminas Adomavicius, Ramesh Sankaranarayanan, Shahana Sen, and Alexander Tuzhilin. Incorporating contextual information in recommender systems using a multidimensional approach. *ACM Trans. Inf. Syst.*, 23(1):103–145, January 2005.

[101] Richard Viereckl, Dietmar Ahlemann, Alex Koster, and Sebastian Jursch. Racing ahead with autonomous cars and digital innovation. *Auto Tech Review*, 4(12):18–23, dec 2015.

[102] Y. L. Murphey, R. Milton, and L. Kiliaris. Driver's style classification using jerk analysis. In *Computational Intelligence in Vehicles and Vehicular Systems, 2009. CIVVS '09. IEEE Workshop on*, pages 23–28, March 2009.

[103] Erion Çano, Riccardo Coppola, Maurizio Morisio, Eleonora Gargiulo, and Marco Marengo. Mood-based on-car music recommendations. In *Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering*, volume 188, pages 154–163. Springer, Leicester, UK, Nov 2016.

[104] Zellig S Harris. Distributional structure. *Word*, 10(2-3):146–162, 1954.

[105] Quoc Le and Tomas Mikolov. Distributed representations of sentences and documents. In *Proceedings of the 31st International Conference on International Conference on Machine Learning - Volume 32*, ICML'14, pages II–1188–II–1196. JMLR.org, 2014.

[106] Erion Çano and Maurizio Morisio. Quality of word embeddings on sentiment analysis tasks. Springer, 22nd International Conference on Applications of Natural Language to Information Systems, NLDB 2017, pages 332–338, Liege, Belgium, June 21-23, 2017.

[107] Omer Levy and Yoav Goldberg. Dependency-based word embeddings. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, ACL 2014, June 22-27, 2014, Baltimore, MD, USA, Volume 2: Short Papers*, pages 302–308, 2014.

[108] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. Distributed representations of words and phrases and their compositionality. *CoRR*, abs/1310.4546, 2013.

[109] Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. Thumbs up? sentiment classification using machine learning techniques. In *Proceedings of EMNLP*, pages 79–86, 2002.

[110] Bo Pang and Lillian Lee. A sentimental education: Sentiment analysis using subjectivity. In *Proceedings of ACL*, pages 271–278, 2004.

[111] Houshmand Shirani-Mehr. Applications of deep learning to sentiment analysis of movie reviews. Technical report, Stanford University, 2014.

[112] Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 142–150, Portland, Oregon, USA, June 2011. Association for Computational Linguistics.

[113] Hadi Pouransari and Saman Ghili. Deep learning for sentiment analysis of movie reviews. Technical report, Stanford University, 2014.

[114] Rie Johnson and Tong Zhang. Effective use of word order for text categorization with convolutional neural networks. In *NAACL HLT 2015, The 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Denver, Colorado, USA, May 31 - June 5, 2015*, pages 103–112, 2015.

[115] Xiao Hu, J. Stephen Downie, and Andreas F. Ehmann. Lyric text mining in music mood classification. In *Proceedings of the 10th International Society for Music Information Retrieval Conference, ISMIR 2009, Kobe International Conference Center, Kobe, Japan, October 26-30, 2009*, pages 411–416, 2009.

[116] Menno van Zaanen and Pieter Kanters. Automatic mood classification using tf*idf based on lyrics. In *Proceedings of ISMIR 2010*, Utrecht, Netherlands, 2010.

[117] Hui He, Jianming Jin, Yuhong Xiong, Bo Chen, Wu Sun, and Ling Zhao. *Language Feature Mining for Music Emotion Classification via Supervised Learning from Lyrics*, pages 426–435. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008.

[118] Won-Sook Lee and Dan Yang. Music emotion identification from lyrics. *2013 IEEE International Symposium on Multimedia*, 00:624–629, 2009.

[119] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[120] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Comput.*, 9(8):1735–1780, November 1997.

