VYTAUTAS MAGNUS UNIVERSITY

FACULTY OF INFORMATICS

DEPARTMENT OF APPLIED INFORMATICS

Joel Kanku Ntumba

SENTIMENT ANALYSIS IN FRENCH FOR IMPROVING SERVICES AND PRODUCTS : A DEEP LEARNING APPROACH.

Master final thesis

Applied informatics study programme, state code 6211BX012

Study field Informatics

Supervisor ______________________ __________ ___________
(degree, name, surname) (signature) (date)

Defended prof.dr. Daiva Vitkutė-Adžgauskienė __________ __________
(Dean of Faculty) (signature) (date)

Kaunas, 2018

TABLE OF CONTENTS

List of Figures

List of Table

ABBREVIATION

CNN : Convolutional Neural Network

SVM:Support Vector Machines

NB:Naive Bayer

UPNs:Unsupervised Pre-trained Networks

RNN: Recurrent Neural Networks

RNN:Recursive Neural Networks

DBNs: Deep Belief Networks

GANs:Generative Adversarial Networks

RELU: Rectified Linear Unit

ANN: Artificial Neural Network

MLP: Multilayer Perceptron

LSTM: Long Short-Term Memory

DNN: Depp Neural Network

CPU: Central Processing Unit

GPU: Graphics Processing Unit

CBOW: Continuous Bag Of Words

Author	Joel Kanku Ntumba
Title	Sentimental analysis in French for improving services and products: a deep learning approach.
Supervisor	Daiva vitkute-adzgauskiene
Number of pages	49

ABSTRACT

Sentiment analysis is a process of evaluating an opinion expressed in a piece of text to determinate the writer's feeling whether it is positive, negative .or neutral. In this field of study, a lot of researchers have used machine learning approach or deep learning approach in their works to evaluate the sentiment expressed in a text. Most of them have used English words using and very few were interested in other languages. Taking this into account, we present a deep learning approach for sentimental analysis using French text. The approach is based on Convolutional Neural Networks for text classification. The goal of this paper is to use convolutional neural network architecture to classify the sentiment expressed in a given French text about services and products. The model build get a brief text of restaurant and laptop reviews as input and determines the sentiment polarity in each sentence as output. To prove the effectiveness of our approach, we have run experiments in two-channel i.e.CNN-rand and CNN-static. To determine the effectiveness of this approach for classifying French text, we conducted experiments in two-channels: CNN-rand and CNN-static. We obtained encouraging results for CNN-rand with 75.44% for restaurant domain and 74.64% for laptop domain. And with CNN-static we got 82.46% for restaurants and 81.16% for laptops.

Keyword: Sentimental Analysis, Convolutional Neural Network,Deep Learning ,Mxnet, Word Embbeding

1.INTRODUCTION

1.1.Context and Motivation

with the rapid increase of e-commerce websites, year by year the online customer's reviews number have dramatically increased. this has resulted in large sources of information to be analyzed by the organizations to understand their customer opinion about their products or services. this field of analyzing opinions is called sentiment analysis or opinion mining. many authors have defined the sentimental analysis but the definition by liu[1] is the most used in the research community. he defined it as follows: sentiment analysis is the field of study that analyzes peoples opinions sentiments evaluations appraisals attitudes and emotions towards entities such as products services organizations individuals issues events topics and their attributes[1]. to capture the opinion expressed by users in each product or service each review has an overall score. but it doesn't provide the complete information. for instance, similar products may get the same overall rating but with different, the aspect opinions, therefore, extracting the aspect in each review is so important and determine the associated sentiment with them give better precision and relevant information which can be used to ameliorate services for the futures consumers. previously researchers used generally two approached semantic orientation approach and machine learning approach to analyze the sentiment expressed in a text. among these two approaches, the machine learning has shown good in many works. but recently a new approach comes up known as deep learning. this approach got researchers attention due to the fact that it performed better than the machine learning methods [2 3]. whereas most of these experiments in machine learning methods and deep learning methods used text from English language and only a few experiments were in other languages. in this context we propose a deep learning approach for sentence classification using convolutional neural networks architecture on mxnet using text from the French language. this approach used a supervised learning method where we trained CNN-models on annotated data of french restaurants reviews and laptop reviews after applying an automatic-translation of data from English to French collected from semeval2015 task12. the main goal of our experimental is to confirm that the convolutional neural network is also suitable to train and classify the sentiment expressed in French sentences. we trained one-dimensional convolutional neural networks to accepts each input sentence based on given sentence and returns an output with the classification of sentiment polarity.

Our work is summarized as below:

- we propose a convolutional neural network for text classification on mxnet based on kims convolutional neural networks for sentence classification [4]. we simply train a CNN-rand with three different convolutional layers by keeping the word vectors random
- alternatively we also train CNN-static using a pre-trained word embedding.

1.2.Research Structure

The rest of work is organized as the following: Section 2. presents previous work done in area of sentimental analyses in two different perspectives: machine learning and deep learning. Section 3. The overview of machine learning and deep learning techniques are presented . Section 4. Describes a comparison of the deep learning framework and the deep learning framework used in our models. Our propose approached and workflow of our are presented in Section 5.Section 6. The dataset structure is presented followed by the model setup then model variant and software tools used to implement our system. the evaluation and results of our models are presents in section 7. Section 8. Presents a comparison of our results with SemEval2015 task 12(slot3) results. Finally , discuss about result and present our observation in section 9 then presents conclusions and future work in section 10.

2. RELATED WORK

In this section we present the relate work for sentimental analysis from two perceptive. Machine Learning and Deep learning.

2.1. Machine learning

Researchers in this approach have used different algorithms such as Support Vector Machines (SVM), Naive Bayes (NB), and others as a classifier for sentiment analysis. In this approach The datasets to be used need to be split into two different sets, one set for training and then another set for evaluation. The purpose of the training set is to learn from features of the domain and the purpose of evaluation set is to build a model from the training set. To determine the performance in this technique, the selected method for feature extraction plays a vital role. Some of the popular method used are n-grams (unigrams, bigrams, and trigrams) [5, 6], features based on POS tagging [7], a bag of words [8], TF-IDF [9] and features based on dependency rules [10]. [11].Table1 shows the Comparison of some sentiment analysis approaches with machine learning Approach in different languages. We found in the literature

Paper	Languages	Machine learning Techniques
Shi and Li[12]	English	NB, SVM
Boiy and Moens[13]	English	SVM, MNB, MaxEnt
Singh et al.[14]	English	NB, SVM
Habernal et al.[15]	Czech	SVM, MaxEnt
Tan and Zhang[16]	Chinese	SVM, NB, K-nearest neighbour classifier, Winnow classifier
Zagibalov and Carroll[17]	Chinese	SVM
Balahur and Turchi[18]	English, French, Italian, German, Spanish	Hybrid, SVM SMO
Zhu et al.[19]	Chinese	SVM
Mahyoub et al.[20]	Arabic	SVM
Al-Ayyoub et al.[21]	Arabic	SVM
Mizumoto et al.[22]	English	Bootstrapping
Ghorbel and Jacot[23]	French	SVM

Table 1.Comparison of multilingual sentiment analysis with machine learning approaches.

2.2. Deep Learning

“*Deep learning approaches are able to automatically capture, to some extent, the syntactic and semantic features from a text without feature engineering, which is labor intensive and time-consuming. They attract much research interest in recent years, and achieve state-of-the-art performances in many fields of NLP, including sentiment classification.”*[24]. For example, this paper [25] proposed an approach to sentiment analysis of short texts. Based on a convolutional neural network this approach is applied to two datasets, movies, and twitter messages. Hu et al. paper [26] introduced a framework composed of two main phases based on neural network for sentiment analysis. The first phase, feature vectors are obtained through linguistic and domain knowledge. And the second phases, a Deep Neural Network is designed. The approach was evaluated on three datasets (electronic products, movies reviews, and hotels reviews). Ruder et al. [26] introduced an approach to aspect-based sentiment analysis. They used a convolutional neural network (CNN) for aspect extraction and sentiment analysis. Several domains such as restaurants, hotels, laptops, phones, and cameras were evaluated in this proposal. Severyn and Moschitti [27] proposed a deep learning model which is applied to two tasks of SemEval 2015, namely, message-level and phrase-level of Twitter sentiment analysis. Sun et al. [28]0 proposed a Deep Neural Network model for sentiment analysis approach applied in Chinese microblog. The proposed method extracted features to obtain semantic and information of words. To evaluate and prove the effectiveness of the method three models were selected SVM, Naïve Bayes, and Deep Neural Network. Kalchbrenner [29] et al. and Yoon Kim [30] presented different CNN models for sentiment classification, respectively. Their models could handle the input sentences with varying length then capture short and long-range relations. The Kim’s model [30] different because his model had some hyperparameter tuning and can be trained on pre-trained word vectors. [11][24]

3. MACHINE LEARNING AND DEEP LEARNING

In this section, we present the overall of machine and deep learning and make a comparison these two methods.

3.1. Machine Learning

“*Machine learning is a data analytics technique that teaches computers to do what comes naturally to humans and animals: learn from experience. Machine learning algorithms use computational methods to “learn” information directly from data without relying on a predetermined equation as a model. The algorithms adaptively improve their performance as the* *number of samples available for learning increases*" [31] .To learns information from data, two main techniques are usually used (figure 1): supervised learning and unsupervised learning.

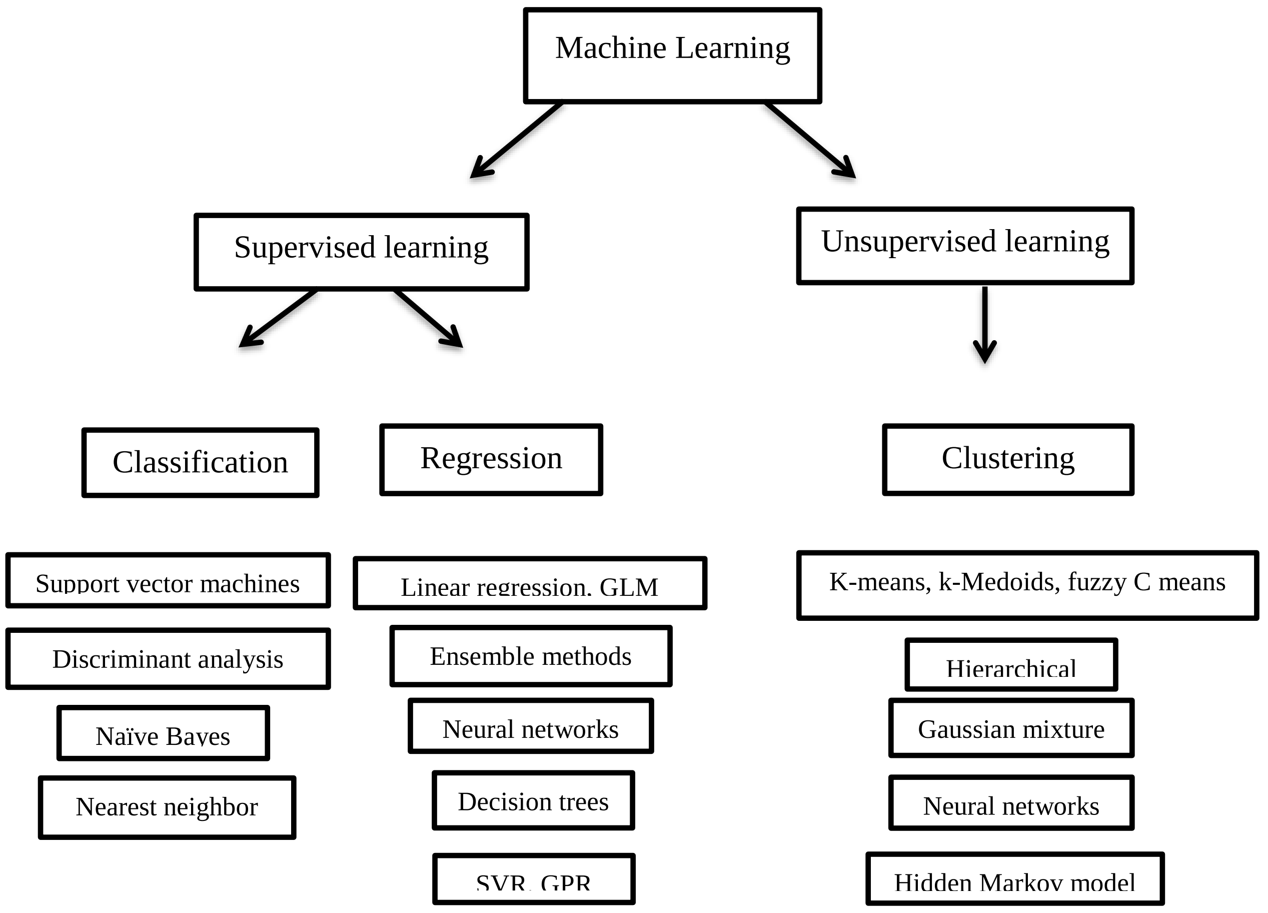


Figure1. Machine learning techniques include both unsupervised and supervised learning

