Karlsruhe Institute of Technology

Concept-level Sentiment Analysis with SenticNet and Word Embeddings

Master's Thesis of

Felix Boltz

at the Department of Informatics
Institute for Program Structures and Data Organization (IPD)

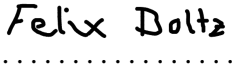
Reviewer:	Prof. Dr. Ralf H. Reussner
Second reviewer:	Prof. Dr. Harald Sack
Advisor:	Dr. Danilo Dessì

05. August 2020 – 06. March 2021

Karlsruher Institut für Technologie
Fakultät für Informatik
Postfach 6980
76128 Karlsruhe

I declare that I have developed and written the enclosed thesis completely by myself, and have not used sources or means without declaration in the text.

Karlsruhe, March 6, 2021



(Felix Boltz)

Abstract

With the shift into a more and more digitalized world over the last decade, online platforms have become increasingly popular. These platforms often allow internet users worldwide to express their opinion on all kinds of products and services. Naturally, there is a growing interest to analyze this data (e.g., to improve customer service). Researchers have developed Sentiment Analysis tools and Machine Learning models to perform this task in automatic fashion where manual evaluation is impossible. SenticNet and Word Embeddings are both semantic resources that can be deployed for several Sentiment Analysis tasks and utilized to increase the models' performance. While utilizing each of these resources led to successes, usually algorithms for Sentiment Analysis rely on only one of them. Since these two semantic resources should be naturally complementary to each other, this thesis explores a variety of strategies to incorporate SenticNet and Word Embeddings into a Machine Learning model for Sentiment Analysis. The proposed model architectures are then evaluated on a dataset of student reviews for online courses. The results indicate that the proposed combination between SenticNet and Word Embeddings improves the performance overall in the Sentiment Analysis regression task.

Zusammenfassung

Mit dem Wandel zu einer zunehmend digitalisierten Welt im letzten Jahrzehnt werden Online-Plattformen immer beliebter. Diese Plattformen ermöglichen Internetnutzern weltweit ihre Meinung zu Produkten und Dienstleistungen aller Art zu äußern. Natürlich gibt es ein wachsendes Interesse diese Daten zu analysieren (um z.B. den Kundenservice zu verbessern). Forscher haben Werkzeuge und Machine-Learning Modelle zur automatischen Sentimentanalyse entwickelt, wo eine manuelle Auswertung unmöglich ist. SenticNet und Word Embeddings sind beide Datenquellen für semantische Information, die für Sentimentanalyse eingesetzt werden können, um die Leistung der Modelle zu verbessern. Das Einbinden jeder dieser Datenquellen konnte zwar bereits Erfolge verzeichnen, normalerweise benutzen Algorithmen zur Sentimentanalyse jedoch nur eine von beiden. Da die zwei Datenquellen sich auf natürliche Weise komplementär zu einander verhalten sollten, erforscht diese Masterarbeit eine Vielfalt von Methoden SenticNet und Word Embeddings in ein Machine-Learning Modell zur Sentimentanalyse einzubinden. Die vorgeschlagenen Modellarchitekturen werden anschließend auf einem Datensatz getestet, der aus Reviews von Kursteilnehmer zu Online-Kursen besteht. Die erzielten Ergebnisse lassen vermuten, dass die vorgeschlagene Kombination von SenticNet und Word Embeddings die allgemeine Leistung der Modelle bei der Regressionsaufgabe zur Sentimentanalyse verbessert.

Contents

List of Figures

List of Tables

1 Introduction

1.1 Motivation and Overview

As the number of internet users all over the world skyrocketed over the last two decades, we are witnessing an increasing amount of user-generated data by people who express their opinion on a variety of different topics. People use online social platforms to express opinions about products and services, which can influence the perspective of their peers. Therefore, there is a growing interest to analyze this data for a variety of domains such as marketing or financial prediction. While humans can read and interpret this data easily, it is an extremely difficult task for machines, because the information is given in form of natural language which is ambiguous and unstructured. As this collectively generated intelligence may be useful for various stakeholders, but manual evaluation is usually not feasible due to the sheer amount of data, researchers try to automate this process by solving Sentiment Analysis tasks to recognize sentiments and emotions from data. Artificial intelligence-based methods make it possible to uncover users' opinions, but still present challenges because of large sources and context dependencies of data.

This thesis targets the exploration of Sentiment Analysis methods for the e-learning domain, whose interactions among students can be envisioned as a dedicated social network. The analysis of students' reviews and the related gathered information are of practical interest because automatically detecting the polarity behind a review can help other learners to assess a course's quality and decide to attend it. In other words, the sentiment behind a review plays a key role as it might influence the actions of other users.

To the end of exploring Sentiment Analysis tasks on the e-learning domain, the thesis assesses the use of SenticNet [11] resources and Word Embeddings for feeding Machine Learning (ML) models and automatically predicts the polarity and scores of students' reviews. SenticNet is a research project with the goal to build a semantic resource on commonsense knowledge. Word Embeddings assign a real-valued vector to every word in a given vocabulary.

The main challenge of this thesis is to build a regressor that benefits from the strengths of both SenticNet as well as Word Embeddings.

1.2 Sentiment Analysis

Sentiment Analysis is the process of systematically detecting positive or negative sentiments expressed in natural language. Since huge numbers of customers can express their opinions easier and quicker every day due to the emergence of online feedback systems, Sentiment Analysis has become an important tool to monitor and understand the resulting data, i.e. reviews or survey responses. Because of how inherent sentiments

and their expressions are to the verbalization of opinions, Sentiment Analysis is often also referred to as opinion mining. However, researchers have pointed out that they have slightly different meanings. Opinion mining extracts and analyzes people's opinion about an entity while Sentiment Analysis attempts to identify sentiments conveyed in natural language, measure and classify them.

Sentiment Analysis models often aim to classify the polarity of a given text, which can be positive, negative or neutral. There have been different methods proposed to measure polarity in text that range from the classification into the three mentioned categories positive, negative and neutral to more fine-grained approaches which assign sentiments with a scalar polarity value, not only indicating that the sentiment is rather positive or negative, but also providing a scale of how positive or negative it is compared to a different expression. Finally, there are also models that focus on detecting emotional states such as angry, happy and sad.

1.2.1 Classification Level

Sentiment Analysis models also vary by which level of classification is used based on the granularity of data observation. The three main classification levels are document-level, sentence-level and aspect-level [25].

Document-level models try to classify an opinion expressed in a document. They treat the whole document as a basic information unit talking about the same topic.

In contrast sentence-level models aim to classify the sentiments expressed in each sentence. There is no fundamental difference between these two approaches since sentences can be treated as short documents, the used level or document length just needs to be chosen appropriately for each application.

Aspect-level models additionally focus on which particular aspect the sentiment is expressed about. Therefore, the first step is to identify mentioned entities and their aspects. Subsequently, the detected sentiments can be classified with respect to the specific aspect of the entity at stake. An often-quoted example is the sentence: *The voice quality of this phone is not good, but the battery life is long.* An aspect-level model should be able to first identify the two mentioned aspects of the entity phone, which are voice quality and battery life, in order to then do classification on the respective detected sentiments. In this case a good model should extract a positive polarity for the aspect battery life and the opposite for the phone's voice quality.

1.2.2 Classification and Regression Techniques

Classification techniques for sentiment classification can be roughly divided into ML approaches, lexicon-based approaches and hybrid approaches [24].

ML approaches use a variety of algorithms and linguistic features and depending on the availability of labeled training data ML models can be either trained supervised or unsupervised. Chapter 2 will cover the Deep Learning tools used in this thesis in detail. Lexicon-based approaches utilize sentiment lexicons which are collections of known sentiment expressions and their respective classification. The approaches can be based on dictionaries or text corpora using statistical or semantic methods and rely on rules

to extract sentiment polarities. Hybrid approaches combine both ML and lexicon-based methods in a beneficial way and are widely used due to the power of ML algorithms and the availability of large sentiment lexicons.

1.2.3 SenticNet and Word Embeddings

SenticNet aims to tackle Sentiment Analysis on a concept-level. It is built on ConceptNet [35] which is a directed graph, constructed by collecting a vast number of facts contributed in natural language by volunteers all over the world and extracting the information these contain into a semantic network. This graph structure can then be used for Natural Language Processing (NLP) to serve as a large commonsense knowledge base. The graph nodes are concepts which represent sets of closely related natural language phrases, its edges represent assertions of common sense about the concepts they connect. SenticNet builds on this work by adding affective information to the concepts with relatively strong positive or negative polarity. Among other advantages this approach can help to understand the sentiments in domain specific language which depend on their own concepts to express different opinions.

Word Embeddings are a different technology to model syntactic and semantic properties of words and can be employed for Sentiment Analysis tasks. They are a class of language models where individual words are mapped to vectors of real numbers in a predefined vector space. While there is a whole range of different methods to generate this mapping, all of them aim to accomplish that words with similar meaning will have similar representations within the vector space. This makes them a very useful tool for a variety of NLP tasks including Sentiment Analysis [29].

1.2.4 Applications of Sentiment Analysis

As a part of the Natural Language Processing (NLP) and more specific text mining research field Sentiment Analysis can be used for a countless number of applications. Whenever there is a huge amount of data available and an interest to analyze or monitor the expressed intentions in it, Sentiment Analysis approaches can help to accomplish this goal, especially in cases where the task is unfeasible manually. As a result, Sentiment analysis methods are used to analyze product reviews, stock markets, news articles or political debates and have proven to be very useful for recommendation systems [1, 22, 32].

1.2.5 Datasets

The availability of high quality datasets is an important topic in the text mining research field, especially for ML approaches. The main sources of data for Sentiment Analysis tasks are from product reviews, most of them coming from review websites or any website where user ratings or comments on products or services can be collected (e.g. YouTube). Another important source of data are social network and blogging sites where people all over the world share and discuss all kinds of topics publicly visible.

1.3 Goal

The goal of this master thesis is to examine how different existing approaches for Sentiment Analysis can be combined. More specifically, the goal is to build a highly accurate regressor utilizing the strengths of different data structures in order to predict the score for user reviews in the e-learning domain in the form of a few sentences. In order to achieve this goal it is explored how SenticNet resources can be fed to a ML model, so it can learn how to detect affective information in said reviews and how to add Word Embeddings in a complementary way to improve its performance.

Although the proposed methodology is domain independent, the study is performed within the e-learning domain, where the exploitation of Sentiment Analysis approaches still needs to close the gap with other domains [41]. Additionally, it is analyzed how effective the use of SenticNet' resources as well as several Word Embeddings in this particular domain is.

1.4 Outline

The remaining parts of the thesis are organized as follows. Chapter 2 covers the fundamentals. It explains the Deep Learning tools which are used in this thesis, it discusses several commonly used Word Embeddings and it describes the SenticNet research project in detail. Chapter 3 will cover related work in this research field. In chapter 4 the different proposed approaches for this task are discussed. The results achieved on the Udemy dataset are presented and discussed in chapter 5. Finally, the last chapter of the thesis summarizes the results and conclusions of this work and mentions related paths for research projects that might still be worth exploring in the future.

2 Fundamentals

2.1 Machine Learning

Machine Learning is the collective term for methods that allow computer algorithms to learn automatically from experience. ML algorithms identify underlying structures, similarities or other patterns without being explicitly programmed to do so and can be divided into the three basic paradigms: supervised learning, unsupervised learning and reinforcement learning. A supervised learning approach requires labeled training data, which means the data consists not just of example inputs for the model, but also the desired outputs. Similarly to how a student learns to improve on a subject due to a teacher telling him what is right or wrong, the ML models learn a general rule that maps the inputs to outputs. Supervised ML algorithms use training data to build models and solve their task using domain specific knowledge. In unsupervised learning no labels are available to the ML model, so it has to discover patterns in its input (e.g. clustering). In reinforcement learning the ML algorithm interacts with a dynamic environment, receives a feedback for its action and learns from the experience to perform specific tasks. Since giving a comprehensive overview for the rapidly developing ML research field is unfeasible here, this chapter covers the most important ML tools used for the supervised learning approaches in this thesis.

2.1.1 Artificial Neural Networks

Like most current approaches applied in NLP this work uses Artificial Neural Networks (ANN), therefore fundamentals about the employed neural network architectures are briefly discussed.

2.1.1.1 Perceptron

A perceptron is a binary classifier and the atomic building block of essentially all neural networks. It is an algorithm defining a linear function, that maps its real vector inputs to a single binary value output. The perceptron separates its vector space into two classes depending on whether it is above or below the plane described by the linear function. Equation (2.1) expresses the perceptron in mathematical terms.

$$f(\vec{x}) = o = \begin{cases} 0 & \text{if } \vec{w}\vec{x} + b \leq 0 \\ 1 & \text{otherwise} \end{cases} \tag{2.1}$$

The possible output values $(0, 1)$ for o represent the class assignment for each input vector $\vec{x}$. Both $\vec{x}$ and the weight vector $\vec{w}$ are element-wise multiplied using the dot product.