[121] Meishan Zhang, Yue Zhang, and Duy-Tin Vo. Gated neural networks for targeted sentiment analysis. AAAI, pp. 3087-3093, 2016.

[122] Siwei Lai, Liheng Xu, Kang Liu, and Jun Zhao. Recurrent convolutional neural networks for text classification. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, AAAI'15, pages 2267–2273. AAAI Press, 2015.

[123] Peng Zhou, Zhenyu Qi, Suncong Zheng, Jiaming Xu, Hongyun Bao, and Bo Xu. Text classification improved by integrating bidirectional lstm with two-dimensional max pooling. In Nicoletta Calzolari, Yuji Matsumoto, and Rashmi Prasad, editors, *COLING*, pages 3485–3495. ACL, 2016.

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

[125] Xiang Zhang, Junbo Zhao, and Yann LeCun. Character-level convolutional networks for text classification. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'15, pages 649–657, Cambridge, MA, USA, 2015. MIT Press.

[126] Alexis Conneau, Holger Schwenk, Loïc Barrault, and Yann LeCun. Very deep convolutional networks for text classification. In *EACL (1)*, pages 1107–1116. Association for Computational Linguistics, 2017.

[127] Rie Johnson and Tong Zhang. Convolutional neural networks for text categorization: Shallow word-level vs. deep character-level. *CoRR*, abs/1609.00718, 2016.

[128] Erion Çano and Maurizio Morisio. Role of data properties on sentiment analysis of texts via convolutions. In *WorldCist'18 - 6th World Conference on Information Systems and Technologies*, pages 330–337. Springer, Napoli, Italy, March 2018.

[129] Bo Pang and Lillian Lee. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, ACL '05, pages 115–124, Stroudsburg, PA, USA, 2005. Association for Computational Linguistics.

[130] Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational*

Linguistics: Human Language Technologies - Volume 1, HLT '11, pages 142–150, Stroudsburg, PA, USA, 2011. Association for Computational Linguistics.

[131] Xiang Zhang, Junbo Jake Zhao, and Yann LeCun. Character-level convolutional networks for text classification. *CoRR*, abs/1509.01626, 2015.

[132] Rie Johnson and Tong Zhang. Deep pyramid convolutional neural networks for text categorization. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*, pages 562–570, 2017.

[133] Jey Han Lau and Timothy Baldwin. An empirical evaluation of doc2vec with practical insights into document embedding generation. *CoRR*, abs/1607.05368, 2016.

[134] Sida Wang and Christopher D. Manning. Baselines and bigrams: Simple, good sentiment and topic classification. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Short Papers - Volume 2*, ACL '12, pages 90–94, Stroudsburg, PA, USA, 2012. Association for Computational Linguistics.

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

[136] Han Zhao, Zhengdong Lu, and Pascal Poupart. Self-adaptive hierarchical sentence model. *CoRR*, abs/1504.05070, 2015.

[137] Rie Johnson and Tong Zhang. Effective use of word order for text categorization with convolutional neural networks. *CoRR*, abs/1412.1058, 2014.

[138] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. Imagenet large scale visual recognition challenge. *Int. J. Comput. Vision*, 115(3):211–252, December 2015.

[139] Alex R Kuefler. Merging recurrence and inception-like convolution for sentiment analysis. 2017.

[140] Cícero Nogueira dos Santos and Maira Gatti. Deep convolutional neural networks for sentiment analysis of short texts. In *COLING*, pages 69–78. ACL, 2014.

[141] Ali Sharif Razavian, Hossein Azizpour, Josephine Sullivan, and Stefan Carlsson. Cnn features off-the-shelf: An astounding baseline for recognition. In *Proceedings of the 2014 IEEE Conference on Computer Vision and Pattern Recognition Workshops*, CVPRW '14, pages 512–519, Washington, DC, USA, 2014. IEEE Computer Society.