3.1.1.Supervised Machine Learning Method

Supervised machine learning is a technique that builds a model to predicts evidence in the presence of uncertainty.it takes a set of input data then responses to an output data and trains a model to generate reasonable predictions for the response to new data. The classification and regression techniques are used in supervised learning to develop and predict model.[31]

"*The Classification techniques is used to predict discrete responses.The common algorithms used to perform classification include support vector machine (SVM), boosted and bagged decision trees, k-nearest neighbor, Naïve Bayes, discriminant analysis, logistic regression, and neural networks. The regression techniques is used to predict continuous responses.it is often used when you want to work with data range such as changes in temperature. The common algorithms used are linear model, nonlinear model, regularization, stepwise regression, boosted and bagged decision trees, neural networks, and adaptive neuro-fuzzy learning.* "[31]

3.1.2. Unsupervised Machine learning method

The Unsupervised machine learning technique is a used to get inferences from datasets consisting of input data without labeled responses. the most common unsupervised learning technique is clustering . the clustering technique is used to explore data analysis and find hidden patterns or grouping in data.It is modeled using a measure of similarity which is defined upon metrics such as Euclidean or probabilistic distance. The common algorithms used are k-means and k-medoids, hierarchical clustering, Gaussian mixture models, hidden Markov models, self-organizing maps, fuzzy c-means clustering, and subtractive clustering.[32]

3.1.3. Other Machine Learning techniques

A part from the two techniques described above . There are other techniques : Semi-supervised Learning and Reinforcement Learning which can also be used. The figure2. Illustrated the overview of machine learning techniques.

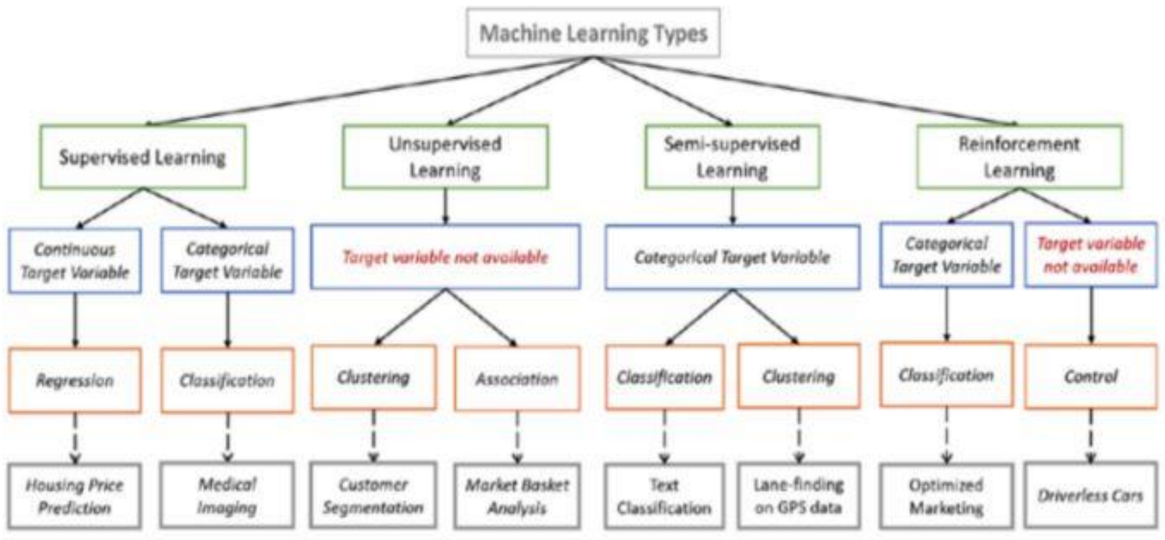


Figure 2. Overview of machine learning techniques[33]

3.2.Deep Learning

“*Deep learning (also known as deep structured learning or hierarchical learning) is part of a broader family of machine learning methods based on learning data representations, as opposed to task-specific algorithms. Learning can be supervised, semi-supervised or unsupervised in that perform classification tasks directly from images, text, or sound*“[34].The deep word means that a number of layers in the network. So having more layers lead to a better and deeper network. In deep learning, each stage learns to modify the received input data in a slightly more drawn away and composite representation for instance with an image recognition application, the image used as input gets a matrix of pixels where the first layers extract the pixels and encode edges; the second layers may comprise and encode arrangements of edges;the third layer encodes some part of the images such as a nose and eyes; and the fourth layer may acknowledge that the image contains a face. More important fact is that a deep learning process learns by its owns the features need to optimally take place in each level. Therefore, the training process of deep neural networks learns to produce an error signal that calculates the difference between the predictions of the network and the values desire then use this error signal to alter the weights or parameters to predict more accurate. Figure 3 shows Training process of deep learning. [34]

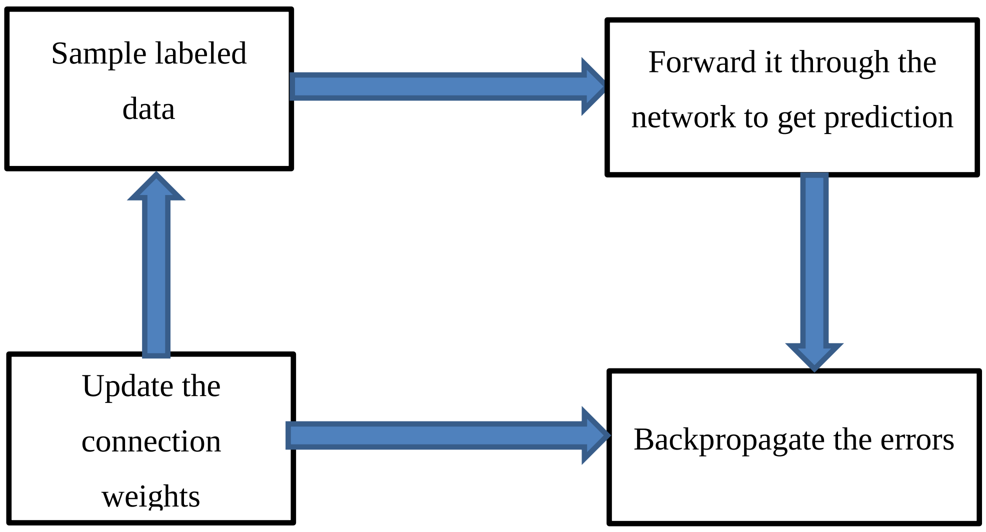

Sample labeled data
Forward it through the network to get prediction
Backpropagate the errors
Update the connection weights

Figure 3.Training process of deep learning

3.2.1. Type of deep learning networks

Generally, we can categorize the deep learning network into four categories: Unsupervised Pre-trained Networks (UPNs), Convolutional Neural Networks (CNNs), Recurrent Neural Networks and Recursive Neural Networks

3.2.1.1. Unsupervised Pre-trained Networks (UPNs)

This network is also known as representation learning. It contains three architectures: Autoencoders, Deep Belief Networks (DBNs), Generative Adversarial Networks (GANs).

The Autoencoders are neural network where the inputs are copied to the outputs. It works by reducing the input into a latent-space representation then rebuilding the output from this representation. This network (see figure 4) is divided into two parts: Encoder and Decoder. The encoder is where the input is compressed into a latent-space representation. While the decoder is where the input from the latent space representation is reconstructed.[35]

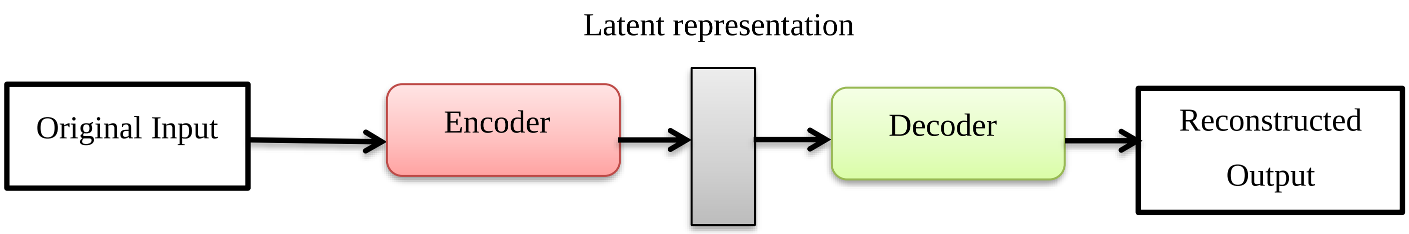


Figure 4. Architecture of an Auto-encoder

Deep Belief Networks are composed of layers of Restricted Boltzmann Machines (RBMs) for the pretrain stage and a feed-forward network for the fine-tune stage Figure 5 shows the network architecture of Deep Belief Networks. [36]

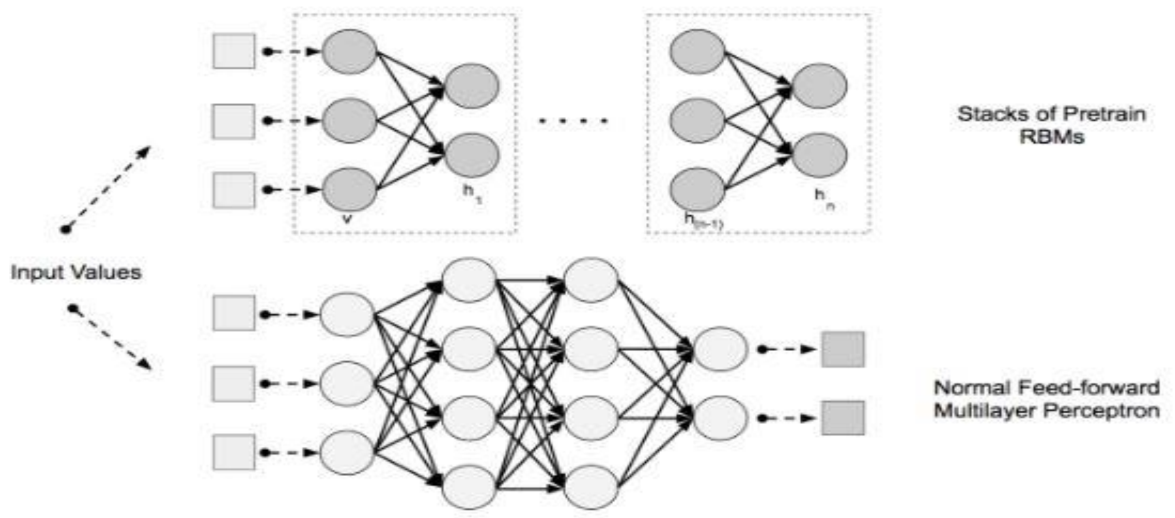


Figure 5. Deep Belief Networks architecture [36]

3.2.1.2. Convolutional Neural Networks (CNNs)

The aim of Convolutional Neural Network to get higher-order features in the data through convolutions, they are suitable for object recognition with images and constantly top image classification competitions. The Convolutional Neural Network can recognize faces, individuals, street signs, platypuses, and many other aspects of visual data. They overlap with text analysis via optical character recognition, but useful when analyzing words as discrete textual units. And also good at analyzing sound. [36]

It is consisted of three main layers (see figure 6): input layers, feature –extracted layers and classification layers. The workflow starts with the input layer which alters the input data then send through all connected layers and set a class score that is given by the output layer. Figure6 shows the high level of Convolutional Neural Network architecture.