Together with the offset b the weight vector $\vec{w}$ defines the linear function and its values are learned by optimizing a cost function based on input vectors and their corresponding outputs in the training process.

2.1.1.2 Feed-forward Neural Networks

Because a single linear function is not sufficient to solve more complex classification tasks, usually multiple perceptrons are layered. In these networks, results of previous perceptrons serve as the input of following ones and the information only moves in the direction forward from the input vector without cycles, hence the networks' name. All feed-forward neural networks consist of an input layer, one or more hidden layers and an output layer. While the size of the initial input vector determines the shape of the input layer, the shape of the output layer can be chosen to fit the desired classification task.

Since the perceptron can also serve as a node in the neural network and transformations of the input vectors might be desired, which are not restricted to a single real value output, the perceptron structure has to be modified. By exchanging the weight vector $\vec{w}$ with a weight matrix $W \in \mathbb{R}^{m \times n}$, where n is the input dimension and m the desired output dimension, results with arbitrary dimension can be produced. Additionally, the offset b also has to be replaced with a bias vector $\vec{b}$. An entry in the weight matrix W signals an existing connection between respective units in a layer.

Since linear functions are closed under the process of chaining them, the concatenation of perceptrons is not yet capable of solving complex classification tasks. The step function needs to be changed to an activation functions which contains a nonlinearity part. Without it the networks computational power would be equal to a single linear function. The derivative of the activation function has a key role in the optimization process for neural networks. Frequently used activation functions are the sigmoid function, the hyperbolic tangent function or the rectified linear activation function (ReLU).

2.1.1.3 Optimization

While the previous section briefly discussed the general structure of neural networks, the optimization process by which the parameters are adapted during training still needs to be addressed. To improve the network's weights towards a better performing parameter setting, the performance of the current parameter set has to be determined, the parameters gradually adjusted and the performance evaluated again. This is done using a cost function (also called loss or objective function), which measures the performance of the network. The cross entropy is a very common loss function for classification tasks. The mean absolute error (MAE) and the mean squared error (MSE) are common loss functions for regression tasks and will be used in this thesis. Their mathematical definitions are depicted in equation (2.2) and (2.3) where N denotes the number of available training samples, y_i the network's desired output and o_i the current output.

$$MAE = \frac{\sum_{N=1}^{i} |y_i - o_i|}{N} \tag{2.2}$$

$$MSE = \frac{\sum_{N=1}^{i}(y_i - o_i)^2}{N} \tag{2.3}$$

The partial derivative of the loss function with respect to each parameter (the weights) describes how the network's output changes for small adjustments to the respective parameter. All parameters may need to be altered in order to minimize the cost function. Therefore, all parameters are adjusted step-wise in the direction of the negative gradient till a local minimum is reached. This is why the approach is called gradient descent.

In order to avoid to jump over a minimum because the step-size for adjustments is too big usually the step-size is decreased with increasing step count. The partial derivatives are calculated by using backpropagation. An algorithm that speeds up the calculation by reusing previously calculated gradients. The error provided by the loss function is propagated reversely through the whole network, giving the approach its name. While the basic gradient descent approach uses all available training examples to calculate a single step, in many cases this is unfeasible due to the computational cost. Only using a few examples which are called mini-batches to estimate the true values is a possible way to solve this issue. This approach is called stochastic gradient descent.

2.1.1.4 Recurrent Neural Networks

Beside the feed-forward networks there are also recurrent neural networks (RNN), which additionally have an internal state (memory), allowing them to better process sequential data. Being able to take the previous activation state (also called hidden state) into account has proven very useful for processing word sequences where the meaning of a phrase can completely change based on previous or subsequent words. Figure 2.1 shows the computational graph of a recurrent network. It maps an input sequence of x values to a corresponding sequence of output o values. A loss function L measures how far each o is from the training target y. The RNN has input to hidden connections parametrized by a weight matrix U, hidden-to-hidden recurrent connections parametrized by a weight matrix W , and hidden-to-output connections parametrized by a weight matrix V.

Equation (2.4) defines the forward propagation of this network. First the weight matrix W and the hidden state $\vec{h}^{(t-1)}$ are multiplied as well as the new input $\vec{x}^{(t)}$ and input to hidden weight matrix U, then the results are summed up with the offset vector $\vec{b}$. The result of equation (2.4) and the nonlinearity σ are then used to calculate the new hidden state $\vec{h}^{(t-1)}$ in equation (2.5). Finally, the output $\vec{o}^{(t)}$ is calculated in equation (2.6) by multiplying $\vec{h}^{(t)}$ with the hidden-to-output weight matrix V and adding the bias vector $\vec{c}$.

$$\vec{a}^{(t)} = W\vec{h}^{(t-1)} + U\vec{x}^{(t)} + \vec{b} \tag{2.4}$$

$$\vec{h}^{(t)} = \sigma(\vec{a}^{(t)}) \tag{2.5}$$

$$\vec{o}^{(t)} = V\vec{h}^{(t)} + \vec{c} \tag{2.6}$$

RNNs can also be trained using backpropagation by unfolding the recurrent connection (displayed in Figure 2.1). A common problem with increasing layer numbers is a tendency

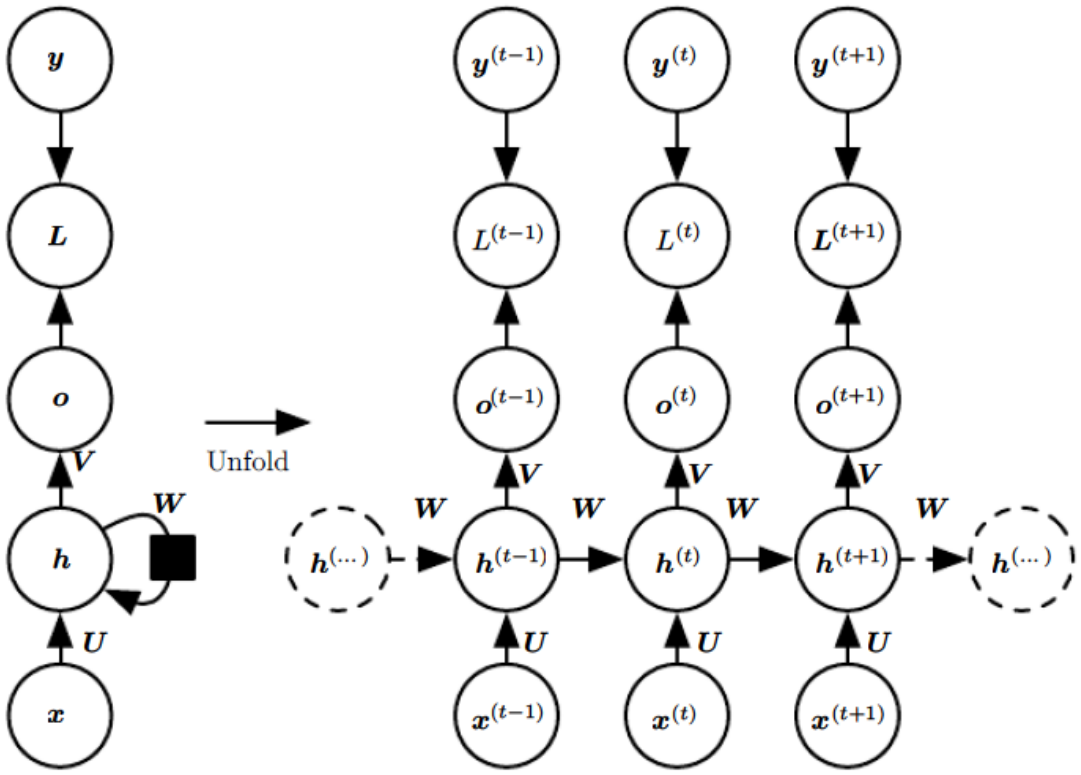


Figure 2.1: (Left)The RNN and its loss drawn with recurrent connections. The black square indicates a delay of a single time step. (Right)The same seen as an timeunfolded computational graph, where each node is now associated with one particular time instance; image taken from [19]

for the backpropagated gradient to either vanish or explode dependent on the nonlinearity component. On the one hand gradients of nonlinearities smaller than one result in vanishing gradients, on the other large gradients will let the values explode. While large gradients can be handled by cutting them off, there is no similar easy solution for vanishing gradients. RNNs using LSTM units at least partially solve the vanishing gradient problem and will be briefly discussed in the next section.

2.1.1.5 Long Short Term Memory

To tackle the vanishing gradient problem Hochreiter and Schmidhuber [21] introduced the Long Short Term Memory as an extension to RNNs. The core idea of LSTMs is adding a memory cell to the RNN structure which decides how much the current and how much the hidden state are taken into account. This can preserve a state over a longer time period and also allows the gradients to flow unchanged.

Figure 2.2 displays the schematic structure of a LSTM cell. The LSTM cells are recurrently connected to each other. A regular artificial neuron unit computes the input feature. If the input gate allows it, its value can be accumulated into the state. The state unit has a linear self-loop controlled by the forget gate and can also be used as an extra input to the gating units. Finally, the output gate can shut off the output of the cell. LSTMs are very frequently used for a variety of NLP tasks and it should be mentioned that many different versions exist.

LSTM units can be arranged in multiple layers, but the signal can also be processed in forward and backward direction (Bidirectional Long Short Term Memory). In this thesis it will be used to process a sentence representation using Word Embeddings.

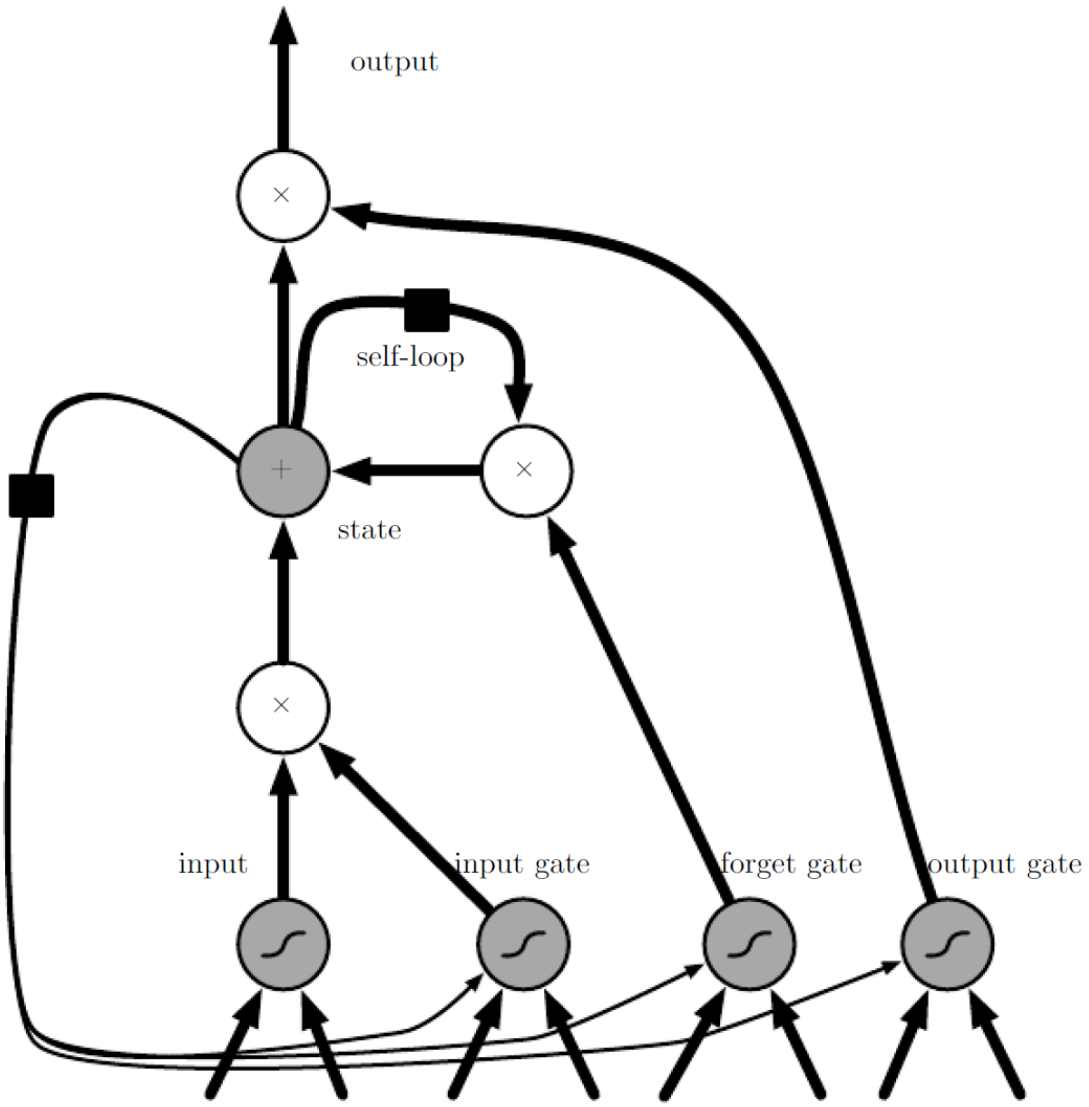


Figure 2.2: Block diagram of the LSTM recurrent network cell. The black square indicates a delay of a single time step. Image taken from [19]

2.2 Word Embeddings

A Word Embedding is a learned representation for text where each word is mapped to one real-valued vector. While it is easy to come up with any such representation, (e.g. one-hot vectors for every word in the vocabulary), key to the approach is the idea of using a dense distributed representation for each word. The distributed representation can be learned based on the usage of words. The linguistic reasoning behind this approach is that words with similar context will have similar meanings. Word Embeddings provide an easy way to measure the similarity of words by comparing their respective vectors and will allow for some arithmetic operations in a lot of cases (e.g. starting with the word vector for 'king', subtracting the vector for 'man' and then adding the vector for 'woman' will lead to a vector representation that is hopefully closer to the vector for 'queen' than to any other word in the space). There is a whole range of methods to generate dense distributed word representations with different strengths. A few of the most prominent and widely used Word Embeddings will be discussed below.