[142] Jost Tobias Springenberg, Alexey Dosovitskiy, Thomas Brox, and Martin A. Riedmiller. Striving for simplicity: The all convolutional net. *CoRR*, abs/1412.6806, 2014.

[143] Erion Çano and Maurizio Morisio. A deep learning architecture for sentiment analysis. In *2018 the International Conference on Geoinformatics and Data Analysis*. ACM, Prague, Czech Republic, April 2018.

[144] Federico Castanedo. A Review of Data Fusion Techniques. *The Scientific World Journal*, 2013:19+, 2013.

[145] A. Hassan and A. Mahmood. Deep learning approach for sentiment analysis of short texts. In *2017 3rd International Conference on Control, Automation and Robotics (ICCAR)*, pages 705–710, April 2017.

Appendix A

Systematic Literature Review Tables

Table A.1 Digital libraries inquired for primary studies

Source	URL
SpringerLink	http://link.springer.com
Science Direct	http://www.sciencedirect.com
IEEExplore	http://ieeexplore.ieee.org
ACM Digital Library	http://dl.acm.org
Scopus	http://www.scopus.com

Table A.2 Inclusion and exclusion criteria for filtering papers

Inclusion criteria
Papers addressing hybrid recommender systems, algorithms, strategies, etc.
Papers that do not directly address hybrid RSs, but describe RS engines that generate recommendations combining different data mining techniques.
Papers from conferences and journals
Papers published from 2005 to date
Papers written in English only
Exclusion criteria
Papers that do not address RSs in any way
Papers addressing RSs but not proposing or implementing any hybrid combination of different strategies or intelligent techniques.
Papers reporting only abstracts or illustration slides that miss detailed information
Grey literature