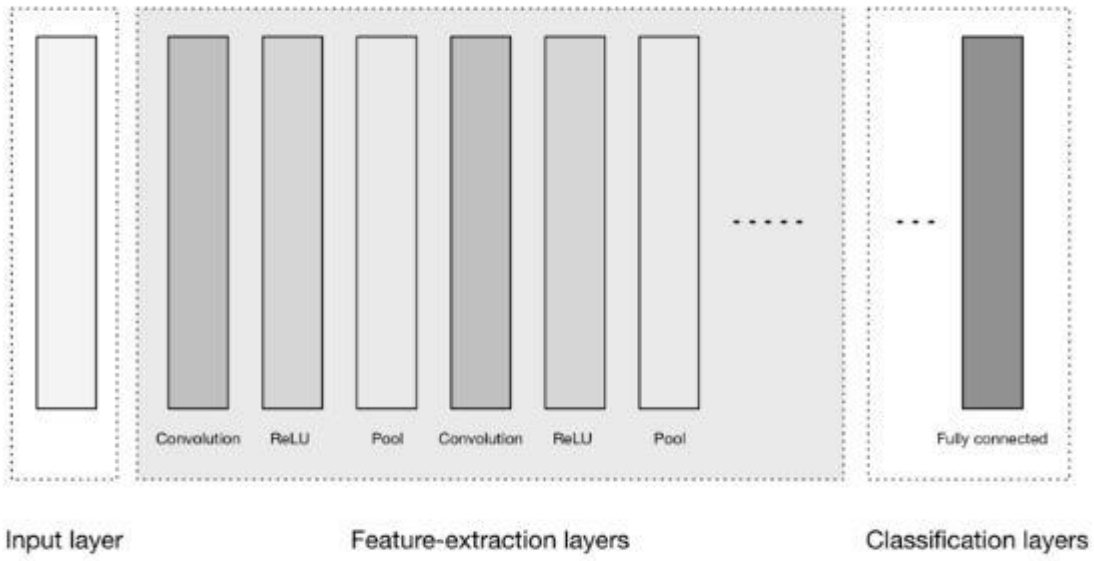


Figure 6. High-level general CNN Architecture [36]

3.2.1.3. Recurrent Neural Networks

Recurrent Neural Networks belong to group of the feed-forward neural networks. It differs with other feed-forward networks by its capability to send information over time-steps. A remarkable description of Recurrent Neural Networks from the researcher Juergen Schmidhuber is as follow: *"Recurrent Neural Networks allow for both parallel and sequential computation, and in principle can compute anything a traditional computer can compute. Unlike traditional computers, however, Recurrent Neural Networks are similar to the human brain, which is a large feedback network of connected neurons that somehow can learn to translate a lifelong sensory input stream into a sequence of useful motor outputs. The brain is a remarkable role model as it can solve many problems current machines cannot yet solve."*[36]

The Recurrent Neural Networks model takes one at a time every vector that comes from a series of input vectors then model them. This mechanism able the network to maintain the state meanwhile it is modeling every input vector across the window of input vectors. Figure7 shows the difference between Normal input vectors and recurrent neural networks input [36].

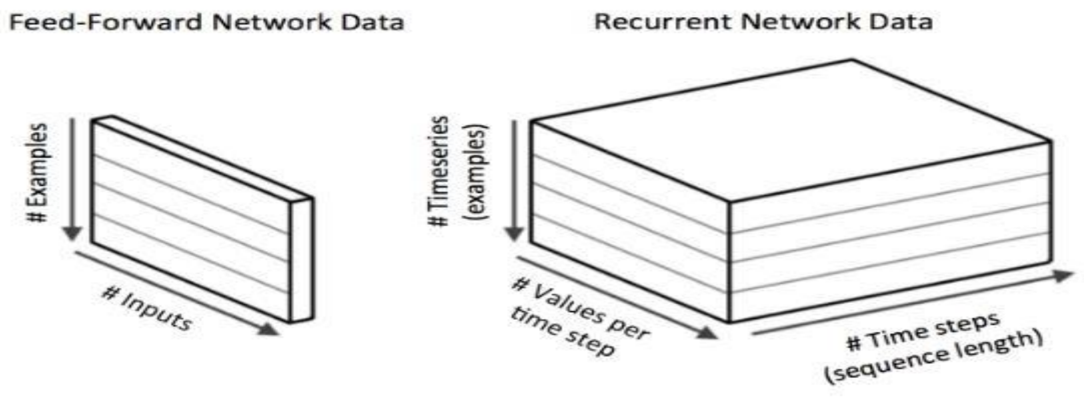


Figure 7.the difference between Normal input vectors and recurrent neural networks input [36]

3.2.1.4. Recursive Neural Networks

Recursive Neural Networks is similar to Recurrent Neural Networks but the difference is from its capability to model the ordered structures in the training dataset. "Its architecture is made of a shared-weight matrix and a binary tree structure that enable the recursive network to learn varying sequences of words or parts of an image. It uses a variation of backpropagation called backpropagation through the structure (BPTS). The feed-forward pass happens bottom-up, and backpropagation is top-down. [36]

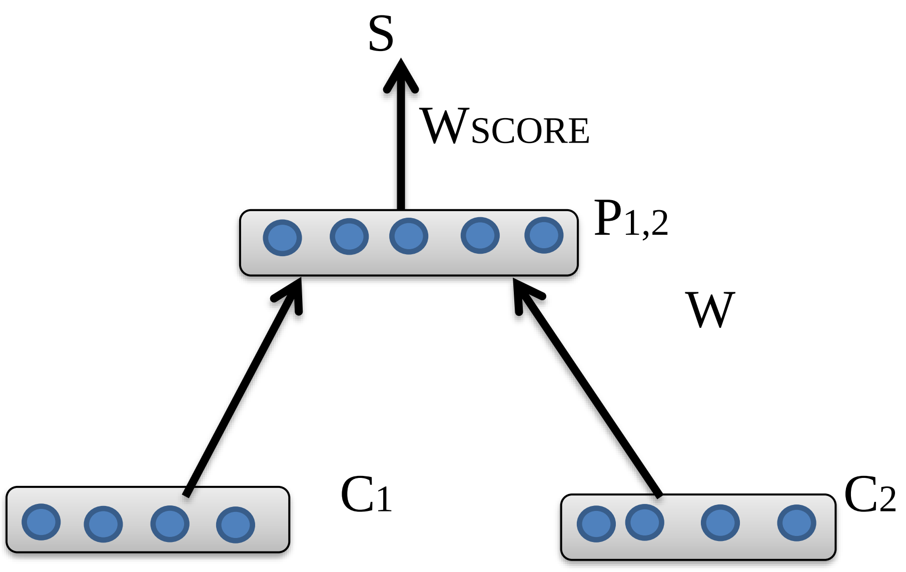


Figure 8.A simple recursive neural network architecture

3.2.2. Other type of deep learning network.

An artificial neural network (ANN) is a nonlinear neural network model based on the neural structure of the brain. It is capable to learn to perform tasks just by considering examples like classification, prediction, decision-making, visualization, and others. It is composed of processing elements and organized in three interconnected layers (see figure 9): input, hidden that may include more than one layer, and output. [37].

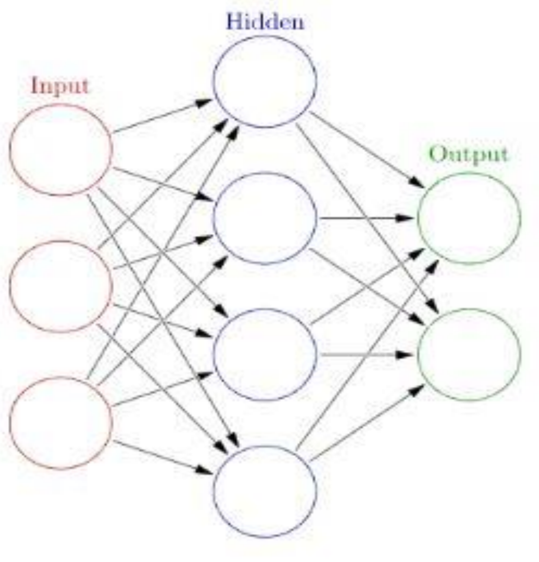


Figure 9. Artificial Neural Network architecture [37]

A multilayer perceptron (MLP) is neural networks that contain three or more layers. It uses nonlinear activation functions that classify data if it is not linearly separable. In this network (see figure 10) each node in one layer is connected to another node of the following layer to form a fully connected network. The activation function is mainly hyperbolic tangent or logistic function. [37]

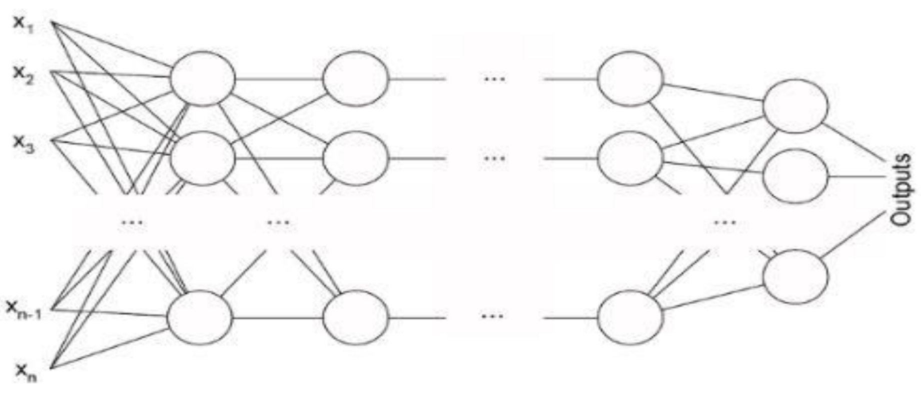


Figure 10. The multilayer perceptron network overview [37]

Long short-term memory (LSTM) is exact as recurrent neural network (RNN) architecture. It is designed to model temporal sequences and their long-range dependencies more accurately than conventional RNNs. The activation function is not used within its recurrent components in

LSTM, this allows the stored values to do not be modified, and the gradients not tend to vanish during training. Usually, units in this network are implemented in “blocks” of several units (see figure 11). These blocks contains three or four “gates” such as input gate, forget gate and output gate that control information flow drawing on the logistic function.[37]

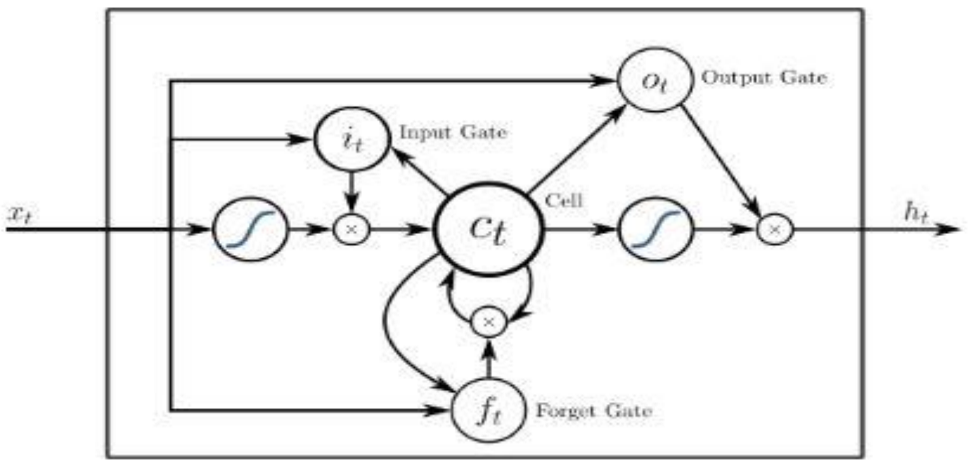


Figure 11. LSTM block with input, output, and forget gates. [37]

In the paper [38] a model for part-of-speech (POS) tagging was presented. This model achieved a performance of 97.40% tagging accuracy. Apple, Amazon, Google, Microsoft and other companies incorporated LSTM as a fundamental element into their products. [37]

3.3. Difference between machine learning and deep learning

This section presents some important points of deep learning and machine learning then compare the two techniques.