2.2.1 Word2vec

Mikolov et al. [27] introduced a software package called Word2vec that includes the two algorithms continuous bag of words (CBOW) and skip-gram as well as the two training methods negative sampling and hierarchical softmax.

2.2.1.1 Continuous bag-of-words

CBOW aims to predict a target word w_c from a fixed size window of its surrounding context words. To do so, it averages the vectors of all context words and calculates a similarity to the target word using the dot product. Based on the similarity measurement the objective function J in equation (2.7) is defined where u_i denotes output vector representation of word w_i and $\hat{v}$ the average over all input word vectors in the context with size c. Starting with one-hot or random vectors the distributed word representation can then be learned by using a neural network and minimizing the loss function via gradient descent on a large corpus of meaningful sentences. In contrast to the standard bag-of-words model CBOW uses a continuous distributed representation of a target words context. The chosen number of dimensions of the resulting word representations is dependent on the vocabulary size and is usually set to be between 50 and 100.

$$J_{CBOW} = -u_c^T \hat{v} + \log \sum_{j=1}^{|V|} \exp(u_j^T \hat{v}) \tag{2.7}$$

2.2.1.2 Skip-gram

The skip-gram algorithm essentially aims to accomplish the opposite of CBOW by predicting the distribution probability of all context words given the center word. More precisely, the current center word is used as an input to a log-linear classifier with continuous projection layer which predicts the context words in a window of fixed size around the

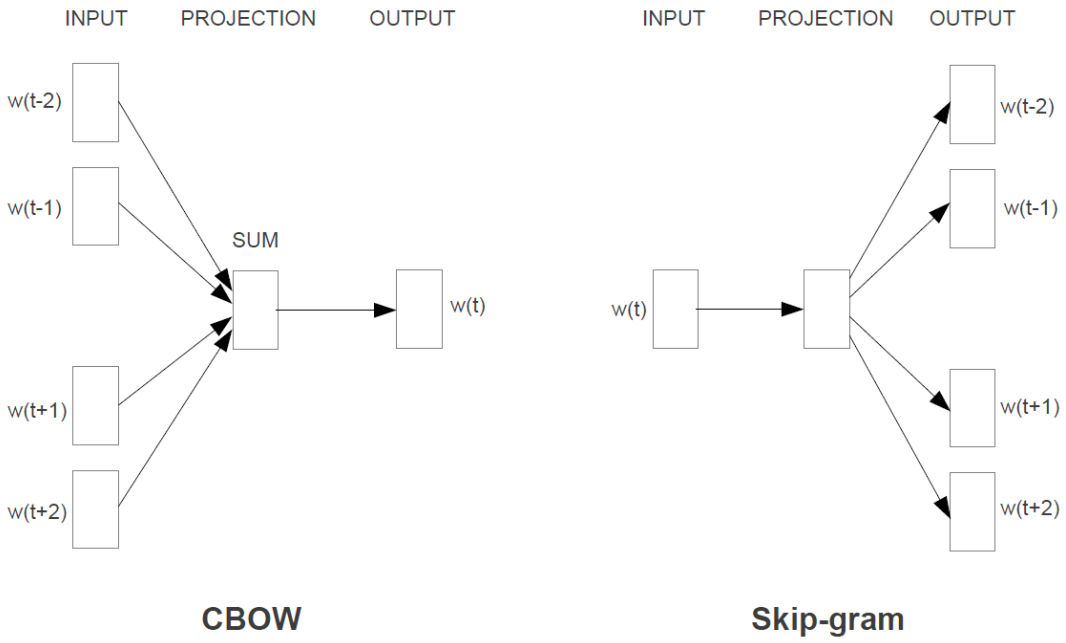


Figure 2.3: CBOW and skip-gram architecture, image taken from [27]

center word. As with CBOW an objective function J is defined (displayed in equation (2.8)) maximizing the probability of predicting the correct context words and used to learn the distributed word representation via gradient descent on the input corpus. It should be noted that both algorithms CBOW and skip-gram can be used to produce a distributed word representation and in both cases there are actually two representations learned for each word, one for word as an input vector and one as the output vector of the respective model. The architecture of both algorithms is displayed in Figure 2.3.

$$J_{skip} = -\sum_{j=0, j\neq m}^{2m} u_{c-m+j}^T v_c + 2m \log \sum_{k=1}^{|V|} \exp(u_k^T v_c) \tag{2.8}$$

2.2.1.3 Subsampling and negative sampling

To avoid sampling very frequent words that provide rather little information (e.g. 'in', 'the' or 'a') each word in the training set is discarded with a probability given by equation (2.9) where the probability for each word w_i is calculated using a chosen threshold t and the frequency of the word $f(w_i)$ [26].

$$P(w_i) = 1 - \sqrt{\frac{t}{f(w_i)}} \tag{2.9}$$

Since the updates to the vectors of very frequent words become increasingly small anyways, this subsampling strategy reduces the computational complexity without affecting the resulting representations for frequent words significantly.

In addition negative sampling is used, which is based on the skip-gram model but defines a new objective function in order to maximize the probability of a word and its context being in the corpus data if it indeed is, and maximize the probability of a word and its context not being in the corpus if it indeed is not. Negative Samples are created 'on the fly' during training by pairing the center word with a random word from the vocabulary. The motivation to do this is how computationally expensive the normalization factor used in the objective functions given above is. So, instead of looping over the entire vocabulary for every training step, now just several negative examples need to be sampled. Equation (2.10) and (2.11) show the updated objective function for the skip-gram and CBOW model. σ denotes the sigmoid function, k the selected samples and $\tilde{u}_k^T$ their respective output word vector.

$$J'_{skip} = -\log \sigma(u_{c-m+j}^T v_c) - \sum_{k=1}^{K} \log \sigma(-\tilde{u}_k^T v_c) \tag{2.10}$$

$$J'_{CBOW} = -\log \sigma(u_c^T \hat{v}) - \sum_{k=1}^{K} \log \sigma(-\tilde{u}_k^T \hat{v}) \tag{2.11}$$

2.2.1.4 Hierarchical softmax

Hierarchical softmax is an efficient approximation of the full softmax with the main advantage of only having to evaluate $\log_2(W)$ nodes in the neural network instead of evaluating W output nodes in order to obtain the probability distribution. To achieve this hierarchical softmax uses a binary tree to represent all words in the vocabulary. The leaves of the tree are words and there is a unique path from the root to each leaf. Each of the remaining nodes represents the relative probabilities of its child nodes. In this model, the probability of a word w given a vector w_i, is equal to the probability of a random walk starting in the root and ending in the leaf node corresponding to w. Mikolov et al. used a binary Huffman tree in their model.

2.2.2 FastText

Bojanowski, Grave et al. introduced with FastText [4] an extension to the previously discussed Skip-gram model, where each word is represented as a bag of character n-grams. Assigning each word with a distinct vector ignores the morphology of words. Many word forms occur rarely in the training corpus, making it difficult to learn a good word representation for them. Since many word formations follow rules, these vector representations can be improved by including character level information. To this end FastText introduces a subword model in which every word is represented by a set of n-grams.

In order to distinguish prefixes and suffixes from other characters sequences the boundary symbols < and > are included to indicate the beginning and end of words. Additionally, the word itself is also included in the set of its n-grams to learn a representation for each word on its own, besides its n-gram representation. Taking the word 'where' and n = 3 as

an example, its subword model will then be given by the character n-grams: <wh, whe, her, ere, re> and the special sequence <where>. Each n-gram will be associated with a vector representation. A word is then represented by the sum of the vector representations of its n-grams. The use of this subword model allows sharing learned representations for n-grams across words and therefore learning more reliable representations for rare words.

2.2.3 GloVe

Global Vectors (GloVe) [28] consists of a weighted least squares model that is trained using global word to word co-occurrence counts. The model produces a word vector space with meaningful sub-structure, allowing it to outperform Word2vec on word analogy tasks.

2.2.3.1 Co-occurrence matrix

An entry $X_i j$ in the co-occurrence matrix X indicates the number of times word j occurs in the context of a fixed size window of word i. Obtaining these co-occurrence matrix entries requires a single pass through the entire corpus in order to count the appearances. While this can be a computationally expensive operation for large corpora, it is a one-time cost and it can be done separately before the training process.

2.2.3.2 Least Squares Objective

Similar to Word2vec the objective function is defined to avoid the computationally expensive summation over the entire vocabulary required by the normalization factor in the cross-entropy-loss. Instead, the least square objective depicted in equation (2.12) is used.

$$J_{GloVe} = \sum_{i,j} f(X_{ij})(\vec{u}_j^T \vec{v}_i - \log X_{ij})^2 \tag{2.12}$$

The objective function allows the GloVe model to efficiently leverage global statistical information by training only on the nonzero elements in a word to word co-occurrence matrix.

2.2.4 BERT

BERT [16] stands for Bidirectional Encoder Representations from Transformers. Its model architecture is a multi-layered bidirectional Transformer encoder based on the implementation described in Vaswani et al. (2017) [43]. The framework consists of two major steps called pre-training and fine-tuning. As a starting point BERT uses WordPiece Embeddings [44]. The sum of an input token's embeddings with its corresponding segment and position embeddings builds its input representation for the pre-training step. A visualization including the special tokens [CLS] indicating the start of a new sequence and [SEP] to separate sentences can be seen below (Figure 2.4).

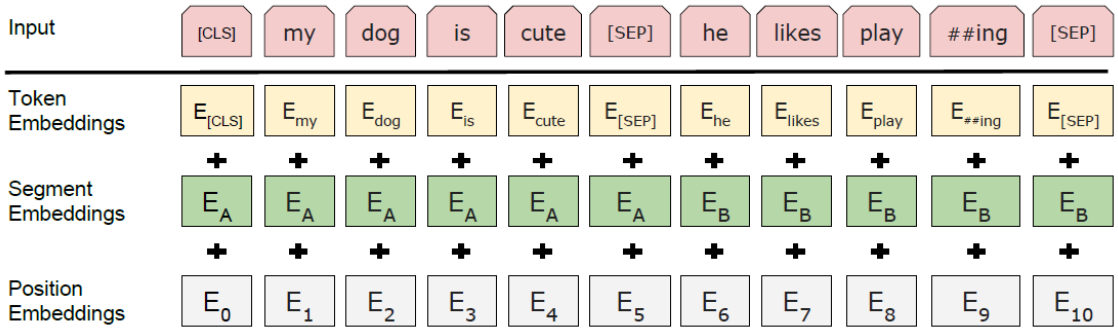


Figure 2.4: BERT input encoding, figure taken from [16]

2.2.4.1 Pre-training

BERT is pre-trained using two unsupervised tasks the model learns simultaneously. For the first task some percentage of the input tokens is masked at random. Then the model learns to predict those masked tokens. The final vectors corresponding to the masked tokens are then fed into an output softmax over the vocabulary. This procedure allows the model to train a deep bidirectional representation and is called masked LM (MLM). The second task of pre-training is called Next Sentence Prediction (NSP) which trains the model in order to make a binary prediction about the next sentence B being the actual sentence following the first sentence A. In 50% of all cases B is the actual next sentence that follows A (labeled as IsNext), the other 50% of the time it is a random sentence from the corpus (labeled as NotNext). NSP trains the model to better understand the relationship between two sentences which many downstream tasks such as Question Answering (QA) are based on. The pre-training for BERT is performed on the BooksCorpus [46] and English Wikipedia.

2.2.4.2 Fine-tuning

In the fine-tuning step supervised training is done for the specific task BERT shall be used for. For this purpose the output layers are replaced and training is done on a task specific dataset. The fine-tuning step and BERT's deep bidirectional architecture allows the same pre-trained model to successfully tackle a broad set of NLP tasks. A visualization of BERT's pre-training and fine-tuning process is displayed below (Figure 2.5).