Table A.3 Final list of selected papers

P	Authors	Year	Title	Source	Publication details
P1	Wang, J.; De Vries, P. A.; Reinders, J. T. M.;	2006	Unifying User-based and Item-based Collaborative Filtering Approaches by Similarity Fusion	ACM	29th Annual International ACM SIGIR Conference on Research & Development on Information Retrieval, Seattle 2006
P2	Gunawardana, A.; Meek, C.;	2008	Tied Boltzmann Machines for Cold Start Recommendations	ACM	2nd ACM Conference on Recommender Systems, Lousanne, Switzerland, 23rd-25th October 2008
P3	Gunawardana, A.; Meek, C.;	2009	A Unified Approach to Building Hybrid Recommender Systems	ACM	3rd ACM Conference on Recommender Systems, New York, October 23-25, 2009
P4	Park, S. T.; Chu, W.;	2009	Pairwise Preference Regression for Cold-start Recommendation	ACM	3rd ACM Conference on Recommender Systems, New York, October 23-25, 2009
P5	Ghazanfar, M. A.; Prugel-Bennett, A.;	2010	An Improved Switching Hybrid Recommender System Using Naive Bayes Classifier and Collaborative Filtering	ACM	Proceedings of the International MultiConference of Engineers and Computer Scientists 2010, Vol I, Hong Kong, March 17-19, 2010
P6	Zhuhadar, L.; Nasraoui, O.;	2010	An Improved Switching Hybrid Recommender System Using Naive Bayes Classifier and Collaborative Filtering	ACM	Proceedings of the International MultiConference of Engineers and Computer Scientists 2010, Vol I, Hong Kong, March 17-19, 2010
P7	Hwang, C. S.;	2010	Genetic Algorithms for Feature Weighting in Multi-criteria Recommender Systems	ACM	Journal of Convergence Information Technology, Vol. 5, N. 8, October 2010
P8	Liu, L.; Mehandjiev, N.; Xu, D. L.;	2011	Multi-Criteria Service Recommendation Based on User Criteria Preferences	ACM	5th ACM Conference on Recommender Systems, Chicago, Oct 23rd-27th 2011
P9	Bostandjiev, S.; O'Donovan, J.; Höllerer, T.;	2012	TasteWeights: A Visual Interactive Hybrid Recommender System	ACM	6th ACM Conference on Recommender Systems, Dublin, Sep. 9th-13th, 2012
P10	Stanescu, A.; Nagar, S.; Caragea, D.;	2013	A Hybrid Recommender System: User Profiling from Keywords and Ratings	ACM	A Hybrid Recommender System: User Profiling from Keywords and Ratings
P11	Hornung, T.; Ziegler, C. N.; Franz, S.;	2013	Evaluating Hybrid Music Recommender Systems	ACM	2013 IEEE/WIC/ACM International Conferences on Web Intelligence (WI) and Intelligent Agent Technology (IAT)
P12	Said, A.; Fields, B.; Jain, B. J.;	2013	User-Centric Evaluation of a K-Furthest Neighbor Collaborative Filtering Recommender Algorithm	ACM	The 16th ACM Conference on Computer Supported Cooperative Work and Social Computing, Texas, Feb. 2013
P13	Hu, L.; Cao, J.; Xu, G.; Cao, L.; Gu, Z.; Zhu, C.;	2013	Personalized Recommendation via Cross-Domain Triadic Factorization	Scopus	22nd ACM International WWW Conference, May 2013, Brasil
P14	Christensen, I.; Schiaffino, S.;	2014	A Hybrid Approach for Group Profiling in Recommender Systems	ACM	Journal of Universal Computer Science, vol. 20, no. 4, 2014
P15	Garden, M.; Dudek, G.;	2005	Semantic feedback for hybrid recommendations in Recommendz	IEEE	IEEE 2005 International Conference on e-Technology, e-Commerce and e-Service
P16	Bezerra, B. L. D.; Carvalho, F. T.; Filho, V. M.;	2006	C2 :: A Collaborative Recommendation System Based on Modal Symbolic User Profile	IEEE	Proceedings of the 2006 IEEE/WIC/ACM International Conference on Web Intelligence
P17	Ren, L.; He, L.; Gu, J.; Xia, W.; Wu, F.;	2008	A Hybrid Recommender Approach Based on Widrow-Hoff Learning	IEEE	IEEE 2008 Second International Conference on Future Generation Communication and Networking
P18	Godoy, D.; Amandi, A.;	2008	Hybrid Content and Tag-based Profiles for Recommendation in Collaborative Tagging Systems	IEEE	IEEE 2008 Latin American Web Conference
P19	Aimeur, E.; Brassard, G.; Fernandez, J. M.; Onana, F. S. M.; Rakowski, Z.;	2008	Experimental Demonstration of a Hybrid Privacy-Preserving Recommender System	IEEE	The Third International Conference on Availability, Reliability and Security, IEEE 2008
P20	Yoshii, K.; Goto, M.; Komatani, K.; Ogata, T.; Okuno, H. G.;	2008	An Efficient Hybrid Music Recommender System Using an Incrementally Trainable Probabilistic Generative Model	IEEE	IEEE TRANSACTIONS ON AUDIO, SPEECH, AND LANGUAGE PROCESSING, VOL. 16, NO. 2, FEBRUARY 2008
P21	Maneeroj, S.; Takasu, A.;	2009	Hybrid Recommender System Using Latent Features	IEEE	IEEE 2009 International Conference on Advanced Information Networking and Applications