3.3.1. Data dependencies

When it comes for data dependencies the Machine Learning can perform well even with a small size of data while Deep Learning doesn’t perform that good with small set of data because it requires a large set of data in order to make a brief conclusion.

3.3.2. Hardware dependencies

Regarding the hardware dependencies, again the Machine Learning tends to perform well even on low-end machines. However, with Deep learning this one need a high-end machine to perform very well because its requirements make GPUs part of its working.

3.3.3. Feature engineering

For the feature engineering, Deep Leaning outperformed Machine learning since Deep learning has the ability to make the feature itself while the Machine Learning requires the user to identify the exactitude of the feature.

3.3.4. Problem Solving approach

The problem-solving method of Deep Learning is better than Machine Learning one because Deep Learning the problem are solved end to end while Machine Learning solve the problem by splitting into small pieces then solve one by one and get a result by combining all of them

3.3.5. Execution time

Regarding the execution time, Machine learning outperformed Deep Learning since it takes less time to train data while Deep Learning requires more time to train due to the number of parameters present. Whereas, the testing time seems to be the opposite of execution time during the testing time a deep learning algorithm doesn‘t take much time compare to machine learning

4. DEEP LEARNING FRAMEWORK

In this section, we introduce a comparison of some most used deep learning framework and present our selected deep learning framework.

4.1. Comparison of Deep Learning Framework

With the advent of Deep Learning, many frameworks have come up to make easy the implementation. The choice regarding deep learning framework depends on the business requirement of organization or personal choice of a researcher who wants to use it but mostly the choice is based on certain criteria such as modeling capability, architecture, and number programing language supported, speed and more. Table 2 shows a comparison of some deep learning framework.

	Language	**Tutorials and training material**	**CNN modeling capability**	**RNN modeling capability**	**Architecture: easy-to-use and modular front end**	**Speed**	**Multiple GPU support**	**Keras compatible**
Theano	**Python**	++	++	++	+	++	+	+
Tensor Flow	**Python, Java, Go,R, Julia**	+++	+++	++	+++	++	++	+
Torch/ Pytorch	**Lua, Python**	+	+++	++	++	+++	++	
Caffe	**C++, Python**	+	++		+	+	+	

Mxnet	**R,Python, Julia,Scala, Go,Perl, C++, JavaScript**	++ ++	++ ++	+ +	++ ++	++ ++	+++ +++
Neon	**Python**	+	++	+	+	++	+
CNTK	**C++**	+	+	+++	+	++	+

Table 2. Ranking Comparison of Deep Learning frameworks (continued).

For our research purpose, we personally choose to use Apache Mxnet as our deep learning framework because after running a test with two frameworks: TensorFlow and Mxnet on CPU. The Apache Mxnet has shown a good running time performance on CPU, it hasn't taken much time compare to Tensorflow. The next section we present the Mxnet and it is architecture.

4.2 MXNET

"*Apache MXNet is a modern open-source deep learning framework used to train, and deploy deep neural networks. It is scalable, allowing for fast model training, and supports a flexible programming model and multiple languages (C++, Python, Julia, Matlab, JavaScript, Go, R, Scala, Perl, Wolfram Language).It is a lean, flexible, and ultra-scalable deep learning framework that supports state of the art in deep learning models, including convolutional neural networks (CNNs) and long short-term memory networks (LSTMs).*" [39]

4.2.1. Mxnet system architecture

The Mxnet system architecture is composed of different module which interacted with one another. The figure 12 shows how the major modules and components of the Mxnet system and their interaction. These modules are classified as follow:

- Runtime Dependency Engine : this module is where schedules ans execution of the operation are taking place according to their read or write dependency.
- Storage Allocator: the role of this module is to efficiently allocates and recycles memory blocks on host (CPU) and devices (GPUs).
- Resource Manager: the resource manager is the one that manages global resources, for instance the random number generator and temporal space.

- NDArray: this role of this NDArray is to dynamic, asynchronous n-dimensional arrays which give flexible imperative programs for MXNet.
- Symbolic Execution: this module is a static symbolic graph executor, in which efficient symbolic graph execution and optimization are provided
- Operator: this is where the static forward and gradient calculation (backprop) is defined by the operators.
- SimpleOp: SimpleOp is an Operators that extend NDArray operators and symbolic operators in a unified fashion.
- Symbol Construction: this module is a symbolic construction, which gives a way to construct a computation graph (net configuration).
- KVStore: this is a key-value store interface for efficient parameter synchronization.
- Data Loading (IO): the data loading is the module where the Efficient distribution of data is loaded and augment.[40]

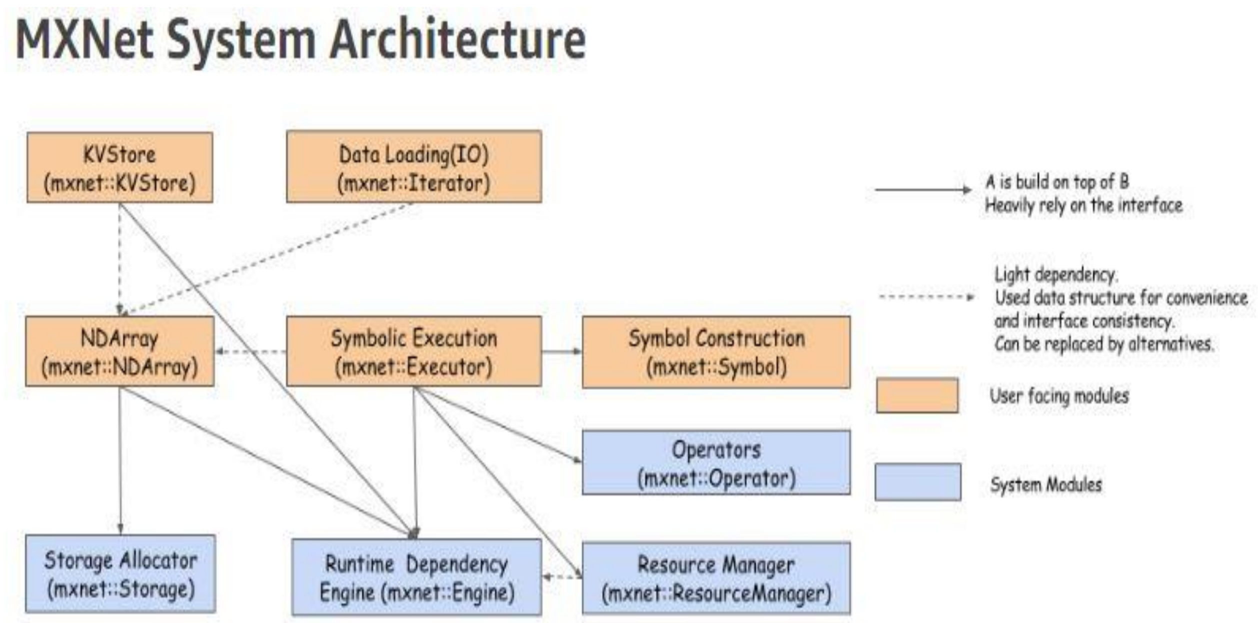

MXNet System Architecture
KVStore (mxnet::KVStore)
Data Loading(IO) (mxnet::Iterator)
NDArray (mxnet::NDArray)
Symbolic Execution (mxnet::Executor)
Symbol Construction (mxnet::Symbol)
Operators (mxnet::Operator)
Storage Allocator (mxnet::Storage)
Runtime Dependency Engine (mxnet::Engine)
Resource Manager (mxnet::ResourceManager)
A is build on top of B
Heavily rely on the interface
Light dependency.
Used data structure for convenience and interface consistency.
Can be replaced by alternatives.
User facing modules
System Modules

Figure 12. The Mxnet system architecture [40]

5. PROPOSED APPROACH

Our proposed approach for text classification using deep learning approach is presented in this section. It is one-dimensional convolutional neural networks model which predicts sentiment polarities of given French sentence. We divided our work four main modules: firstly, we collect dataset need to be used to feed our network then using an automatic creation to get French corpus Secondly we use the preprocessing method to clean our text then use word embeddings to obtain the feature vectors. Finally, configure the CNN model to train and classify text as positives or negatives. Figure 13 shows the workflow of our proposed approach

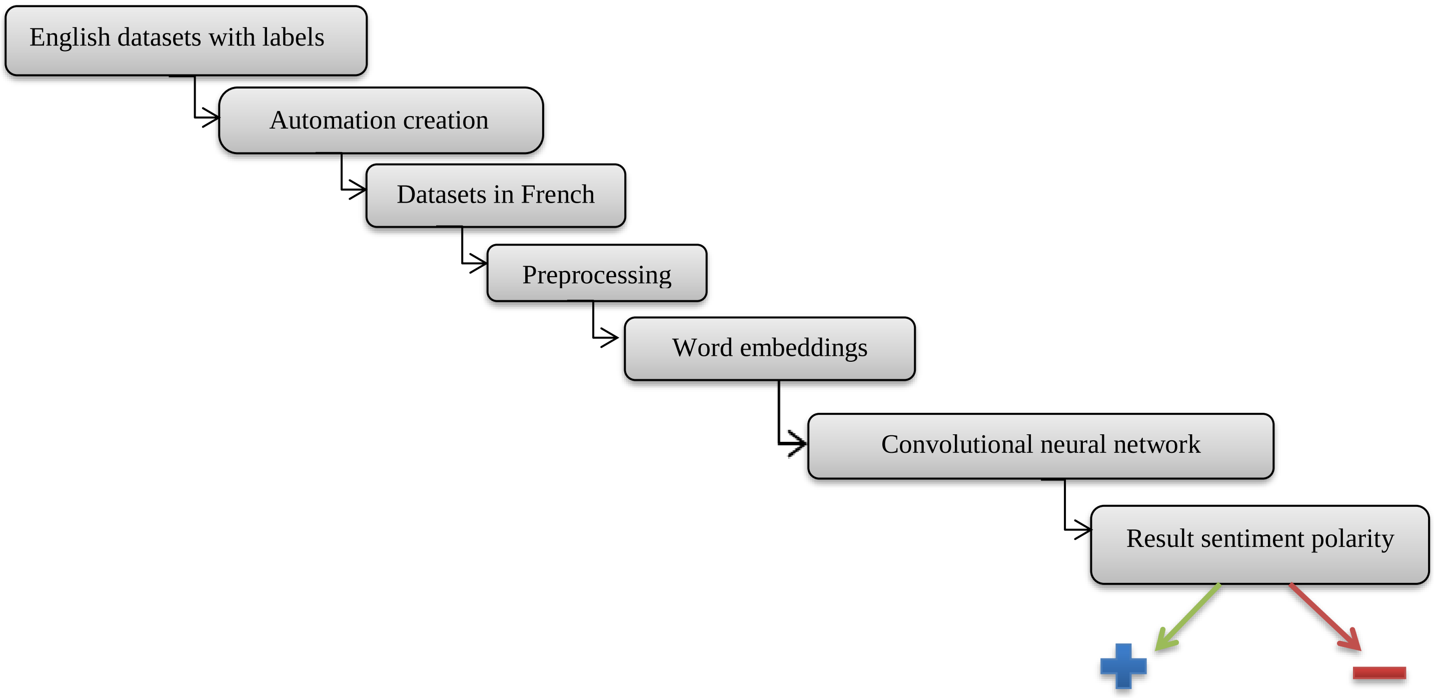

English datasets with labels
Automation creation
Datasets in French
Preprocessing
Word embeddings
Convolutional neural network
Result sentiment polarity