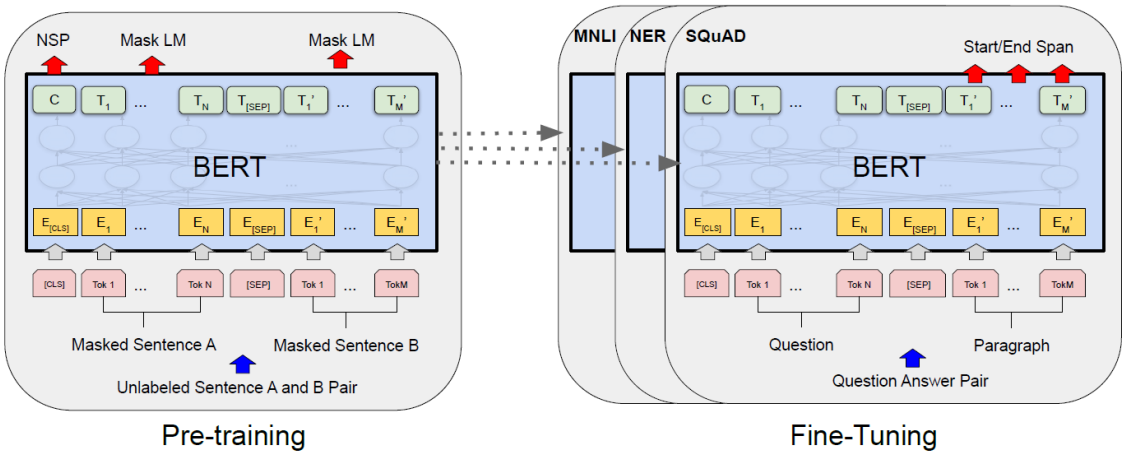


Figure 2.5: Overall pre-training and fine-tuning procedures for BERT, figure taken from [16]

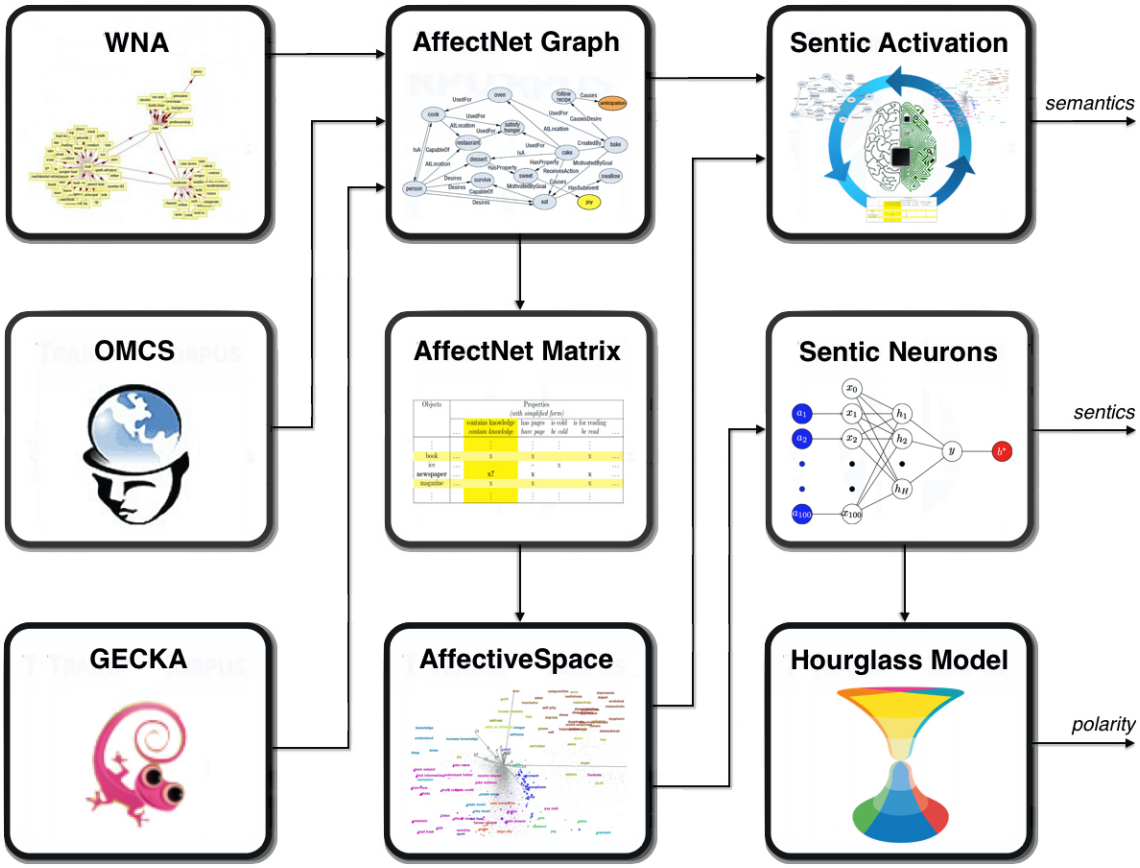


Figure 2.6: Graph structure of the SenticNet project, image taken from the SenticNet website

2.3 SenticNet

SenticNet is a semantic resource built on the combined affective commonsense knowledge from WordNet-Affect (WNA), OMCS (Open Mind Common Sense) and GECKA (Game Engine for Commonsense Knowledge Acquisition). What each of these sources contain and how the SenticNet research project utilizes them to create a useful tool for Sentiment Analysis is briefly discussed in this section. Figure 2.6 shows a graph structure of the different modules in the SenticNet research project from its website [1] for orientation.

2.3.1 WordNet-Affect

WordNet-Affect (WNA) [36] is a linguistic resource with a lexical representation of affective knowledge. It is built on WordNet, a lexical database of semantic relations between words which covers the entire English lexicon. Those relations include synonyms, hyponyms and meronyms. Synonyms are grouped into so called synsets, a set of one or more synonyms which carry the same meaning. A subset of WordNet synsets has been chosen manually to represent affective concepts and build the core of WordNet-Affect. The core has then been

[1] https://sentic.net/downloads/

A-Labels	Examples
EMOTION	noun anger#1, verb fear#1
MOOD	noun animosisy#1, adjective amiable#1
TRAIT	noun aggressiveness#1, adjective competitive#1
COGNITIVE STATE	noun confusion#2, adjective dazed#2
PHYSICAL STATE	noun illness#1, adjective all_in#1
EDONIC SIGNAL	noun hurt#3, noun suffering#4
EMOTION-ELICITING SITUATION	noun awkwardness#3, adjective out_of_danger#1
EMOTIONAL RESPONSE	noun cold_sweat#1, verb tremble#2
BEHAVIOUR	noun offense#1, adjective inhibited#1
ATTITUDE	noun intolerance#1, noun defensive#1
SENSATION	noun coldness#1, verb feel#3

Figure 2.7: A-labels of WordNet-Affect, table taken from [36]

extended by examining how WordNet's relations either preserve or change the affective meaning of the synsets already in WordNet-Affect. The additional affective information to define the precise affective meaning of a synset in WordNet-Affect is added by assigning these synsets with one or more out of 11 different affective labels (a-labels). A complete list of all labels can be seen in Figure 2.7.

2.3.2 OMCS and ConceptNet

OMCS is a knowledge acquisition system that aims to gather commonsense knowledge from the general public over the web. Volunteers use the OMCS website to provide commonsense statements that range from simple facts to rules, stories and descriptions [34]. Since the start of the project in 2000 OMCS has had more than 17,000 contributors and has collected more than a million pieces of commonsense data in the English language [20]. The OMCS project has also been expanded to acquire commonsense knowledge in other languages.

ConceptNet is a semantic network built on the OMCS corpus to make it accessible to AI applications. It is a directed graph whose nodes are called concepts, which consist of either a word or a short natural language phrase. Its edges represent assertions, which express common sense relationships about the two concepts they connect. The edges are labeled with one out of a selected closed class of 36 generalized relations, such as IsA, SimilarTo and PartOf. If the relation is symmetric, i.e. SimilarTo, the edge connecting two nodes is bidirectional. Additionally, each assertion has a score representing its reliability [35].

While ConceptNet is originally intended as just a parsed representation of the OMCS corpus, it has since expanded to include knowledge from a variety of different sources. In its current version ConceptNet 5.5 contains over 21 million edges and over 8 million nodes. Its largest source of input is Wiktionary with 18.1 million provided edges.

The authors of SenticNet have been continuously optimizing their work and expanding the amount of concepts SenticNet assigns with their respective sentiments from 6,000 concepts in 2010 up to 200,000 concepts in its current 6th version.

2.3.3 GECKA: using a game environment to collect knowledge

Over the past few years, Games with a purpose (GWAPs) have been developed to gather information from casual gamers on tasks that are relatively easy to complete for humans, but are still problematic for machines. Ideally, these games can be a source for knowledge on a multitude of tasks where the players have fun while contributing to a useful purpose.

On this topic Cambria et. al [7] created a new GWAP concept called GECKA, that provides a framework for users to create their own GWAPs. While the problem with current GWAPs is that the gathered information is often only applicable to the specific stimuli encountered during gameplay, GECKA aims to solve that, so the collected knowledge is actually reusable and multi-purpose.

2.3.4 Blending WNA and ConceptNet

The collected information represented in WNA and ConceptNet is combined in SenticNet by aligning the lemma forms (relations and concepts) of both datasets and blending their respective matrix representations. The synonyms, hyponyms and meronyms of WNA are mapped to their corresponding relation type in ConceptNet, allowing the combined information to coexist in one matrix [6].

Originally, singular value decomposition (SVD) was performed on the resulting matrix, followed by a trial and error search to remove components with relatively small variation. The remaining 50 principal components of this truncated SVD then yield a 50-dimensional vector space called AffectiveSpace where each dimension represents a different way to make binary distinctions among emotions.

Since SenticNet's first version, the amount of concepts present in the ConceptNet dataset has drastically increased and SVD can not be computed on a matrix that is too high-dimensional and too sparse. Therefore the SVD operation has been replaced with random projection which achieves the same dimensionality reduction while preserving the pairwise distances in the original space with high probability [5]. The number of dimensions for AffectiveSpace has also been increased to 100.

2.3.5 Hourglass of Emotions

SenticNet uses an affective categorization model called the Hourglass of Emotions. In the model emotions are organized around the four independent dimensions Pleasantness, Attention, Sensitivity and Aptitude. Each of these four dimensions is further categorized by six different levels of activation, which determine the intensity of the respective emotion resulting in a total of 24 elementary emotion labels.

The categorization model is applied by a clustering approach in the 50-dimensional AffectiveSpace described above. First, for every emotion category in the model an ad-hoc category is created. An ad-hoc category is a synthetic concept made of the weighted sum of existing concepts in the space. This set of concepts for each sentic level is extracted from WordNet's list of hypernyms and hyponyms. The resulting ad-hoc category vectors are then used as initial centroids for clustering the vector space by means of sentic medoids [8]

to find related concepts in AffectiveSpace [12]. A schematic representation of the hourglass model displaying all 24 different categories can be seen in Figure 2.8.

While the previous sections cover SenticNet's fundamental building blocks that have been used since its first version, the following sections focus on the specifics of SenticNet's latest version (SenticNet 6).

2.3.6 Key Polar State Specification

In order to assign polarity values and emotion labels to relevant concepts in AffectiveSpace a discrete path between each key polar state and its opposite (e.g., CLEAN and DIRTY) throughout the vector space is calculated. To include as many semantically and affectively relevant concepts as possible this path needs to follow the topological structure of the vector space. Such a path is found by using regularized k-means, that is a minimization of a standard k-means cost function with the addition of a regularization term that considers the distance between ordered centroids (the emotion label clusters of the hourglass model). Since there are many low-intensity concepts towards the center of the vector space only the first 20 nearest concepts to each of the two polar states are considered. Based on the average distance between the concepts of the path and the key concepts in AffectiveSpace that represent the 24 emotion labels of the hourglass model, the key polar states are assigned with the emotion label of their respective closest category. All the concepts connected to these key polar states then inherit the same emotion and polarity classification by transitivity [11].

2.3.7 Conceptual Primitives

The idea of conceptual primitives is to generalize SenticNet's commonsense knowledge by decomposing it far enough to reach a level of conceptual structure. The elements on this level represent bits of basic understanding and commonsense that can be used to build a variety of different meanings. Since there are countless ways to express the same concept in natural language, the motivation of this process is to move away from the almost impossible task of collecting a comprehensive list. Instead lemmatization and analogical reasoning are used to cover lexical and semantic reflections respectively [9]. Developing such a knowledge representation for general commonsense reasoning would be an immense task, however it is feasible in the context of Sentiment Analysis because only subjective commonsense concepts (i.e., concepts that convey either a positive or negative polarity) need to be encoded.