Continued on next page

Table A.3 – *Continued from previous page*

P	Authors	Year	Title	Source	Publication details
P22	Meller, T.; Wang, E.; Lin, F.; Yang, C.;	2009	New Classification Algorithms for Developing Online Program Recommendation Systems	IEEE	IEEE 2009 International Conference on Mobile, Hybrid, and On-line Learning
P23	Shambour, Q.; Lu, J.;	2010	A Framework of Hybrid Recommendation System for Government-to-Business Personalized e-Services	IEEE	IEEE 2010 Seventh International Conference on Information Technology
P24	Deng, Y.; Wu, Z.; Tang, C.; Si, H.; Xiong, H.; Chen, Z.;	2010	A Hybrid Movie Recommender Based on Ontology and Neural Networks	IEEE	A Hybrid Movie Recommender Based on Ontology and Neural Networks
P25	Yang, S. Y.; Hsu, C. L.;	2010	A New Ontology-Supported and Hybrid Recommending Information System for Scholars	Scopus	13th International Conference on Network-Based Information Systems
P26	Basiri, J.; Shakery, A.; Moshiri, B.; Hayat, M.;	2010	Alleviating the Cold-Start Problem of Recommender Systems Using a New Hybrid Approach	IEEE	IEEE 2010 5th International Symposium on Telecommunications (IST'2010)
P27	Valdez, M. G.; Alanis, A.; Parra, B.;	2010	Fuzzy Inference for Learning Object Recommendation	IEEE	IEEE 2010 International Conference on Fuzzy Systems
P28	Choi, S. H.; Jeong, Y. S.; Jeong, M. K.;	2010	A Hybrid Recommendation Method with Reduced Data for Large-Scale Application	IEEE	IEEE Transactions on systems, man and cybernetics - Part C: Applicatios and Reviews, VOL. 40, NO. 5, September 2010
P29	Ghazanfar, M. A.; Prugel-Bennett, A.;	2010	Building Switching Hybrid Recommender System Using Machine Learning Classifiers and Collaborative Filtering	IEEE	IEEE IAENG International Journal of Computer Science, 37:3, IJCS_37_3_09
P30	Castro-Herrera, C.;	2010	A Hybrid Recommender System for Finding Relevant Users in Open Source Forums	Scopus	IEEE 3rd International Conference on Managing Requirements Knowledge, Sept. 2010
P31	Tath, I.; Biturk, A.;	2011	A Tag-based Hybrid Music Recommendation System Using Semantic Relations and Multi-domain Information	IEEE	11th IEEE International Conference on Data Mining Workshops, Dec. 2011
P32	Kohi, A.; Ebrahimi, S. J.; Jalali, M.;	2011	Improving the Accuracy and Efficiency of Tag Recommendation System by Applying Hybrid Methods	IEEE	IEEE 1st International eConference on Computer and Knowledge Engineering (ICCKE), October 13-14, 2011
P33	Kohi, A.; Ebrahimi, S. J.; Jalali, M.;	2011	Improving the Accuracy and Efficiency of Tag Recommendation System by Applying Hybrid Methods	IEEE	IEEE 1st International eConference on Computer and Knowledge Engineering (ICCKE), October 13-14, 2011
P34	Fenza, G.; Fischetti, E.; Furno, D.; Loia, V.;	2011	A hybrid context aware system for tourist guidance based on collaborative filtering	Scopus	2011 IEEE International Conference on Fuzzy Systems, June 27-30, 2011, Taipei, Taiwan
P35	Shambour, Q.; Lu, J.;	2011	A Hybrid Multi-Criteria Semantic-enhanced Collaborative Filtering Approach for Personalized Recommendations	IEEE	2011 IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology
P36	Li, X.; Murata, T.;	2012	Multidimensional Clustering Based Collaborative Filtering Approach for Diversified Recommendation	IEEE	The 7th International Conference on Computer Science & Education July 14-17, 2012. Melbourne, Australia
P37	Shahriyary, S.; Aghabab, M. P.;	2013	Recommender systems on web service selection problems using a new hybrid approach	IEEE	IEEE 4th International Conference on Computer and Knowledge Engineering, 2014
P38	Yu, C. C.; Yamaguchi, T.; Takama, Y.;	2013	A Hybrid Recommender System based Non-common Items in Social Media	IEEE	IEEE International Joint Conference on Awareness Science and Technology and Ubi-Media Computing, 2013
P39	Buncle, J.; Anane, R.; Nakayama, M.;	2013	A Recommendation Cascade for e-learning	IEEE	2013 IEEE 27th International Conference on Advanced Information Networking and Applications
P40	Bedi, P.; Vashisth, P.; Khurana, P.;	2013	Modeling User Preferences in a Hybrid Recommender System using Type-2 Fuzzy Sets	Scopus	IEEE International Conference on Fuzzy Systems, July 2013
P41	Andrade, M. T.; Almeida, F.;	2013	Novel Hybrid Approach to Content Recommendation based on Predicted Profiles	IEEE	2013 IEEE 10th International Conference on Ubiquitous Intelligence & Computing
P42	Yao, L.; Sheng, Q. Z.; Segev, A.; Yu, J.;	2013	Recommending Web Services via Combining Collaborative Filtering with Content-based Features	IEEE	2013 IEEE 20th International Conference on Web Services
P43	Luo, Y.; Xu, B.; Cai, H.; Bu, F.;	2014	A Hybrid User Profile Model for Personalized Recommender System with Linked Open Data	IEEE	IEEE 2014 Second International Conference on Enterprise Systems