Figure 13. Workflow of our propose approach

5.1. Automate creation

Before using the automatic creation we manually separate dataset collected into two files based on the polarity (positive and negative)correct then grammar of some text, remove short the contraction because some of them may not get translated well and proceed for the automatic creation. The goal of this module is to get in French all the English text from the datasets. Usually, the translation is done in three different resources i.e. an aligned resource; a comparable corpus and a multilingual encyclopedia. In our approach, we use the automatic translator available on the web because we lack all these resources. Three online translators were used in this case i.e. Google Translate, Bing Translate, and Collins Translator. After translation, the translation from Collins translator was kept since it has responded to our satisfaction compare to the other two.

5.2. Preprocessing

Before training our model with translated data, we need to do the preprocessing of text. This is the process where the two dataset files are which is split into words and generates labels and returns split sentences and labels. After this, data are tokenized then turn them into lowercase and build vocabulary file from training data followed by padding all sentences to the maximum sentence length which is defined by the longest sentence and padded sentences are returned. It is very useful to do padding sentences to the same length because of it able us to proficiently batch our data. In our model implementation the python code data_loader.py will take care of all these processes. After all, these processes will be completed our input data will be ready to be used in our deep neural network.

5.3. Word embeddings

Word embeddings are techniques use to represent every word in the vocabulary as a real-valued vector in a predefined vector dimension, usually hundreds of dimensions. In this approach, every phrase gets map to individual vector and the vector values learn the way to look like a neural network. Word embeddings process of learning is either link to some task with the neural network model for instance document classification or is an unsupervised process. Generally, there are three techniques used to learn a word embedding from corpora of text i.e.: embedding layer, word2vec, and gloves. The next section describes these three techniques. [41]

5.3.1.Embedding layer

This is word embedding which learns collectivily along with neural network model on a particular task of natural language processing for example document classification. in this technique a corpus text is needed to be cleaned and get ready every word as one-hot encode. the vector space size is usually integrated part of the mode for example 50, 100, 300 and more dimensions.vectors are made ready with a small random number.*“ The embedding layer is used on the front end of a neural network and is fit in a supervised way using the Backpropagation algorithm.*“[41]

In this layer the mapping of word vectors of are done through one hot encode words . this mapping happens whether the a multilayer Perceptron model is used or a recurrent neural network is used. In the first model the word vectors are joined together before to being supplied as one input to the model while for the second model individual word is taken sequently as one input. This way of learning need much data for trainng and may be slow however it will learns the embedding in both way : particular text data and the natural language process task.[41]

5.3.2.Word2vec

The word2vec is a collection of similar models that are used to make word embeddings. These models are simple, neural networks with two-layer that are used to train and build linguistic contexts of phrases. Word2vec gets an input of large corpora of text and creates a vector space, generally hundred of dimensions, where each individual word in the corpora is being designated to a corresponding vector space. Those word vectors are placed in the vector space so that words with similar contexts in the corpus are placed close one to another in the space.It was developed at Google by Tomas Mikolov and his team. The algorithm developed was consequently analyzed and many other researchers have explained it too. Additionally to this, analyzing the learning vectors and examining to represent words in the vector math were part of this work.. The technique used two different learning approach to learn the word embedding: CBOW model and Skip-Gram Model. The Continuous Bag-of-Words is a model that predicts the actual word based on its content while learning the embedding. And the skip-gram is model that predicts the words around for a given a actual word. Figure 14 shows Word2Vec Training Models.[42]

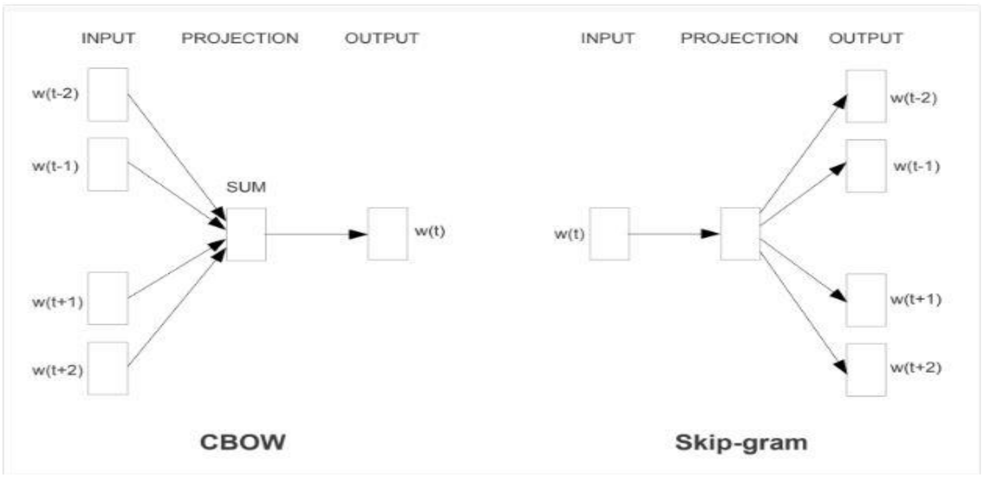


Figure 14. Word2Vec Training Models.[41]

The main advantage of using this approach is that word embeddings learn effevectively in high-quality word and allow great size of embeddings to be learned from much bigger corpus of text.[41]

5.3.3.Gloves

"*GloVe is an unsupervised learning algorithm for obtaining vector representations for words. Training is performed on aggregated global word-word co-occurrence statistics from a corpus,* *and the resulting representations showcase interesting linear substructures of the word vector space*"[43].It was created by team of researchers led by Pennington at Stanford in 2014. It is a methods that put the global statistics of matrix factorization methos such as Latent Semantic Analysis with the content-based learning in word2vec."*it is a new global log-bilinear regression model for the unsupervised learning of word representations that outperforms other models on word analogy, word similarity, and named entity recognition tasks*"[41][44].

Based on these differences of word embedding techniques described above, in our CNN-rand model, we use embeddings layer and set size of the vector space to learn from scratch. However, for CNN-static we use a specific French word embeddings models named frWac2Vec. Since this specific pre trained word embedding is a French-English corpus. We use the 1000-dimensional embeddings "*computed using Word2vec skip-gram approach on a 1.6 billion word corpus constructed from the Web limiting the crawl to the .fr domain*".[45] Hence, this pre trained word embedding respond to our need of training a corpus of French text with a word embedding.

5.4. Convolutional neural networks

"*In machine learning, a convolutional neural network is a class of deep, feed-forward artificial neural networks that has successfully been applied to analyzing visual imagery*."[46]. It is composed of neurons that have learnable weights and biases. Each neuron receives some inputs, performs a dot product and optionally follows it with non-linearity. The whole network still expresses a single differentiable score function: from the raw image pixels on one end to class scores at the other. And they still have a loss function (e.g. SVM/Softmax) on the last (fully-connected) layer. [47].

In this research, we use a convolutional neural network for text classification applied with French text because the Convolutional Neural Network has shown to be the best choice in many literature for text classification compare other neural networks. A good example is from Xiang Zhang et al. [48] , they have proven that deep CNN don't need to get knowledge about the syntactic or semantic structure of a language. This mechanism could be important for a single system that needs to work with different languages; because characters always compose a necessary construct despite of splitting up into words is possible. [48]. we train our model to learn and classify the sentiment in French text by polarity either positive or negative. Regarding the implementation of the model baseline, CNN-rand and CNN-static baseline models are used. The architecture of convolutional neural network (see figure 15.) is a block of layer built from three different layers: convolutional layer, pooling layer, and fully-connected layer.

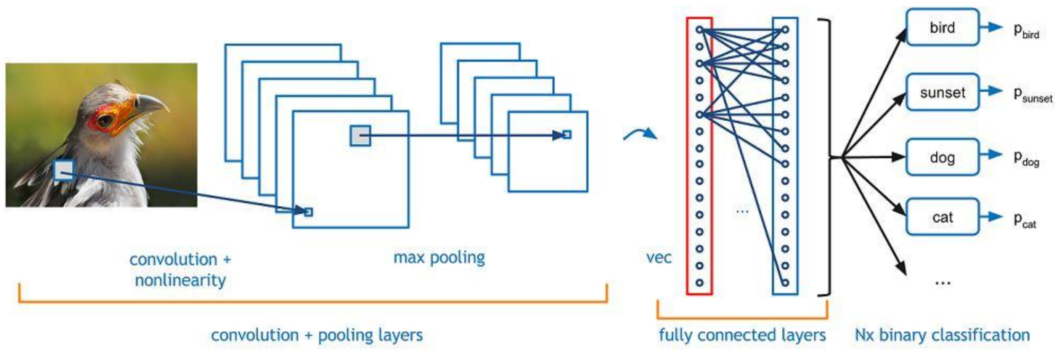


Figure 15. Overview of Convolutional Neural Network architecture [49]

5.4.1. Convolutional layer

This is the main part of CNN. The parameters consist of a number of learnable kernels that contains a small receptive field but through the full depth of the input, the volume is extended. During the forward pass, each filter convolves to across the width and height of the input volume by computing the dot product between the entries of the filter and the input then producing a 2-dimensional activation map of that filter. As a result, the network learns filters that activate when it detects some specific type of feature at some spatial position in the input. By intuition, the network will learn filters that activates when they see some type of visual feature such as an edge of some orientation on higher layers of the network. Now, we will have an entire set of filters in each CONV layer (e.g. 10 filters) then each of them will produce a separate 2-dimensional activation map and will stack these activation maps along the depth dimension to produce the output volume. Generally, the convolutional layer get an input volume of size $W1 \times H1 \times D1$ then requires four hyperparameters to be set(Number of filters (K), spatial extent (F), stride (S) and an amount of zero padding (P)) and produces a volume of size $W2 \times H2 \times D2$ where: $W2 = (W1-F+2P)/S+1$, then $H2 = (H1-F+2P)/S+1$ (i.e. width and height are computed equally by symmetry) and $D2=K$. When the parameter is shared, it introduces $F \cdot F \cdot D1$ weights per filter, for a total of $(F \cdot F \cdot D1) \cdot K$ weights and K biases. In the output volume, the d-th depth slice (of size $W2 \times H2$) is the result of performing a valid convolution of the d-th filter over the input volume with a stride of S and then offset by d-th bias.[47]