"The idea behind this generalization is that there is a finite set of mental primitives for affect-bearing concepts and a finite set of principles of mental combination governing their interaction." [10]

2.3.8 Inference of Sentiment Polarities of New Concepts

With the use of conceptual primitives there is no longer the need to infer polarity based on emotion links in the semantic network. Instead polarity bearing states (e.g. INTACT) and

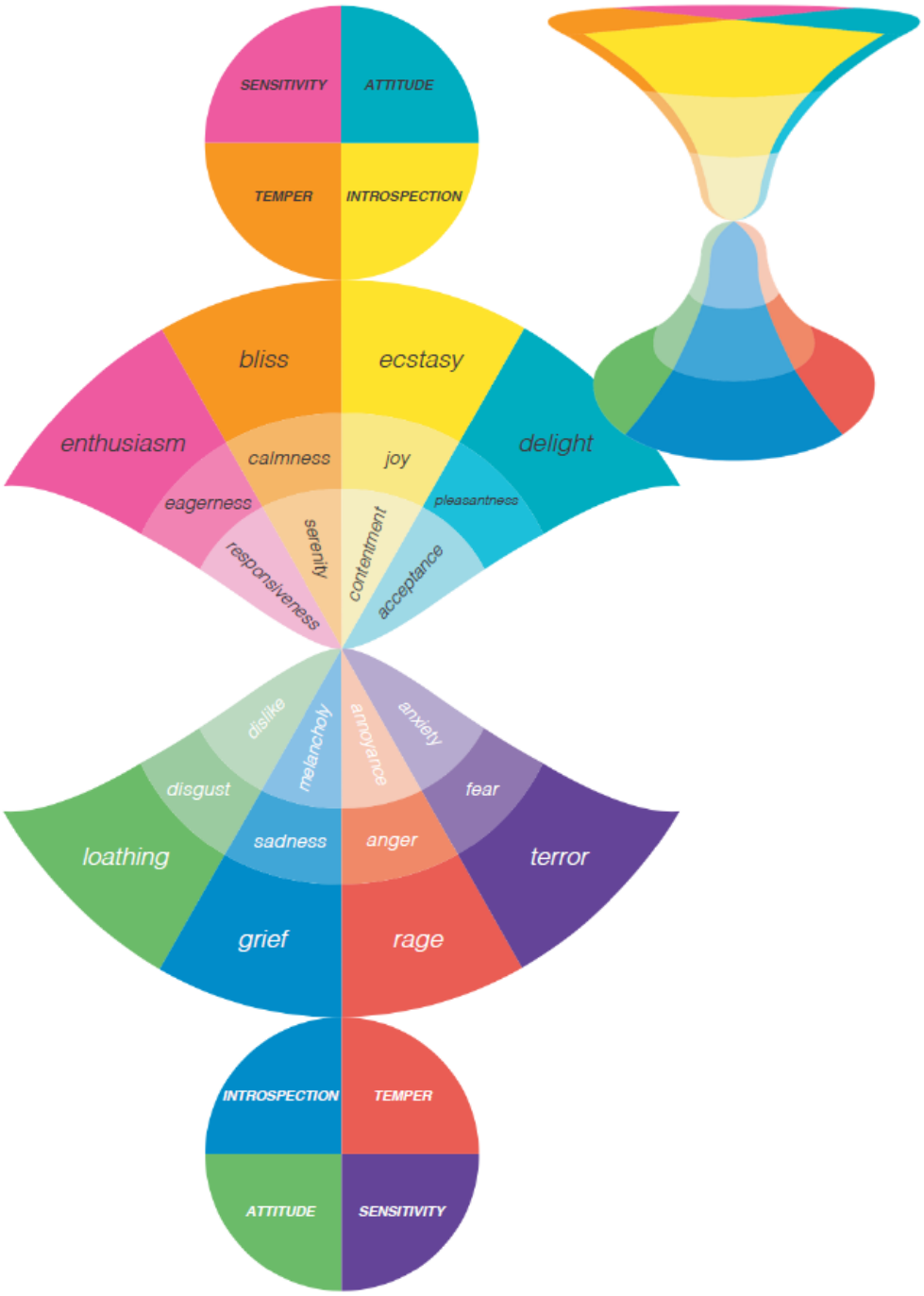


Figure 2.8: All 24 categories of the Hourglass model, diagram taken from [37]

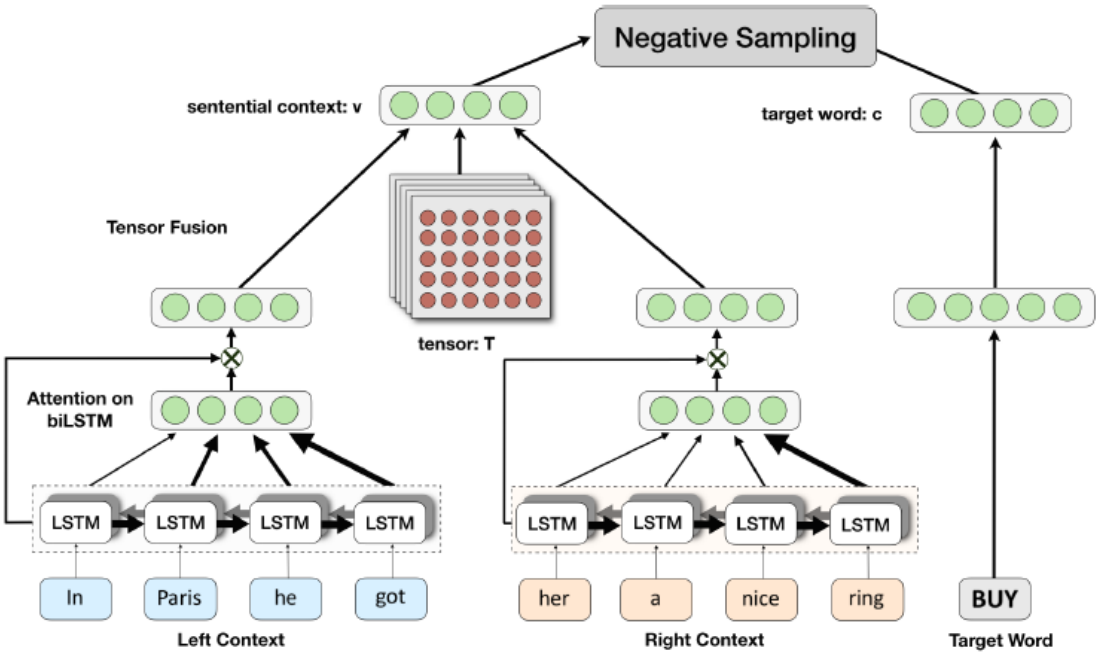


Figure 2.9: Visualization of the conceptual primitives discovery, image taken from [11]

verbs that modify such states (e.g. BREAK and FIX) are used as a basis to perform affective reasoning at a primitive level. The clusters discovered by the Deep Learning approach described in subsection 2.3.9 are utilized to link such primitives to their lexical substitutes. Once one of the polarity bearing states is assigned with a particular polarity, automatically all its lexical substitutes are assigned with the same polarity as well as its opposite state and its lexical substitutes with the respective opposite polarity. It also defines how verbs related to the state and their lexical substitutes can change it (e.g. BREAK changes INTACT into its opposite state).

2.3.9 Conceptual Primitives Discovery

In order to extract conceptual primitives for Sentiment Analysis in automatic fashion, as a first step each word is represented by its vector from the pre-trained 300-dimensional Word2vec model. By doing this any particular target concept C as well as its context sentences to the left L and to the right R can now be represented by sequences of vectors in the form of matrices. Then, a biLSTM model is used to extract the contextual features H_{LC} and H_{RC} in matrix form. In addition, an attention model is used to provide weights corresponding to the relevance of the underlying context across the sentence. Finally, the two sentential context representations are fused using a neural tensor to perform a bilinear fusion across both matrices and extract a sentential context vector v. The feature vector c for the target word is generated by passing C through a neural network [11]. A visualization of this process can be seen in Figure 2.9.

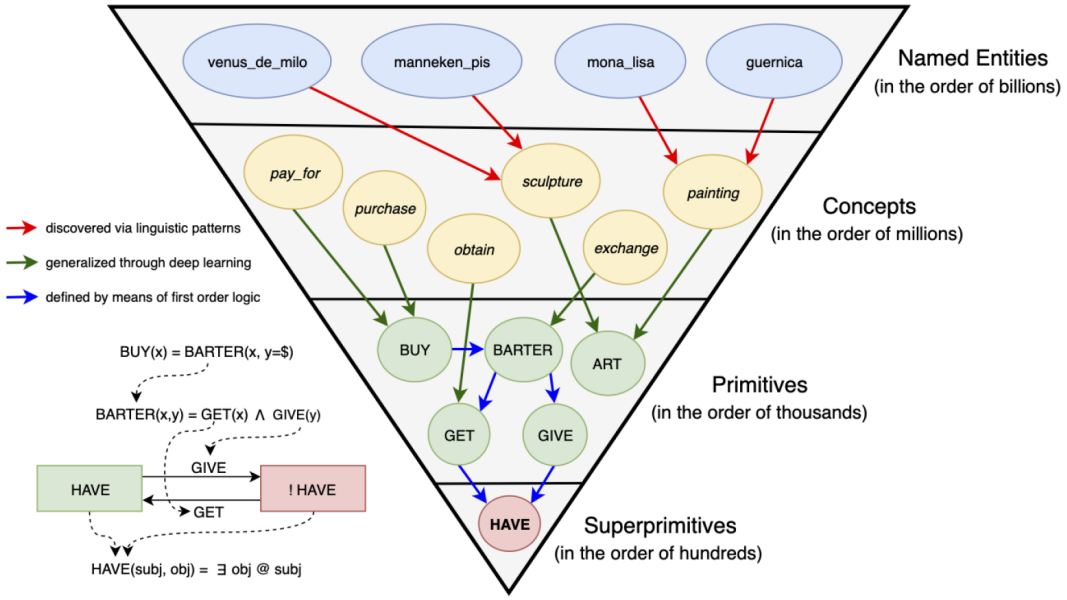


Figure 2.10: Graph structure of the dependencies in SenticNet 6, image taken from [11]

2.3.10 Context Embedding using BERT

Following this, the BERT [16] architecture is used to generate the sentential context embedding of a word. First, the pre-trained BERT network is fine-tuned on the ukWaC corpus [3]. Then the embedding for the context vector v is calculated by removing the target word c from the sentence and feeding the remainder of the sentence to the BERT architecture. The embedding for the target word c is obtained by simply feeding the word to the BERT pre-trained network. All potential replacement candidates b for the target word c in the embedding hyperspace are then ranked utilizing the similarity measure given in equation (2.13). $BERT(v, b)$ denotes the BERT-produced embedding of the sentence formed by replacing word c with the candidate word b in the sentence.

$$dist(b, (c, v) = cos(b, c) + cos(b, v) + cos(BERT(v, b), BERT(v, c)) \tag{2.13}$$

The automatic extraction of conceptual primitives is then completed by calculating these candidates and their respective similarity for every concept of the form verb-noun or adjective-noun in ConceptNet 5 [35]. To do this one word from the concept is taken as the target word and an example sentence for each concept serves as the context.

2.3.11 Superprimitives

Using symbolic logic the extracted conceptual primitives are manually deconstructed into so called superprimitives, which are assigned with emotion labels and a polarity value based on AffectiveSpace and the Hourglass Model. A visualitation displaying the dependencies in SenticNet can be seen in Figure 2.10.

3 Related Work

In literature, there is already quite a variety of existing work in the Sentiment Analysis field. In this thesis specifically, we deployed Sentiment Analysis to the e-learning domain leveraging SenticNet and Word Embedding resources. There has been preceding work on this topic.

3.1 Sentiment Analysis in E-learning platforms

The e-learning domain is a particularly interesting field of study for Sentiment Analysis, because of a range of potential benefits that come with a better understanding of e-learning systems. For instance, by analyzing students emotions, it might be possible to increase the students' motivation and improve learning processes in general. The study of sentiments in an e-learning platform can also lead to a better learning and teaching evaluation, help to assess the efficiency of e-learning tools and improve learning content recommendation systems [30].

Therefore, a variety of research projects already used Sentiment Analysis to study different aspects of e-learning systems. Clarizia et al. [13] used Sentiment Analysis tools to analyze students interactions in collaborative tools while still guaranteeing the students' communication privacy.

To the end of teachers' assessment Esparza et al. [18] adopted a Support Vector Machine in order to evaluate the teachers performance that was reviewed in comments of systems engineering students.

There has also been work on building an adaptive learning environment with improved recommendations based on Sentiment Analysis. Rodriguez et al. [31] described how to choose a fitting learning activity for a student based on personal goals and an emotional profile.

The study of Sentiment Analysis tools and applications within the e-learning domain is still an open research area. This thesis aims to make a contribution by exploring different ways to combine SenticNet resources with Word Embeddings in order to discover sentiments in learners reviews.

3.2 Deep Learning for Sentiment Analysis

The first Deep Learning approaches were already studied in the 1990s, but, due to relatively low computational power at the time, scientific communities quickly lost interest [45]. However, with the rapid increase of computational power and availability of big data over the last years, Deep Learning approaches have become state-of-the-art solutions across various domains and Sentiment Analysis has been no exception.