Continued on next page

Table A.3 – *Continued from previous page*

P	Authors	Year	Title	Source	Publication details
P44	Sharif, M. A.; Raghavan, V. V.;	2014	A Clustering Based Scalable Hybrid Approach for Web Page	IEEE	2014 IEEE International Conference on Big Data
P45	Xu, S.; Watada, J.;	2014	A Method for Hybrid Personalized Recommender based on Clustering of Fuzzy User Profiles	IEEE	IEEE International Conference on Fuzzy Systems (FUZZ-IEEE) July 6-11, 2014, Beijing, China
P46	Lee, K.; Lee, K.;	2014	Using Dynamically Promoted Experts for Music Recommendation	IEEE	IEEE Transactions on Multimedia, VOL. 16, NO. 5, August 2014
P47	Chughtai, M. W.; Selamat, A.; Ghani, I.; Jung, J. J.;	2014	E-Learning Recommender Systems Based on Goal-Based Hybrid Filtering	IEEE	International Journal of Distributed Sensor Networks Volume 2014pages
P48	Li, Y.; Lu, L.; Xufeng, L.	2005	A hybrid collaborative filtering method for multiple-interests and multiple-content recommendation in E-Commerce	Science Direct	Expert Systems with Applications 28 (2005) 67–77
P49	Kunaver, M.; Pozrl, T.; Pogacnik, M.; Tasic, J.;	2007	Optimisation of combined collaborative recommender systems	Science Direct	International Journal of Electronics and Communications (AEU), 2007, 433-443
P50	Albadvi, A.; Shahbazi, M.;	2009	A hybrid recommendation technique based on product category attributes	Scopus	Expert Systems with Applications 36 (2009) 11480–11488
P51	Capos, L. M.; Fernandez-Luna, J. M.; Huete, J. F.; Rueda-Morales, M. A.;	2010	Combining content-based and collaborative recommendations: A hybrid approach based on Bayesian networks	Science Direct	International Journal of Approximate Reasoning 51 (2010) 785–799
P52	Barragans-Martínez, A. B.; Costa-Montenegro, E.; Burguillo, J. C.; Rey-Lopez, M.; Mikic-Fonte, F. A.; Peleteiro, A.;	2010	A hybrid content-based and item-based collaborative filtering approach to recommend TV programs enhanced with singular value decomposition	Science Direct	International Journal of Information Sciences 180 (2010) 4290–4311
P53	Wen, H.; Fang, L.; Guan, L.;	2012	A hybrid approach for personalized recommendation of news on the Web	Science Direct	International Journal of Expert Systems with Applications 39 (2012) 5806–5814
P54	Porcel, C.; Tejeda-Lorente, A.; Martinez, M. A.; Herrera-Viedma, E.;	2012	A hybrid recommender system for the selective dissemination of research resources in a Technology Transfer Office	Science Direct	International Journal of Information Sciences 184 (2012) 1–19