5.4.2. Pooling layer

The role of pooling layer is to reduce the spatial size of the representation progressively in order to reduce the amount of parameters, computation in the network and control overfitting. It operates individually on each depth slice of input and by the used of MAX operation, it gets resizes spatially (see figure 16). the common form of pooling layer is with filters of size 2x2 applied with a stride of 2 down samples every depth slice in the input by 2 along both width and height, discarding 75% of the activations.in this case every MAX operation would take a max over 4 numbers. The depth dimension remains unchanged. Generally, the pooling layer takes an input volume of size W1×H1×D1 then requires two parameters to be set spatial extent (F) and stride (S)and produce an output volume of size W2×H2×D2 where: W2=(W1−F)/S+1 then H2=(H1−F)/S+1 and D2=D1. Commonly in practice, there are only two seen variations of the max pooling layer: a pooling layer with F=3, S=2 also known as overlapping pooling and with F=2, S=2. Pooling sizes with larger receptive fields tend to be too destructive. Additionally, other functions such as average pooling or even L2-norm pooling but it has fallen out of favor compared to the max pooling operation due to the better work in practice with max pooling operation compare average pooling.[47]

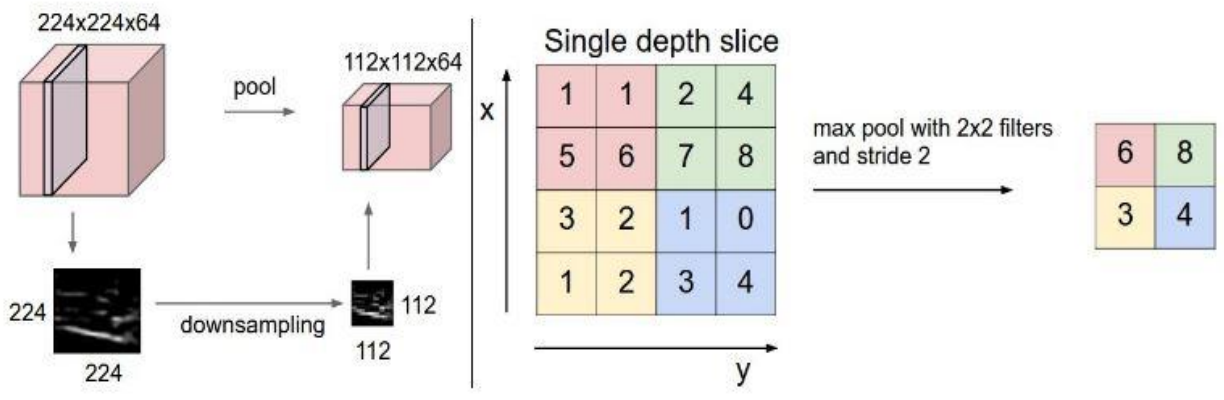


Figure 16. Example of pooling layer workflow[47]

„*Pooling layer downsamples the volume spatially, independently in each depth slice of the input volume. Left: In this example, the input volume of size [224x224x64] is pooled with filter size 2, stride 2 into output volume of size [112x112x64]. Notice that the volume depth is preserved. Right: The most common downsampling operation is max, giving rise to max pooling, here shown with a stride of 2. That is, each max is taken over 4 numbers (little 2x2 square)*“.[47]

5.4.2. Fully-connected layer

The function of the fully connected layer is to connect each neuron in one layer to every neuron in another layer meaning that all neurons have full connections to all activations in the previous layer. Hence their activation is computed with a matrix multiplication followed by a bias offset. This principle is the same as the traditional multi-layer perceptron neural network [47].

For our proposed model we define first, embedding layer which encodes the word vectors as a float vector. This layer will learn to map word vectors into a lower dimensional vector space where distances between words correspond. So, in this way we let the network to find out what is the best embedding for this specific dictionary problem and objective function by taking as input of N vectors of integer and returns NxN float matrixes. The embedding layer is only used in the CNN-rand model. Figure 17 shows Convolutional Neural Network Architecture for Sentence Classification. [52]

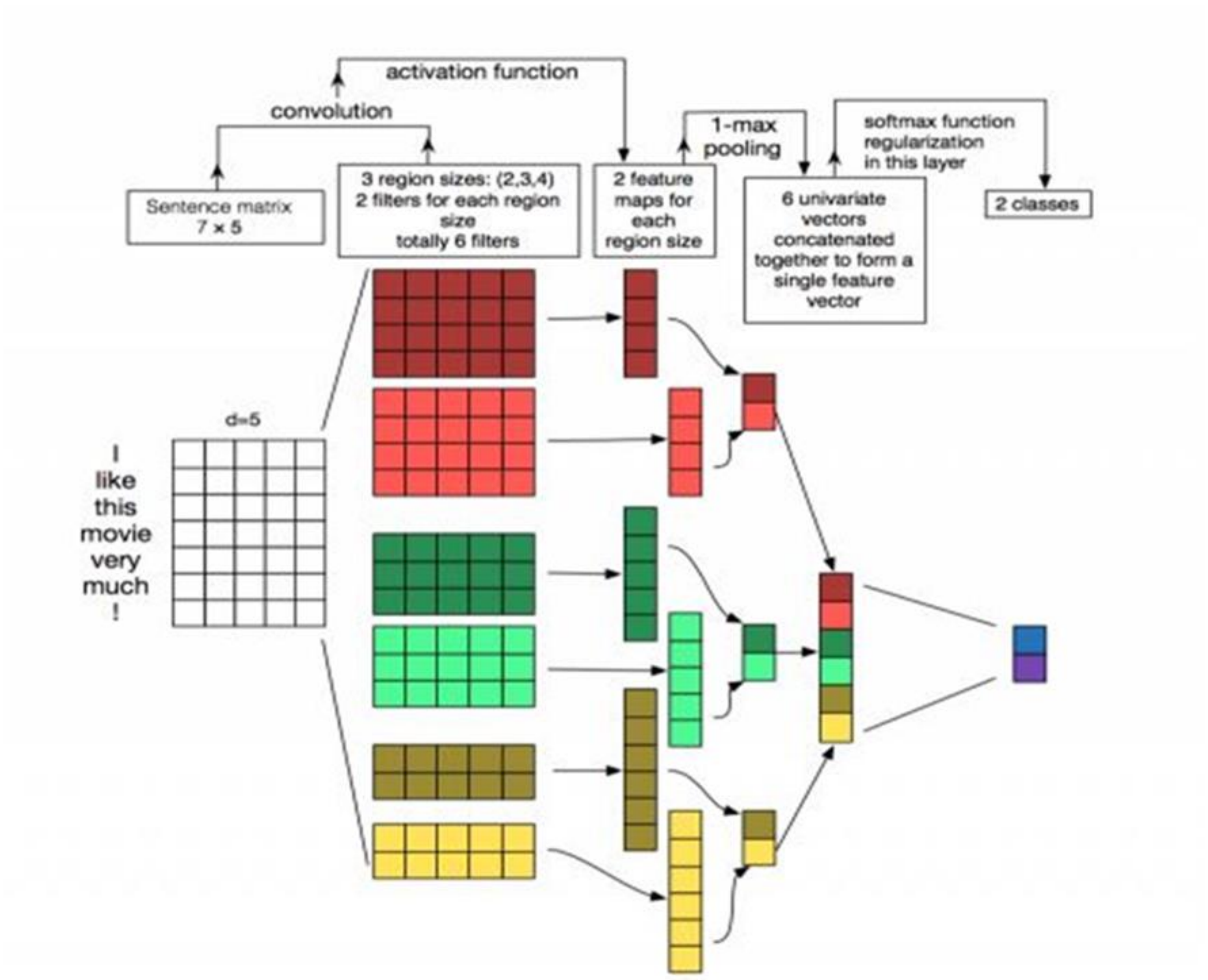

activation function
convolution
1-max pooling
softmax function regularization in this layer
Sentence matrix 7 × 5
3 region sizes: (2,3,4) 2 filters for each region size totally 6 filters
2 feature maps for each region size
6 univariate vectors concatenated together to form a single feature vector
2 classes
d=5
I
like
this
movie
very
much
!

Figure 17. Convolutional Neural Network Architecture for Sentence Classification [51]

The next layer consists of the use of multiple filter sizes that slides over 3, 4or 5 words at a time to perform in the network over the ordered embedded word vectors in a sentence. This will give us an understanding of how words contribute to sentiment in the context of those around them which is the equivalent of looking at all 2-grams, 3-grams and 4-grams in a sentence. Then we add a max pool for each convolution to extract the most elements significantly in each convolution and turn them into a feature vector. This is because each convolution with pool filter produces tensors of different shapes we need to create a layer for each of them, and then concatenate the results of these layers into one big feature vector. And adding a dropout for regularization, which randomly disables a fraction of neurons in the layer to ensure that that model does not overfit. This prevents neurons from co-adapting and forces them to learn individually useful features. Finally, we add a fully connected layer to add non-linearity to the model. We then classify the resulting output of this layer using a softmax function, yielding a result between 0 (negative sentiment) and 1 (positive). [52][53]

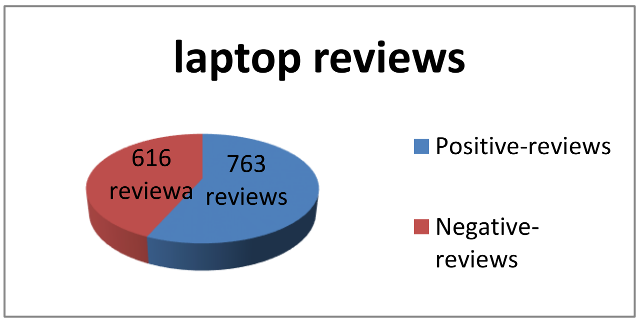
6. EXPERIMENTAL SETUP AND DATASETS

The experimental setup and sentimental analysis dataset are presented. The dataset structure is described first then the model setup and variant. And also the software tool used for implementing.

6.1. Sentimental Analysis Dataset

The dataset we use for our experimental are collected from SemEval2015-ABSA (Task 12). This task was a continue aspect based sentimental analysis task of SemEval14-ABSA (task 4). In SE-ABSA15 there was two subtasks in-domain ABSA and an out-of-domain ABSA. The first one the task was to identify entity and attribute expressed in the text(slot1) then Opinion Target Expression which is an expression used in the given text to refer to the reviewed entity E of a pair E#A(slot2) and sentiment polarity to identify the polarity of each aspect(slot3) . While the second one there was just task perform the slot1 and slot3. The in-domain subtask used data from laptop and restaurant while the out-of-domain focuses in hotel domain.

Since the goal of our research is to classify the sentiment polarity related services and products. We use corpus from in-domain ABSA: laptops and restaurants. And focus on the task performed on slot3. These are ready-made dataset where each review is tagged with aspect entity type, aspect attribute type, and polarity. Figure 18 shows the distribution size of a dataset in both domains. [50]



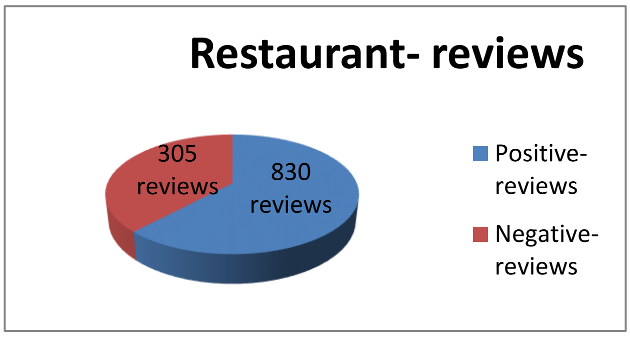


Figure 18. The distribution size of dataset in laptops and restaurants.