Dos Santos et al. [17] designed a Convolutional Neural Network (CNN) composed of two layers to capture features from character to sentence level.

Araque et al. [2] proposed an ensemble Deep Learning method, combining various sentiment classifiers trained on different sets of features. They tested their approach on six different data sets coming from Twitter and movies reviews.

Furthermore, Tang et al. [38] exploited a combination of CNN with a LSTM to learn continious representations of words for Sentiment Analysis. They managed to assign fixed-length vectors to sentences with varying lengths, showing how Deep Learning approaches can outperform common ML algorithms.

While the use of Deep Learning models has lead to amazing improvements in many domains, they have not been extensively studied for applications in e-learning yet.

3.3 Word Embeddings for Sentiment Analysis

Word Embeddings have been put to use with success in various domains. Within Sentiment Analysis they have been widely employed to improve the accuracy of baselines methods. Since traditional methods to create Word Embeddings usually do not incorporate word distributions for specific tasks, resulting representations might lack important information for a given task.

Tang et al. [40] joined context semantics and sentiment characteristics for the generation of Word Embeddings. The intention behind this is that vectors that are close to each other in the embedding model have similar meaning, therefore they might have a similar sentiment. However, in traditional embedding models usually many words with a similar context are mapped to similar vector representations despite having an opposite sentiment polarity (e.g. good and bad).

Focusing specifically on sentiment classification for postings on Twitter, Tang et al. [39] also trained sentiment-sensitive Word Embeddings. Their approach utilized three neural networks designed to detect the sentiment polarity of texts and successfully encoded sentiment information in the continuous representation of words. On a benchmark Twitter classification dataset in SemEval 2013 their methods outperformed the competitors.

Rudkowsky et al. [33] proposed a procedure on Word Embeddings in order to estimate different levels of negativity in a dataset of 56,000 plenary speeches from the Austrian parliament. They discovered that the different levels of negativity shown by speakers match with expected patterns indicated by common sense hypotheses.

Dessi et al. [15] used context-trained Word Embeddings to feed a neural network. This approach has shown to yield good results in a polarity detection task where the feedback of learners after attending a course has been analyzed.

3.4 SenticNet for Sentiment Analysis

As a semantic resource SenticNet is a powerful tool which can be utilized for a variety of NLP tasks. Its authors proposed and explored a wide range of applications for SenticNet such as integrating it into a two-channel Deep Learning based Sentiment Analysis model,

combining CNN and LSTM branches in a parallel manner [23] or using a set of heuristic linguistic patterns, which leverage on SenticNet to enhance the performance of an aspect extraction method [42].

On the other hand, SenticNet resources and applications have not been comprehensively evaluated by the scientific community. The reader notices that the works described above do not consider to use concepts in their Sentiment Analysis models. Therefore, the work of this thesis makes a contribution in this direction by studying models that use both Word Embeddings and concept vector representations.

4 The Proposed Approach

The use case of this thesis is to predict a rating for short user-generated comments. All the proposed models are trained on a dataset from the e-learning domain (comments about Udemy [2] online courses). However, the main goal for this task is to explore how SenticNet resources and Word Embeddings can be combined effectively for Sentiment Analysis tasks in general, thus it can easily be applied in other scenarios such as reviews of Amazon products. While the user-generated comments could be from any domain, every comment used to train, test and validate the suggested ML models, needs to have an assigned rating (i.e. a user rating).

From a mathematical point of view it is a regression task, the models presented in this chapter intend to predict a rating for each comment that is as close to the actual user rating as possible.

The following sections discuss three different proposed models to accomplish this task that are deployed and analyzed in this thesis.

4.1 Concept Vector Model

The basic idea for this approach is to build a 2-branch model where each branch processes a different input feature (Word Embeddings on one side, a SenticNet feature on the other). Subsequently, merging these two branches together should allow the model to then make a prediction with more information available. Each branch starts off with an input representation of the phrase to analyze. On one side, every word in a sentence gets replaced by a corresponding word vector, so the input comment becomes a sequence of real value vectors (Word Vector Module). On the other side, a parser searches the comment for concepts from a SenticNet database and replaces the found concepts with their AffectiveSpace vector representation, so the comment representation becomes a sequence of Concept vectors (Concpet Vector Module). Each branch then gets processed by a BiLSTM module. Subsequently, the outputs of both BiLSTM modules are merged together by concatenating them (Concatenation Module), which then serves as the final feature vector for the model to predict the rating (Regression/Output Module). The following parts of this section describe the components used by the Concept Vector Model displayed in Figure 4.1 in more detail.

4.1.1 Word Vector Module

After the input sentence(s) get transformed into a sequence of tokens in a preprocessing step, the Word Vector Module checks for each token if there is a word vector available in

[2] https://www.udemy.com/

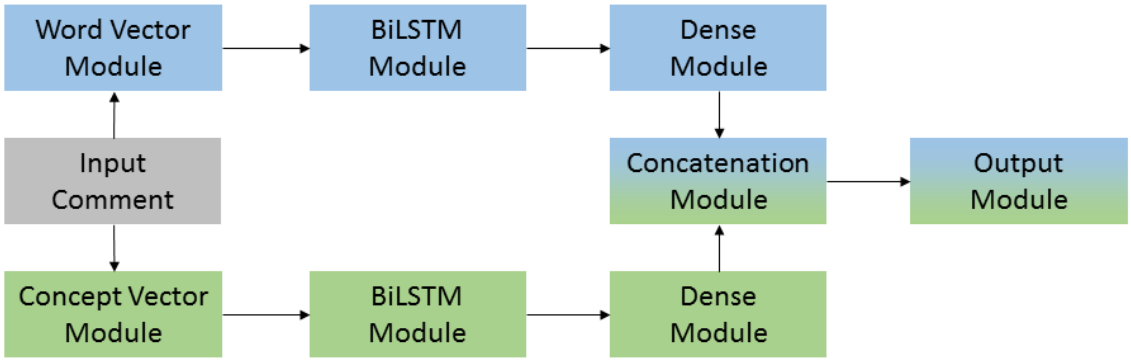


Figure 4.1: Structure of the proposed Concept Vector Model

the chosen Word Embedding database. If a vector representation is found the word gets replaced with its respective vector form and with a zero vector otherwise. This procedure is very common for ML approaches in NLP tasks, since it transforms a given input in natural language into a form that can be processed by neural networks.

4.1.2 Concept Vector Module

In a similar fashion to the Word Vector Module this module transforms a sequence of tokens into a sequence of real value concept vectors. An input comment is parsed first to detect used concepts. Since concepts can be multi-word expressions and distributed throughout the sentence this is not a trivial task. For the approach used in this thesis a parser was deployed which only detects multi-word concepts where every single one of its words directly follows each other in the input comment. Once the list of found concepts is obtained, each of them is replaced by its vector representation in AffectiveSpace, publicly available on the SenticNet Website. The sequence of real value vectors can then be processed in the same way as the word vectors in the other branch. It should be noted that this transformation step might already discard quite a bit of information contained in the original comment depending on how many concepts are found in it. Since SenticNet is not built on the target domain, it might not cover the whole set of concepts used in e-learning reviews. Additionally, depending on the parsers quality not all used concepts will be detected.

4.1.3 BiLSTM Module

BiLSTM modules have proven to be very effective at calculating a feature vector for NLP tasks in general since previous or subsequent words can completely change the meaning of a sentence and BiLSTM cells can consider a word's context. Therefore, after experimentation with different neural networks structures (e.g. a combination of a flatten layer and dense layers), which have shown to be less effective, it has also become the processing unit of choice for the models in this thesis. While the BiLSTM Module in the first branch (depicted in blue) takes a sequence of word vectors as an input, the BiLSTM

Module in the second branch (depicted in green) takes a sequence of concept vectors instead.

4.1.4 Dense Module

The purpose of this additional fully connected Dense layer in both branches before the concatenation enables the model to learn non-linear combinations of the two branches. In particular this module allows the model to learn a complex function to combine the two feature vectors produced by the BiLSTM modules effectively.

4.1.5 Concatenation Module

In this module the two branches are merged by concatenating the output of their respective Deep Learning modules to one single output vector. The two branches of the model carry a similar amount of information and have almost symmetrical structure, they just use different vector representations. Additionally, the concept vector branch might discard less relevant information right from the start by only searching for concepts. Therefore, a combination of the two branches is chosen which gives equal weight to the results of both and allows the model to learn to rely on either one. Concatenation is a simple method to achieve this.

4.1.6 Regression/Output Module

The Output Module consists of one fully connected dense layer which takes the output of the Concatenation Module and produces a single floating-point number. The output can then be compared to the actual user rating to calculate the used loss function.

4.2 Polarity Vector Model

The second proposed model also uses a 2-branch architecture, but utilizes a different representation of SenticNet than the AffectiveSpace vectors mentioned in subsection 4.1.2. While the branch tailored to use Word Embeddings of the model stays the same as in subsection 4.1.1, the second branch works differently. Instead of assigning each concept found by the parser with a vector representation, the Polarity Vector Module looks up available information on these concepts in the publicly available SenticNet python package [3]. For each concept the package contains a polarity label and value, and a list of related moodtags as well as a list of semantically related words. For this proposed model just the polarity values were used. So here, instead of a sequence of Concept Vectors, the Polarity Vector Module transforms an input comment into a sequence of real value polarity scores for each individual concept detected by the parser. This polarity vector also passes a BiLSTM layer and a fully connected dense layer before being merged with the models word vector branch by concatenation. The architecture of this model is depicted in Figure 4.2. The

[3] https://pypi.org/project/senticnet/

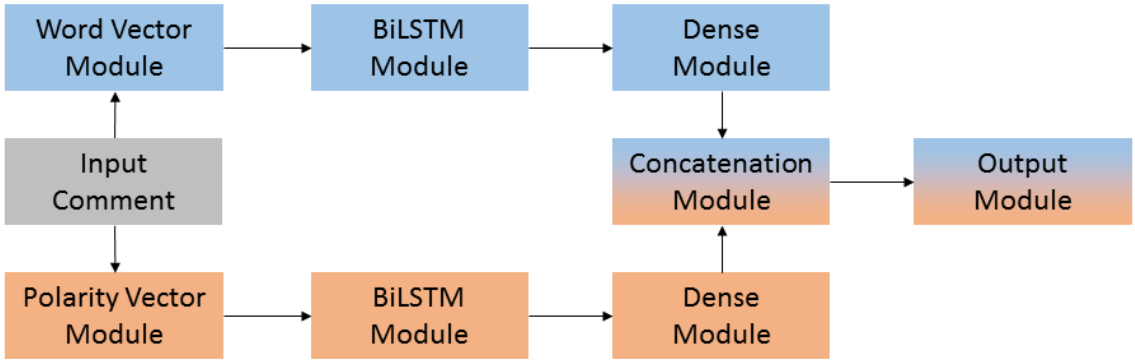


Figure 4.2: Structure of the proposed Polarity Vector Model

changed components compared to the previously described Concept Vector Model are discussed below.

4.2.1 Polarity Vector Module

Like in subsection 4.1.2 the input comment needs to be parsed for finding concepts first in order to calculate their polarity. The SenticNet python package can be used as a python dictionary to both get a comprehensive list of 200k concepts the parser should search for, as well as access the particular concepts polarity. Keeping the order of how the concepts appear in the input comment, the polarity scores of all found concepts are subsequently put together into a sequence vector of real values, which serves as the input of the BiLSTM Module in this branch.

4.2.2 BiLSTM Module

The input of the BiLSTM Module in the second branch (depcited in orange) now is a sequence of real values, the polarity scores for each detected concept. Otherwise the module remains unchanged, see subsection 4.1.3.

4.2.3 Unchanged Modules

This model's architecture is similar to the proposed Concept Vector Model. To make a fair comparison with the results obtained by every model, its modules are not modified if not necessary. Modules that remained unchanged are listed here: Word Vector Module (see subsection 4.1.1), Dense Module (see subsection 4.1.4), Concatenation Module (see subsection 4.1.5)

4.3 Polarity Score Model

The third proposed model of this thesis stirs away from the previous idea of an almost symmetric 2-branch model which utilizes different vector representations in each branch.