P55	Noguera, J. M.; Barranco, M. J.; Segura, R. J.; Martinez, L.;	2012	A mobile 3D-GIS hybrid recommender system for tourism	Science Direct	International Journal of Information Sciences 215 (2012) 37–52
P56	Salehi, M.; Pourzaferani, M.; Razavi, S. A.;	2013	Hybrid attribute-based recommender system for learning material using genetic algorithm and a multidimensional information model	Science Direct	Egyptian Informatics Journal (2013) 14, 67–78
P57	Zang, Z.; Lin, H.; Liu, K.; Wu, D.; Zhang, G.; Lu, J.;	2013	A hybrid fuzzy-based personalized recommender system for telecom products/services	Science Direct	International Journal of Information Sciences 235 (2013) 117–129
P58	Kardan, A. A.; Ebrahimi, M.;	2013	A novel approach to hybrid recommendation systems based on association rules mining for content recommendation in asynchronous discussion groups	Science Direct	International Journal of Information Sciences 219 (2013) 93–110
P59	Lucas, J. P.; Luz, N.; Moreno, M. N.; Anacleto, R.; Figueiredo, A. A.; Martins, C.;	2013	A hybrid recommendation approach for a tourism system	Science Direct	International Journal of Expert Systems with Applications 40 (2013) 3532–3550
P60	Son, L. H.;	2014	HU-FCF: A hybrid user-based fuzzy collaborative filtering method in Recommender Systems	Science Direct	International Journal of Expert Systems with Applications 41 (2014) 6861–6870
P61	Son, L. H.;	2014	HU-FCF++: A novel hybrid method for the new user cold-start problem in recommender systems	Scopus	Engineering Applications of Artificial Intelligence 41(2015)207–222
P62	Lekakos, G.; Caravelas, P.;	2006	A hybrid approach for movie recommendation	Springer	Multimed Tools Appl (2008) 36:55–70 DOI 10.1007/s11042-006-0082-7, Springer
P63	Lekakos, G.; Giaglis, G. M.;	2007	A hybrid approach for improving predictive accuracy of collaborative filtering algorithms	Springer	User Model User-Adap Inter (2007) 17:5–40 DOI 10.1007/s11257-006-9019-0, Springer
P64	Degemmis, M.; Lops, P.; Semeraro, G.;	2007	A content-collaborative recommender that exploits WordNet-based user profiles for neighborhood formation	Springer	User Model User-Adap Inter (2007) 17:217–255, DOI 10.1007/s11257-006-9023-4, Springer
P65	Cho, J.; Kang, E.;	2010	Personalized Curriculum Recommender System Based on Hybrid Filtering	Springer	ICWL 2010, LNCS 6483, pp. 62–71, 2010, Springer