6.3. Model Variations

In our experimental two baseline models are used i.e.CNN-rand and CNN-static to evaluate the effectiveness of our model approach. The first one is a model in which we used embeddings layer so that all words are randomly initialized and then modified during training. And the second one is a model in which we use a pre-trained vectors frWac2Vec. All words— including the unknown ones that are randomly initialized—are kept static and only the other parameters of the model are learned.[30]

6.2. CNN configuration

For CNN-rand, we set an embedding vector size to follow a maximum length of the sequence and a vocabulary size. We also set kernel (filter windows) with feature maps, a dropout, learning rate, a mini batch size and percentage of test set from training data. Also, use transfer function RELU activation for convolution layers. Table 3 shows the hyperparameter overview for CNN-rand.

embedding vector size	maximum length of sequence	vocabulary size	filter windows	feature maps	dropout	learning rate	batch size	percentage of test set	epochs
128	50	8000	3,4,5	50	0.2	0.02	50	0.1	10

Table 3.the overview of hyper parameter for CNN-rand

For CNN-static, we use pre-trained word embeddings and set kernel (filter windows), the feature maps, a dropout number of epochs, a learning rate, a mini batch size and percentage of test set from training data and also use a transfer function RELU activation for convolution layers, Table 4 shows the hyperparameter overview for CNN-static.

Word embeddings	filter windows	feature maps	Dropout	learning rate	batch size	percentage of test set	Epochs
frWac2Vec	3,4,5	50	0.5	0.01	50	0.1	10

Table 4.the overview of hyper parameter for CNN-static

6.4. Software Tools

Regarding the implementation of our proposed model, we use python for coding, a deep learning framework: Mxnet. And install different machine learning functions and utilities from Numpy, itertools, scikit-learn, gensim, pandas.

7. EVALUATION AND RESULT

In this section we present, the evaluation our baseline models: CNN-rand and CNN-static. To measure the performance and prove the effectiveness of our models, we evaluate each model by test accuracy and also use well-known metrics precision, recall, and F-measure. The Precision represents the proportion of predicted correct labels to the total number of actual labels, averages over all instances. The recall is the proportion of correct labels to the number of predicted labels, averages over all instances. And the F-measure is the harmonic mean of precision and recall.

7.1. FIRST EVALUATION AND RESULT

In the first model, we conducted a number of test-runs for both the domains. We did not implement any cross-validation but Cnn_rand.py over and over to get a better accuracy. Table 5 and 5-1 show CNN-rand result for restaurants and laptops and Table 6 shows the test accuracy results.

	Precision	Recall	F-measure	support
Positive	0.75	0.97	0.85	80
Negative	0.80	0.24	0.36	34

Table 5. CNN-rand result for restaurants

	Precision	Recall	F-measure	Support
Positive	0.79	0.78	0.78	81
Negative	0.69	0.70	0.70	57

Table 5-1.CNN-rand result for laptops

	Restaurant	**Laptop**
CNN-rand	**75.44**	74.64

Table 6. Best test Accuracy obtain with CNN-rand

7.2. SECOND EVALUATION AND RESULT

In the second model, multiple of test-runs have also been conducted in both domains. We use the pre-trained word embedding frWac2Vec and L2-normalization of weights on penultimate layer wasn't implemented but provides an L2-normalization of gradients. Table 7 and 7-1 show CNN-static result for restaurants and laptops and table 8 shows the test accuracy results for CNN-static.

	Precision	Recall	F-measure	Support
Positive	0.84	0.94	0.89	82
Negative	0.77	0.53	0.63	32

Table 7. CNN-static result for restaurants

	Precision	Recall	F-measure	Support
Positive	0.78	0.91	0.84	73
Negative	0.86	0.70	0.77	63

Table 7-1. CNN-static result for laptop

	Restaurant	**Laptop**
CNN-static	**82.46**	**81.16**

Table 8.test Accuracy obtained with CNN-static

8. COMPARISON WITH ABSA2015 RESULT

In SemEval2015 (task12), the sentiment polarity classification has involved 12 teams for the restaurants and 10 teams for the laptops. The best accuracy obtained in laptop domain was 79.34% while for the restaurant domain was 78.69%. Respect to the restaurant domains, the Sentiue team used a MaxEnt classifier along with features based on n-grams, POS tagging, lemmatization, negation words and publicly available sentiment lexica (MPQA, Bing Liu's lexicon, AFINN). While the ECNU team's system has obtained an accuracy of 78.29% for laptops and 78.10% for restaurants by using features based on n-grams, PMI scores, POS tags, parse trees, negation words, and scores based on 7 sentiment lexica. And the lsislif team's system based on a logistic regression model (Liblinear) with various features: syntactic (e.g., unigrams, negation), semantic (Brown dictionary), sentiment (e.g., MPQA, SentiWordnet) has scored 77.87% of accuracy for laptops and 75.50% for the restaurants. Table 9 shows the result obtains by each team and our models in both domains. [53]

Laptop		Restaurants	
Team	Accuracy	Team	Accuracy
CNNstatic	82.46	CNNstatic	81.16
Sentiue	79.34	Sentiue	78.69
ECNU	78.29	ECNU	78.10
Lsislif	77.87	Lsislif	75.50
CNNrand	75.44	LT3	75.02
ECNU	74.49	CNNrand	74.64
LT3	73.76	UFRGS	71.71
TJUdeM	73.23	Wnlp	71.36
EliXa	72.91	UMDuluthC	71.12
Wnlp	72.07	EliXa	70.05
EliXa	71.54	ECNU	69.82
V3	68.38	V3	69.46
UFRGS	67.33	TJUdeM	68.87
SINAI	65.85	EliXa	67.33
SINAI	51.84	SINAI	60.71
		SIEL	70.76
SVM+	69.96	SVM+ BOW	63.55

BOW Baseline		Baseline	
Majority Baseline	57.00	Majority Baseline	53.72

Table 9. The results of our CNN models against SemEval2015 results (continued)

9. RESULTS AND DISCUSSION

Although the size of our corpora was small, our CNN-rand model has surprisingly performed well on its own as well as our CNN-static. The decision regarding the test accuracy for these models was made after running the python Cnn_rand.py and Cnn_static.py(see Appendix 5) over and over and modified some parameters such as the embedding size, the number of filters, learning rate, and the dropout. In CNN-rand model, the best test accuracy was obtained after changing the dropout from 0.5 to 0.3, learning rate from 0.005 to 0.03, the number of feature maps from 100 to 50; and for CNN-static we have to change the dropout 0.03 to 0.5. Based on our observation these three parameters had an effect on our models and while implementing it. We have also observed that the test accuracy, in both models, did perform well with the laptop corpus than the restaurant corpora. As you can see in Table 9 the test accuracy for the laptop is 82.46 % with CNN-static and 75.44% with CNN-rand while for the restaurants the accuracy for CNN-static was 81.16% and 74.64% for CNN-rand.This might be to same fact observation reported to the original paper[50] that the laptops domain has performed better over the restaurant domain for all teams specifically in slot 3 since the restaurant domain had more frequent the positive polarity in training data than in the test data which could lead to biased models. Since our training data to use in our model is similar to the same corpus. This might be the reason we have got such accuracy in that particular domain.

10. CONCLUSION

In this work, we present a deep learning approach for sentimental analysis applied on French text. The goal of this proposal is to use neural network model as a classifier to categorize the sentiment polarity of a given French text. The model is based on Convolutional Neural Network model for text classification on Mxnet and it builds based on the convolutional architecture described by Yoon Kim since it has achieved good classification performance across a range of text classification tasks. The choice of this particular neural network was made for its ability claimed by many researchers to perform for text classification and for the framework we selected Mxnet for its memory-efficient feature. We used one-dimensional Convolution Neural Network to build a classifier for detecting sentiment expressed in a sentence; the evaluation was done in two models and has obtained encouraging results. With CNN-rand and CNN-static; with respect to CNN-rand we obtained 75.44% and 74.64% of test accuracy. And CNN-static we obtained 82.46% and 81.16%. It can be noticed that the CNN-static has performed better than the CNN-rand due to the word embedding used. Therefore, with this CNN-static result we suggest that the pretrained vectors are good for the performance gains. Hence these results have shown that; the deep learning model (Convolutional Neural Network) outperformed traditional models and can be used for sentence classification in French and other languages.

As future work, we are considering to explore other neural network models such as Recursive Neural Tensor Networks (RNTN) and Long Short-Term Memory (LSTM). Also, we plan to evaluate other deep learning frameworks such as Keras, Tensorflow, deeplearning4j, and Pytorch. Finally, we have considered applying our approach to other languages and evaluate other word embedding features.

11. REFERENCES

1. B. Liu, “Sentiment Analysis and Opinion Mining,” Synthesis Lectures on Human Language Technologies, vol. 5, no. 1, pp. 1–167, 2012.

2. M. Cliche, “BB_twtr at SemEval-2017 Task 4: Twitter Sentiment Analysis with CNNs and LSTMs,” Apr. 2017.

3. A. Severyn and A. Moschitti, “UNITN: training deep convolutional neural network for twitter sentiment classification,” in Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval '15), pp. 464–469, 2015.

4. Hatem Ghorbel and David Jacot,” Sentiment Analysis of French Movie Reviews” in Advances in Intelligent and Soft Computing book series (AINSC, volume 86)

5. M. del Pilar Salas-Zárate, M. A. Paredes-Valverde, J. Limon-Romero, D. Tlapa, and Y. Baez-Lopez, “Sentiment classification of Spanish reviews: an approach based on feature selection and machine learning methods,” Journal of Universal Computer Science, vol. 22, no. 5, pp. 691–708, 2016

6. P. Smith and M. Lee, “Cross-discourse development of supervised sentiment analysis in the clinical domain,” in Proceedings of the 3rd Workshop in Computational Approaches to Subjectivity and Sentiment Analysis, pp. 79–83, 2012.

7. I. Habernal, T. Ptáček, and J. Steinberger, “Reprint of “supervised sentiment analysis in Czech social media”,” Information Processing and Management, vol. 51, no. 4, pp. 532–546, 2015

8. N. F. F. Da Silva, E. R. Hruschka, and E. R. Hruschka, “Tweet sentiment analysis with classifier ensembles,” Decision Support Systems, vol. 66, pp. 170–179, 2014

9. E. Fersini, E. Messina, and F. A. Pozzi, “Sentiment analysis: bayesian ensemble learning,” Decision Support Systems, vol. 68, pp. 26–38, 2014.

10. B. Agarwal, S. Poria, N. Mittal, A. Gelbukh, and A. Hussain, “Concept level sentiment analysis using dependency-based semantic parsing: a novel approach,” Cognitive Computation, vol. 7, no. 4, pp. 487–499, 2015

11. https://www.hindawi.com/journals/sp/2017/1329281/

12. Shi H-X, Li X-J. A sentiment analysis model for hotel reviews based on supervised learning. In: 2011 international conference on machine learning and cybernetics (ICMLC). IEEE; 2011, p. 950–54.

13. Boiy E, Moens M-F. A machine learning approach to sentiment analysis in multilingual Web texts. Inf. Retr. 2009;12:526–58.

14. Singh VK, Piryani R, Uddin A, Waila P, et al. Sentiment analysis of textual reviews; Evaluating machine learning, unsupervised and SentiWordNet approaches. In: 2013 5th international conference on knowledge and smart technology (KST). IEEE; 2013, p. 122–27.

15. Habernal I, Ptácek T, Steinberger J. Sentiment analysis in Czech social media using supervised machine learning. In: Proceedings of the 4th workshop on computational approaches to subjectivity, sentiment and social media analysis. 2013, p. 65–74.