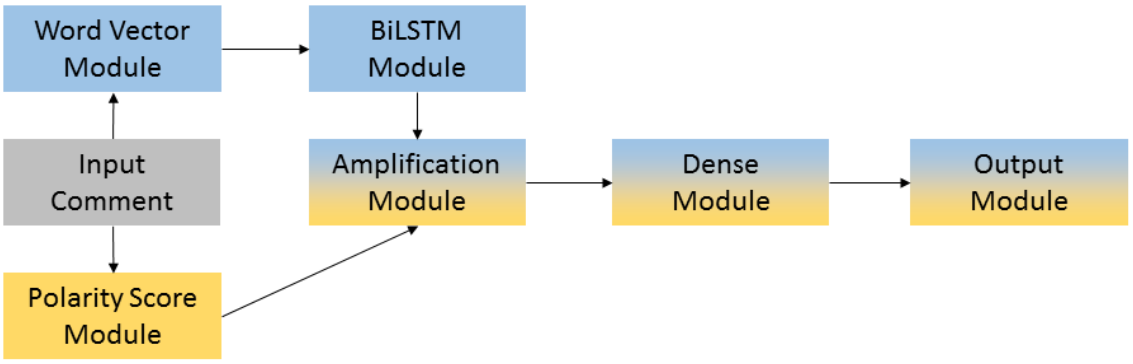


Figure 4.3: Structure of the proposed Polarity Score Model

In fact, the Polarity Score Module calculates a single real value polarity score for each comment by simply adding up the values found for each concept. This polarity score is then used to amplify the output of the word vector branch by multiplication. The idea here is to stress the tendency in either positive or negative direction found in a comment by amplifying the real value numbers in the feature vector produced by the word vector branch. The architecture of this model is shown in Figure 4.3. The new components compared to the previous models are discussed below.

4.3.1 Polarity Score Module

This module does not keep all the polarity scores for each discovered concept by the parser in a vector as in section 4.3, instead they are simply summed up in this module. While this does not consider the semantic structure of the sentence, the total polarity score of all discovered concepts should still be a good indicator of whether the expressed sentiments are rather positive or negative and provide a simple measurement of their intensity.

4.3.2 Amplification Module

Since the Polarity Score Module is just adding up values disregarding whether there are both positive and negative values in a comment, it is not sufficient to use only this polarity score to predict the rating of a comment. It can however be used to change the feature vector produced by the word vector branch. Figuratively speaking the information gathered by the Polarity Score Module can be used to suggest a more bold prediction for the Output Module to either a very low or high rating.

4.3.3 Word Vector Module

This module remains unchanged, see subsection 4.1.1.

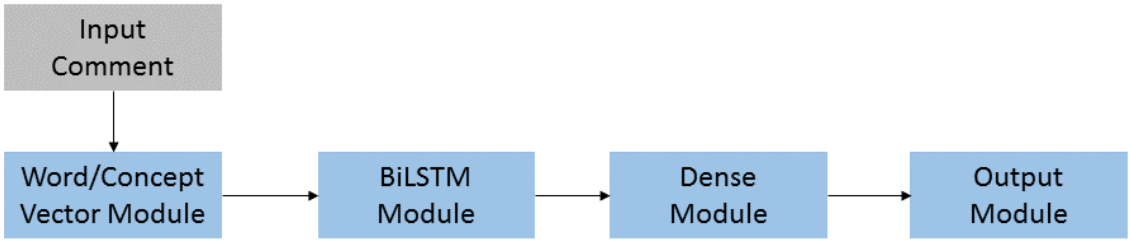


Figure 4.4: Structure of the single representation model

4.3.4 BiLSTM Module

While there is no BiLSTM module anymore in the second branch (depicted in yellow), the BiLSTM Module in the first branch remains unchanged, see subsection 4.1.3.

4.3.5 Dense Module

This model also uses a Dense Module following the Amplification Module. Its purpose is to both reduce the dimension of the feature vector as well as allowing the model to adjust the amplified feature vector to a particularly suited form before the Output Module produces the models prediction.

4.4 Single Representation Models

In addition to the three proposed models, models that only used one vector representation for each comment are also analyzed for comparison. These models either use one of the word vector representations or the concept vector representation as input and are all followed by a BiLSTM, a Dense and an Output module. The model architecture for these single representation models was proposed by Dessì et al. [15] and can be seen in Figure 4.4.

5 Evaluation

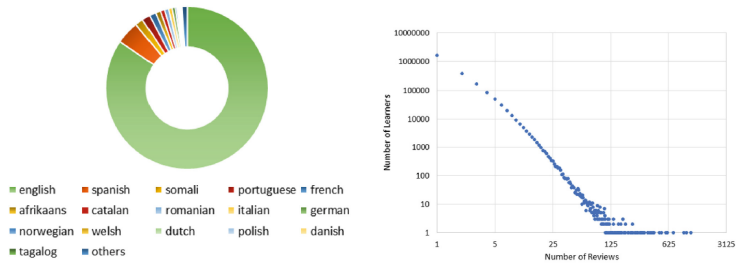
In this chapter, first, the used dataset and training parameters are introduced. Subsequently, the results are presented and discussed.

5.1 Data Analysis and Preprocessing

As already mentioned in chapter 4 the case of study chosen to analyze the presented models is a set of learners' reviews for online courses collected by a crawler [14] from Udemy. Udemy is an American massive open online course provider founded in 2010 and one of the leading global marketplaces for online learning and teaching. In the dataset only learners with at least one rating are included, each course can have zero or more ratings. In Figure 5.1 the distribution of learners per review language (a) and per number of provided ratings (b) are depicted.

From the 4,530,505 reviews in total only 1,266,313 included a text message in their review and could be used for the purpose of this thesis. Since the data was collected automatically and the comments were user-generated, the dataset contains quite a bit of noise, e.g. spelling mistakes. Initially, the rate with which a corresponding word vector representation in the embedding models was found for a unique token from the vocabulary was low. Depending on the deployed Word Embedding GloVe (GlV), Word2vec (W2V) or FastText (FaT) the accuracy varied between 50% and 60%. To improve this rate a spellchecker [4] was applied to all the comments in order to correct spelling mistakes and get a better vector representation of the affected comments. While this cleaning step improved the rate by about 10%, the original intended meaning of some comments in

[4]https://pypi.org/project/pyspellchecker/



(a) (b)

Figure 5.1: The distribution of learners per (a) review language and (b) number of reviews. (Image taken from [14])

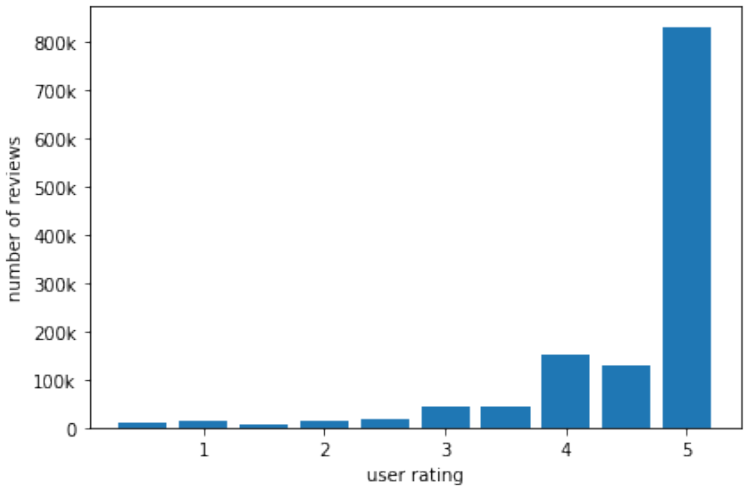


Figure 5.2: Original distribution of all user ratings for the whole Udemy dataset.

the dataset could not be restored, i.e. because some of them were written in a different language than English.

Additionally, on the rating scale from 0.5 to 5 (step-size 0.5), the vast majority of user ratings were either either 4, 4.5 or 5. Therefore, the original distribution displayed in Figure 5.2 was evened out to avoid possible biases during the training of the model. Otherwise, all the proposed models would default in their prediction to a very high user rating in order to minimize the loss function. To achieve an even distribution over all 10 different valid user ratings, we determined the user rating with the lowest number of samples (5779 for the user rating 1.5) and randomly discarded comments and their respective rating for all other user ratings until their sample size was also at that number. In consequence all three models presented in the previous chapter were trained on a set of 57790 comments with evenly distributed user ratings.

5.2 Training Parameters

- **Dataset split:** The evened out dataset was split into 70% training, 10% validation and 20% test data.
- **Activation function:** Every dense layer deployed in the analyzed models used the ReLU activation function as nonlinearity.
- **Early Stopping:** The early stopping conditions used during training were a patience value of at least 3 epochs per training session and a minimum change in the validation loss of 0.01.
- **Loss function:** It is not clear whether MAE or MSE is the better suited loss function for this task. While MAE is a solid choice, MSE in contrast punishes the model more for samples for which the prediction is far off the actual user rating. Samples that lead to a high loss value can be both comments that do not provide a valid information

basis to make a good prediction as well as comments with just either a very high or low user rating. Therefore, using MSE stresses the noisy samples in the dataset, but it also incentivizes the model to stray away from defaulting to a 'middleground' average rating. Since there are both potential benefits and downsides, we decided to train every model on both loss functions for comparison.

- **Word vector dimensions** For the deployed Word Embeddings we used the highest dimensional word vectors available which were 300-dimensional vectors for Word2vec [5], FastText [6] and GloVe [7].

- **Unit numbers** Every dense layer in the analyzed models consisted of 10 units (the amount of different valid user ratings), all BiLSTM layers had a size of 64 units.

5.3 Results

Table 5.1 shows the testing results of all 13 different examined model types for the two different loss functions used during training. Every model was trained using MAE as loss function and MSE for comparison. The first three columns contain the results for the case where MAE was used as the loss function. Listed are the MAE and MSE loss over the whole test set after training finished and the epoch number where each model first reached the early stopping condition. The remaining three columns show the results for the case where MSE was used as the loss function instead.

5.4 Discussion

It is remarkable, how well the concept vector representation performed on its own. The model only using a sequence of AffectiveSpace vectors as an input had one of the best results for MAE loss and the best out of all models for MSE loss. The training graph depicted in Figure 5.3 also indicates good generalization. The other single representation models had similar results and number of training epochs.

Compared to how well the single vector representation models on their own did, the Concept Vector Model and the Polarity Vector Model both struggled to pick up on the additional information to make better predictions. The representative training graph for the model ConceptVector(FaT) in Figure 5.4 shows validation loss quickly reaches a plateau. While training loss still decreases quickly the additional learned information did not generalize and improve the models loss on the test set. The Polarity Vector Model performed slightly better and improved the total loss values on the test set compared to just using a word vector representation by a small margin.

The Polarity Score Model performed the best out of all different models on MAE loss and second best on MSE loss. The feature vector amplification has shown to be a clear

[5] https://s3.amazonaws.com/dl4j-distribution/GoogleNews-vectors-negative300.bin.gz
[6] https://dl.fbaipublicfiles.com/fasttext/vectors-english/crawl-300d-2M.vec.zip
[7] https://nlp.stanford.edu/projects/glove/

Table 5.1: This table reports the testing results for every model that was trained with either MAE or MSE loss function.

Model	MAE loss function			MSE loss function		
	MAE	MSE	Epochs	MAE	MSE	Epochs
Word2vec	1.517	3.413	8	1.512	3.391	10
GloVe	1.505	3.365	7	1.474	3.211	7
FastText	1.587	3.766	7	1.526	3.458	8
AffectiveSpace	1.483	3.257	15	1.394	2.817	18
Concept Vector (W2V)	1.558	3.625	10	1.476	3.222	7
Concept Vector (GlV)	1.595	3.808	8	1.498	3.328	7
Concept Vector (FaT)	1.557	3.620	8	1.554	3.607	6
Polarity Vector (W2V)	1.511	3.392	8	1.500	3.337	12
Polarity Vector (GlV)	1.489	3.284	8	1.513	3.402	11
Polarity Vector (FaT)	1.554	3.608	9	1.489	3.283	8
Polarity Score (W2V)	**1.454**	**3.145**	11	1.468	3.217	18
Polarity Score (GlV)	1.483	3.293	12	1.462	**3.173**	16
Polarity Score (FaT)	1.541	3.567	11	**1.455**	3.175	13

Figure 5.3: Training graph for the AffectiveSpace model using MAE loss

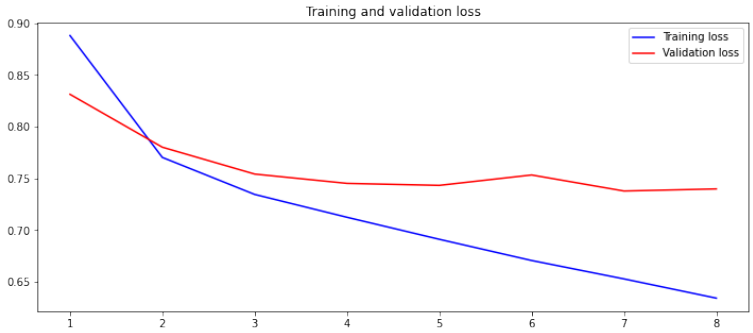


Figure 5.4: Training graph for the ConceptVector(FaT) model using MAE loss

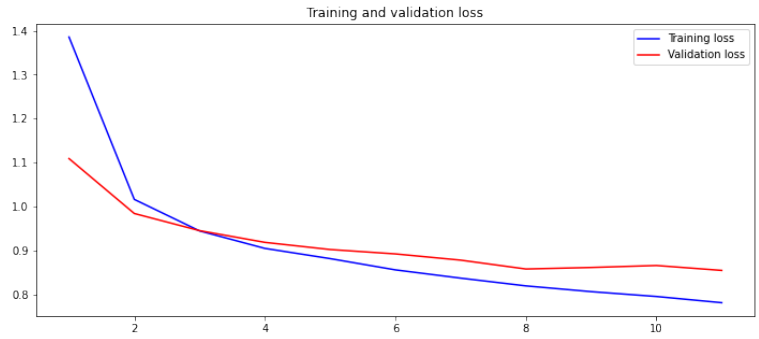


Figure 5.5: Training graph for the PolarityScore(W2V) model using MAE loss

improvement compared to only using a word vector input. Its training graph for using Word2vec word vectors and MAE loss is shown in Figure 5.5.