Continued on next page

Table A.3 – *Continued from previous page*

P	Authors	Year	Title	Source	Publication details
P66	Aksel, F.; Biturk, A.;	2010	Enhancing Accuracy of Hybrid Recommender Systems through Adapting the Domain Trends	Scopus	Workshop on the Practical Use of Recommender Systems, Algorithms and Technologies held in conjunction with RecSys 2010. Sept. 30, 2010, Barcelona
P67	Lampropoulos, A. S.; Lampropoulos, P. S.; Tsihrintzis, G. A.;	2011	A Cascade-Hybrid Music Recommender System for mobile services based on musical genre classification and personality diagnosis	Springer	Multimed Tools Appl (2012) 59:241–258 DOI 10.1007/s11042-011-0742-0, Springer
68	Chen, W.; Niu, Z.; Zhao, X.; Li, Y.;	2012	A hybrid recommendation algorithm adapted in e-learning environments	Springer	World Wide Web (2014) 17:271–284 DOI 10.1007/s11280-012-0187-z
P69	Sanchez, F.; Barrileo, M.; Uribe, S.; Alvarez, F.; Tena, A.; Mendez, J. M.;	2012	Social and Content Hybrid Image Recommender System for Mobile Social Networks	Springer	Mobile Netw Appl (2012) 17:782–795 DOI 10.1007/s11036-012-0399-6, Springer
P70	Zheng, X.; Ding, W.; Xu, J.; Chen, D.;	2013	Personalized recommendation based on review topics	Scopus	SOCA (2014) 8:15–31 DOI 10.1007/s11761-013-0140-8
P71	Cao, J.; Wu, Z.; Wang, Y.; Zhuang, Y.;	2013	Hybrid Collaborative Filtering algorithm for bidirectionalWeb service recommendation	Springer	Knowl Inf Syst (2013) 36:607–627 DOI 10.1007/s10115-012-0562-1
P72	Burke, R.; Vahedian, F.; Mobasher, B.;	2014	Hybrid Recommendation in Heterogeneous Networks	Springer	UMAP 2014, LNCS 8538, pp. 49–60, 2014, Springer
P73	Nikulin, V.;	2014	Hybrid Recommender System for Prediction of the Yelp Users Preferences	Springer	ICDM 2014, LNAI 8557, pp. 85–99, 2014, Springer
P74	Sarne, G. M. L.;	2014	A novel hybrid approach improving effectiveness of recommender systems	Springer	J Intell Inf Syst DOI 10.1007/s10844-014-0338-z
P75	Zhao, X.; Niu, Z.; Chen, W.; Shi, C.; Niu, K.; Liu, D.;	2014	A hybrid approach of topic model and matrix factorization based on two-step recommendation framework	Springer	J Intell Inf Syst DOI 10.1007/s10844-014-0334-3, Springer
P76	Nilashi, M.; Ibrahim, O. B.; Ithnin, N.; Zakaria, R.;	2014	A multi-criteria recommendation system using dimensionality reduction and Neuro-Fuzzy techniques	Springer	Soft Comput DOI 10.1007/s00500-014-1475-6, Springer-Verlag Berlin Heidelberg 2014

Table A.4 Questions for quality assessment

Quality Question	**Score**	**Weight**
QQ1. Does it clearly describe the addressed problems ?	yes/partly/no (1/0.5/0)	1
QQ2. Does it review related work on the problems?	yes/partly/no (1/0.5/0)	0.5
QQ3. Does it recommend future research work?	yes/partly/no (1/0.5/0)	0.5
QQ4. Does it describe each module of the system?	yes/partly/no (1/0.5/0)	1.5
QQ5. Does it empirically evaluate the system?	yes/partly/no (1/0.5/0)	1.5
QQ6. Does it present a clear formulation of findings?	yes/partly/no (1/0.5/0)	1

Table A.5 Form for data extraction

Extracted Data	Explanation	RQ
ID	A unique identifier of the form Pxx we set to each paper	-
Title	-	RQ1
Authors	-	-
Publication year	-	RQ1
Conference year	-	-
Volume	Volume of the journal	-
Location	Location of the conference	-
Source	Digital library from which was retrieved	-
Publisher	-	-
Examiner	Name of person who performed data extraction	-
Participants	Study participants like students, academics, etc.	
Goals	Work objectives	-
Application domain	Domain in which the study is applied	RQ5
Approach	Hybrid recommendation approach applied	RQ3
Contribution	Contribution of the research work	-
Dataset	Public dataset used to train and evaluate the algorithm	RQ6
DM techniques	Data mining techniques used	RQ3
Evaluation methodology	Methodology used to evaluate the RS	RQ6
Evaluated characteristic	RS characteristics evaluated	RQ6
Future work	Suggested future works	RQ7
Hybrid class	Class of hybrid RS	RQ4
Research problem	-	RQ2
Score	Overall weighted quality score	-
Other Information	-	-