5.5 Takeaways

The ability to analyze learners reviews and automatically detect the expressed sentiment in them offers multiple benefits. On the one hand it enables online course providers to better understand their customers feedback and improve their service accordingly, on the other hand understanding the sentiment in learner reviews for a specific course can lead to a better decision for other learners whether they want to attend it. The proposed approached can be integrated in recommender systems which play an important role in the e-learning domain as they assist the learners to find a fitting course that meets their learning needs.

On a higher abstraction level, the examined approach of combining word vector representations with SenticNet resources for Sentiment Analysis is interesting, because it aims to combine a statistical (bottom-up) approach with a conceptual (top-down) one. While each of these approaches has already shown some success individually, they should be able to complement each other, since they focus on discovering different features, and therefore improve ML models performances in Sentiment Analysis tasks. The work of this thesis provides a step forward in this direction. The results indicate that concept vector representations of comments are a potent resource on their own for Sentiment Analysis and that SenticNet information can be used succesfully to improve the single representation models performance.

6 Conclusion

In this thesis different methods to combine SenticNet resources with Word Embeddings for Sentiment Analysis are explored. Three proposed model architectures are presented and their performance in a prediction task on the Udemy dataset of online reviews is analyzed. While utilizing an AffectiveSpace vector representation of each comment has shown to perform well on the given task, combining it with a Word Embedding representation on a second branch did not lead to an improvement (Concept Vector Model). Using SenticNet resources to extract polarity values, collect them in vector form and merging it with the feature vector, calculated by a word vector branch, did slightly increase the models performance compared to just using word vectors (Polarity Vector Model). However, clearly the best results generated a model architecture which uses SenticNet resources in order to calculate a single polarity score for each comment and then amplifies the feature vector produced by the word vector branch with it.

While we have found first successes to improve Deep Learning models performances on Sentiment Analysis tasks by feeding them both word vector representations as well as SenticNet resources, there are several promising ways to directly continue this work. A more sophisticated semantic parser to discover SenticNet concepts in natural language should improve the results for all three models discussed in this thesis. Rewarding could also be to experiment with alternative ways to calculate the polarity score of a comment instead of summarizing. Since the Udemy dataset had to be trimmed down to a small portion in order to even out the distribution, a larger training corpus could lead to better generalization. Furthermore, interesting to explore are also other manipulation methods than amplification used on the feature vector produced by the word vector branch.

Although research on Sentiment Analysis in general has produced a variety of solid methods, it still poses some interesting challenges that also require further investigation:

1. **Availability of datasets:** Most Sentiment Analysis studies in the e-learning domain still use small datasets which are not publicly available. In order to better evaluate and compare model performances a large collection of diverse and cleaned datasets should be established.

2. **Overloaded word vectors:** Word Embeddings created by traditional methods assign only one vector to each word in the vocabulary. While words can have multiple different meanings (e.g., 'bear' for the animal and the verb), they will only have one vector representation mixing up all its different use cases. Thus, a promising feature direction is the exploration of BERT embeddings.

3. **Data and algorithm biases:** With the rapid increase of ML methods for Sentiment Analysis, addressing biases will also become more and more important. Biases can for example be introduced through the use of training data which is not an accurate

sample of the target population. Furthermore, the methods used to collect or measure data and the algorithms leveraged for predicting sentiments can propagate existing biases. Future research should therefore control these biases in the developed models as much as possible in order to promote fair, transparent, and accountable systems.

4. **Multi-aspect Modeling:** Sentiment Analysis in the e-learning domain has been concentrated on determining either the polarity (e.g., positive or negative) or the sentiment rating (e.g., on a scale from one to five stars) of a review. It is not sufficient to only estimate overall ratings in order to represent the multiple potential aspects on which an educational element can be reviewed (e.g., the course content, the instructor, and the platform). To gather insightful knowledge on how people perceive each of them, aspect-level Sentiment Analysis is needed.

We expect that the contribution in this thesis will be a step towards bridging the gaps between concept-level Sentiment Analysis and statistical approaches.

An implementation of all examined models can be found on a Github repository [8].

[8] https://github.com/FelixBoltz/COLESEAN

Bibliography

[1] Zahra Abbasi-Moud, Hamed Vahdat-Nejad, and Javad Sadri. "Tourism recommendation system based on semantic clustering and sentiment analysis". In: *Expert Systems with Applications* 167 (2021), p. 114324.

[2] Oscar Araque et al. "Enhancing deep learning sentiment analysis with ensemble techniques in social applications". In: *Expert Systems with Applications* 77 (2017), pp. 236–246.

[3] Marco Baroni et al. "The WaCky wide web: a collection of very large linguistically processed web-crawled corpora". In: *Language resources and evaluation* 43.3 (2009), pp. 209–226.

[4] Piotr Bojanowski et al. "Enriching word vectors with subword information". In: *Transactions of the Association for Computational Linguistics* 5 (2017), pp. 135–146.

[5] Erik Cambria et al. "AffectiveSpace 2: Enabling affective intuition for concept-level sentiment analysis". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 29. 1. 2015.

[6] Erik Cambria et al. "Affectivespace: Blending common sense and affective knowledge to perform emotive reasoning". In: *WOMSA at CAEPIA, Seville* (2009), pp. 32–41.

[7] Erik Cambria et al. "GECKA: game engine for commonsense knowledge acquisition". In: *The Twenty-Eighth International Flairs Conference*. 2015.

[8] Erik Cambria et al. "Sentic medoids: Organizing affective common sense knowledge in a multi-dimensional vector space". In: *International Symposium on Neural Networks*. Springer. 2011, pp. 601–610.

[9] Erik Cambria et al. "SenticNet 4: A semantic resource for sentiment analysis based on conceptual primitives". In: *Proceedings of COLING 2016, the 26th international conference on computational linguistics: Technical papers*. 2016, pp. 2666–2677.

[10] Erik Cambria et al. "SenticNet 5: Discovering conceptual primitives for sentiment analysis by means of context embeddings". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 32. 1. 2018.

[11] Erik Cambria et al. "SenticNet 6: Ensemble application of symbolic and subsymbolic AI for sentiment analysis". In: *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*. 2020, pp. 105–114.

[12] Erik Cambria et al. "Senticnet: A publicly available semantic resource for opinion mining." In: *AAAI fall symposium: commonsense knowledge*. Vol. 10. 0. Citeseer. 2010.

[13] Fabio Clarizia et al. “E-learning and sentiment analysis: a case study”. In: *Proceedings of the 6th International Conference on Information and Education Technology*. 2018, pp. 111–118.

[14] Danilo Dessı et al. “Coco: Semantic-enriched collection of online courses at scale with experimental use cases”. In: *World Conference on Information Systems and Technologies*. Springer. 2018, pp. 1386–1396.

[15] Danilo Dessı et al. “Evaluating neural word embeddings created from online course reviews for sentiment analysis”. In: *Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing*. 2019, pp. 2124–2127.

[16] Jacob Devlin et al. “Bert: Pre-training of deep bidirectional transformers for language understanding”. In: *arXiv preprint arXiv:1810.04805* (2018).

[17] Cicero Dos Santos and Maira Gatti. “Deep convolutional neural networks for sentiment analysis of short texts”. In: *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*. 2014, pp. 69–78.

[18] Guadalupe Gutiérrez Esparza et al. “A sentiment analysis model to analyze students reviews of teacher performance using support vector machines”. In: *International Symposium on Distributed Computing and Artificial Intelligence*. Springer. 2017, pp. 157–164.

[19] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. `http://www.deeplearningbook.org`. MIT Press, 2016.

[20] Catherine Havasi et al. “Open mind common sense: Crowd-sourcing for common sense”. In: *Proceedings of the 2Nd AAAI Conference on Collaboratively-Built Knowledge Sources and Artificial Intelligence*. 2010, pp. 53–53.

[21] Sepp Hochreiter and Jürgen Schmidhuber. “Long short-term memory”. In: *Neural computation* 9.8 (1997), pp. 1735–1780.

[22] Hui Li et al. “An intelligent movie recommendation system through group-level sentiment analysis in microblogs”. In: *Neurocomputing* 210 (2016), pp. 164–173.

[23] Wei Li et al. “User reviews: Sentiment analysis using lexicon integrated two-channel CNN–LSTM family models”. In: *Applied Soft Computing* 94 (2020), p. 106435.

[24] Diana Maynard and Adam Funk. “Automatic detection of political opinions in tweets”. In: *Extended Semantic Web Conference*. Springer. 2011, pp. 88–99.

[25] Walaa Medhat, Ahmed Hassan, and Hoda Korashy. “Sentiment analysis algorithms and applications: A survey”. In: *Ain Shams engineering journal* 5.4 (2014), pp. 1093–1113.

[26] Tomas Mikolov et al. “Distributed representations of words and phrases and their compositionality”. In: *arXiv preprint arXiv:1310.4546* (2013).

[27] Tomas Mikolov et al. “Efficient estimation of word representations in vector space”. In: *arXiv preprint arXiv:1301.3781* (2013).

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

[29] Ruggero Petrolito and Felice Dell’Orletta. “Word Embeddings in Sentiment Analysis.” In: *CLiC-it*. 2018.

[30] Marcos Wander Rodrigues, Seiji Isotani, and Luiz Enrique Zarate. “Educational Data Mining: A review of evaluation process in the e-learning”. In: *Telematics and Informatics* 35.6 (2018), pp. 1701–1717.

[31] Pilar Rodriguez, Alvaro Ortigosa, and Rosa M Carro. “Extracting emotions from texts in e-learning environments”. In: *2012 Sixth International Conference on Complex, Intelligent, and Software Intensive Systems*. IEEE. 2012, pp. 887–892.

[32] Renata Lopes Rosa et al. “A knowledge-based recommendation system that includes sentiment analysis and deep learning”. In: *IEEE Transactions on Industrial Informatics* 15.4 (2018), pp. 2124–2135.

[33] Elena Rudkowsky et al. “More than bags of words: Sentiment analysis with word embeddings”. In: *Communication Methods and Measures* 12.2-3 (2018), pp. 140–157.

[34] Push Singh et al. “Open mind common sense: Knowledge acquisition from the general public”. In: *OTM Confederated International Conferences" On the Move to Meaningful Internet Systems"*. Springer. 2002, pp. 1223–1237.

[35] Robyn Speer, Joshua Chin, and Catherine Havasi. “Conceptnet 5.5: An open multilingual graph of general knowledge”. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 31. 1. 2017.

[36] Carlo Strapparava, Alessandro Valitutti, et al. “Wordnet affect: an affective extension of wordnet.” In: *Lrec*. Vol. 4. 1083-1086. Citeseer. 2004, p. 40.

[37] Yosephine Susanto et al. “The hourglass model revisited”. In: *IEEE Intelligent Systems* 35.5 (2020), pp. 96–102.

[38] Duyu Tang, Bing Qin, and Ting Liu. “Document modeling with gated recurrent neural network for sentiment classification”. In: *Proceedings of the 2015 conference on empirical methods in natural language processing*. 2015, pp. 1422–1432.

[39] Duyu Tang et al. “Learning sentiment-specific word embedding for twitter sentiment classification”. In: *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 2014, pp. 1555–1565.

[40] Duyu Tang et al. “Sentiment embeddings with applications to sentiment analysis”. In: *IEEE transactions on knowledge and data Engineering* 28.2 (2015), pp. 496–509.

[41] John K Tarus, Zhendong Niu, and Ghulam Mustafa. “Knowledge-based recommendation: a review of ontology-based recommender systems for e-learning”. In: *Artificial intelligence review* 50.1 (2018), pp. 21–48.

[42] Ana Valdivia et al. “What do people think about this monument? Understanding negative reviews via deep learning, clustering and descriptive rules”. In: *Journal of Ambient Intelligence and Humanized Computing* 11.1 (2020), pp. 39–52.

[43] Ashish Vaswani et al. “Attention is all you need”. In: *arXiv preprint arXiv:1706.03762* (2017).

[44] Yonghui Wu et al. “Google’s neural machine translation system: Bridging the gap between human and machine translation”. In: *arXiv preprint arXiv:1609.08144* (2016).

[45] Lei Zhang, Shuai Wang, and Bing Liu. “Deep learning for sentiment analysis: A survey”. In: *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery* 8.4 (2018), e1253.

[46] Yukun Zhu et al. “Aligning books and movies: Towards story-like visual explanations by watching movies and reading books”. In: *Proceedings of the IEEE international conference on computer vision*. 2015, pp. 19–27.