16. Tan S, Zhang J. An empirical study of sentiment analysis for Chinese documents. Expert Syst Appl. 2008;34:2622–9.

17. Zagibalov T, Carroll J. Automatic seed word selection for unsupervised sentiment classification of Chinese text. In: Proceedings of the 22nd international conference on computational linguistics, vol. 1. Association for Computational Linguistics; 2008, p. 1073–80

18. Balahur A, Turchi M. Improving sentiment analysis in twitter using multilingual machine translated data. In: RANLP; 2013, p. 49–55.

19. Zhu S, Xu B, Zheng D, Zhao T. Chinese microblog sentiment analysis based on semi-supervised learning. In: Semantic web and web science. New York: Springer; 2013, p. 325–31.

20. Mahyoub FHH, Siddiqui MA, Dahab MY. Building an Arabic sentiment lexicon using semi-supervised learning. J King Saud Univ Comput Inf Sci. 2014;26(4):417–24.

21. Al-Ayyoub M, Essa SB, Alsmadi I. Lexicon-based sentiment analysis of arabic tweets. Int J Soc Netw Min. 2015;2:101–14.

22. Mizumoto K, Yanagimoto H, Yoshioka M. Sentiment analysis of stock market news with semi-supervised learning. In: 2012 IEEE/ACIS 11th international conference on computer and information science (ICIS). IEEE, 2012; p. 325–28.

23. Ghorbel H, Jacot D. Further experiments in sentiment analysis of french movie reviews. In: Advances in Intelligent Web Mastering–3. Berlin, Heidelberg: Springer; 2011, p. 19–28.

24. Tao Chena, Ruifeng Xua,Yulan Hec, Xuan Wanga " Improving sentiment analysis via sentence type classification using BiLSTM-CRF and CNN",https://www.sciencedirect.com/science/article/pii/S0957417416305929,9 November 2016.

25. C. N. Dos Santos and M. Gatti, "Deep convolutional neural networks for sentiment analysis of short texts," in Proceedings of the 25th International Conference on Computational Linguistics (COLING '14), pp. 69–78, 2014.

26. Z. Hu, J. Hu, W. Ding, and X. Zheng, "Review Sentiment Analysis Based on Deep Learning," in Proceedings of the 12th IEEE International Conference on E-Business Engineering (ICEBE '15), pp. 87–94, 2015.

27. S. Ruder, P. Ghaffari, and J. G. Breslin, "INSIGHT-1 at SemEval-2016 Task 5: Deep Learning for Multilingual Aspect-based Sentiment Analysis," in Proceedings of the 10th

International Workshop on Semantic Evaluation (SemEval '16), pp. 330–336, San Diego, CA, USA, June 2016.
28. A. Severyn and A. Moschitti, "UNITN: training deep convolutional neural network for twitter sentiment classification," in Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval '15), pp. 464–469, 2015.
29. X. Sun, C. Li, and F. Ren, "Sentiment analysis for Chinese microblog based on deep neural networks with convolutional extension features," Neurocomputing, vol. 210, pp. 227–236, 2016.
30. Y. Kim,"Convolutional neural networks for sentence classification", Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP) (2014), pp. 1746-1751
31. https://www.mathworks.com/discovery/machine-learning.html
32. https://www.mathworks.com/discovery/unsupervised-learning.html
33. David Fumo," Types of Machine Learning Algorithms You Should Know",https://towardsdatascience.com/types-of-machine-learning-algorithms-you-should-know 953a08248861, Jun 15, 2017
34." Deep learning",https://en.wikipedia.org/wiki/Deep_learning18 May 2018
35. Nathan Hubens," Deep inside: Autoencoders", https://towardsdatascience.com/deep-inside-autoencoders-7e41f319999f , Feb 25
36. Adam Gibson, Josh Patterson,"DeepLearning" ,https://www.safaribooksonline.com/library/view/deep-learning/9781491924570/ch04.html, 2018
37. Olga Davydova," 7 types of Artificial Neural Networks for Natural Language Processing", https://medium.com/@datamonsters/artificial-neural-networks-for-natural-language-processing-part-1-64ca9ebfa3b2, Sep 26, 2017
38. Peilu Wang, Yao Qian, Frank K. Soong, Lei He, Hai Zhao, " Part-of-Speech Tagging with Bidirectional Long Short-Term Memory Recurrent Neural Network" https://arxiv.org/abs/1510.06168, 21 Oct 2015
39. Apache MXNet"https://en.wikipedia.org/wiki/Apache_MXNet , 8 May 2018
40. https://mxnet.incubator.apache.org/architecture/overview.html
41. Jason Brownlee," What Are Word Embeddings for Text?",https://machinelearningmastery.com/what-are-word-embeddings/,October 11, 2017
42. " Word2vec" https://en.wikipedia.org/wiki/Word2vec, 14 April 2018
43. Jeffrey Pennington, Richard Socher, Christopher D. Manning," GloVe: Global Vectors for Word Representation",https://nlp.stanford.edu/projects/glove/, August 2014
44. Jeffrey Pennington, Richard Socher, Christopher D. Manning," GloVe: Global Vectors for Word Representation", https://nlp.stanford.edu/pubs/glove.pdf
45. Martinpella," Neural machine translation with seq2seq model",https://github.com/martinpella/lang-translator , Mar 9,2018
46. " Convolutional neural network",https://en.wikipedia.org/wiki/Convolutional_neural_network, 10 May 2018
47. Karpathy," Convolutional Neural Networks for visual recognition", http://cs231n.github.io/convolutional-networks/
48. Xiang Zhang ,Junbo Zhao ,Yann LeCun," Character-level Convolutional Networks for Text Classification"https://papers.nips.cc/paper/5782-character-level-convolutional-networks-for-text-classification.pdf

49. Adit Deshpande,“ A Beginner's Guide To Understanding Convolutional Neural Networks“ https://adeshpande3.github.io/A-Beginner%27s-Guide-To-Understanding-Convolutional-Neural-Networks/
(52. https://arxiv.org/pdf/1408.5882.pdf)
50. Maria Pontiki, Dimitrios Galanis, Haris Papageorgiou,Suresh Manandhar, Ion Androutsopoulos,“ SemEval-2015 Task 12: Aspect Based Sentiment Analysis“, http://www.aclweb.org/anthology/S15-2082 ,Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015), pages 486–495
51. Ye Zhang, Byron Wallace,“ A Sensitivity Analysis of (and Practitioners' Guide to) Convolutional Neural Networks for Sentence Classification“ ,https://arxiv.org/abs/1510.03820, 6 Apr 2016
52. “ Text Classification Using a Convolutional Neural Network on MXNet“http://mxnet.apache.org/tutorials/nlp/cnn.html
53. Denny Britz,“ Implementing a CNN for Text Classification in TensorFlow“,http://www.wildml.com/2015/12/implementing-a-cnn-for-text-classification-in-tensorflow/, December 11, 2015

Research Project no1. A COMPARISON OF VARIOUS PROCESS MODELLING TECHNIQUES FOR SOFTWARE DEVELOPMENT: A CASE STUDY OF WEB APPLICATIONS.

In this research project, we have analyzed in different perceptive the model techniques methodologies and requirements engineering for web application projects. The goal was to analyze web application process model techniques and propose the best-fit methodology for web application development. The result of analysis has shown that there is still much research to do in this field and such research should be of immense help to organizations and individuals in their endeavor to engineer the best web applications. This is because there is no single methodology that can support all the best features development of web application. Since the domain of requirements engineering for web application projects clearly presents enormous opportunities for researchers in this area that can lead to an optimal methodology that can address the maximum level of detail in this crucial area

Research Project no2. CLOUD BASED TESTING TECHNIQUES FOR WEB APPLICATION: NEED , CHALLENGE AND SCENARIO.

The goal of this research paper was to give a broad overview of cloud-based testing. The major contributions of this paper included an intelligent discussion about cloud testing in terms of its requirements, tools, challenges, and Scenario. The analysis of the paper was focused more on the security challenge because of its failure which might cause significant damage to customer's private data, information and among other things. This challenge was appearing to be a critical issue when it comes to testing in the cloud. Our suggestion to overcome the security challenge was an encrypted system since this will help the organization to protect their data while testing their product in the cloud. We have also found that due to the new arise of this topic among researchers advance research need to be done in this field by addressing address the open challenges in cloud testing.

Research Project no3: THE IMPLEMENTATION OF MULTILINGUAL SENTIMENT ANALYSIS: APPROACHES AND TOOLS.

In this research project, the goal was to give an overview of multilingual sentiment analysis since the majority of research efforts are devoted to English-language data, while a great share of information is available in other languages. We have analyzed the various current approaches and tools used; identify challenges of approaching multilingual sentimental analysis based on different research done. After analysis, we have found that the main problem of multilingual sentiment analysis is the lack of lexical resources. The future proposes work in a paper is to generate corpora in French from English corpus using the SMT system (such as Google Translate, Bing Translator, etc.) to evaluate the sentiment expressed and compare our result with other results in the same field using our selected language. The result of the experiment will help us to recommend an approach to follow for the sentimental Analysis in the French Language.

5.2. PREPROCESSING

The preprocessing of our datasets was done from data_helpers.py . the folder was called from the main class of both the models : Cnn_rand.py and Cnn_static.py. The data_helpers.py code for preprocessing our data look like this :

```
from collections import Counter
import numpy as np
import re
import os


def open_file(filename, mode='r'):
    """
    Commonly used file reader and writer, change this to switch between python2 and python3.
    :param filename: filename
    :param mode: 'r' and 'w' for read and write respectively
    """
    return open(filename, mode, encoding='utf-8', errors='ignore')


def clean_str(string):
    """
    Tokenization/string cleaning for all datasets except for SST.
    Original taken from https://github.com/yoonkim/CNN_sentence/blob/master/process_data.py
    """
    string = re.sub(r"[^A-Za-z0-9(),!?\'\`]", " ", string)
    string = re.sub(r"\'s", " \'s", string)
    string = re.sub(r"\'ve", " \'ve", string)
    string = re.sub(r"n\'t", " n\'t", string)
    string = re.sub(r"\'re", " \'re", string)
    string = re.sub(r"\'d", " \'d", string)
    string = re.sub(r"\'ll", " \'ll", string)
    string = re.sub(r",", " , ", string)
    string = re.sub(r"!", " ! ", string)
    string = re.sub(r"\(", " \( ", string)
    string = re.sub(r"\)", " \) ", string)
    string = re.sub(r"\?", " \? ", string)
    string = re.sub(r"\s{2,}", " ", string)
    return string.strip().lower()

def build_vocab(data, vocab_dir, vocab_size=8000):
    """
    Build vocabulary file from training data.
    """
    print('Building vocabulary...')

    all_data = []  # group all data
    for content in data:
        all_data.extend(content.split())

    counter = Counter(all_data)  # count and get the most common words
    count_pairs = counter.most_common(vocab_size - 1)
    words, _ = list(zip(*count_pairs))

    words = ['<PAD>'] + list(words)  # add a padding with id 0 to pad the sentence to same length
    open_file(vocab_dir, 'w').write('\n'.join(words) + '\n')


def read_vocab(vocab_file):
    """
    Read vocabulary from file.
    """
    words = open_file(vocab_file).read().strip().split('\n')
    word_to_id = dict(zip(words, range(len(words))))
    return words, word_to_id


def process_text(text, word_to_id, max_length, clean=True):
    """tokenizing and padding"""
    if clean:  # if the data needs to be cleaned
        text = clean_str(text)
    text = text.split()

    text = [word_to_id[x] for x in text if x in word_to_id]
    if len(text) < max_length:
        text = [0] * (max_length - len(text)) + text
    return text[:max_length]


class Corpus(object):
    """
    Preprocessing training data.
    """

    def __init__(self, pos_file, neg_file, vocab_file, dev_split=0.1, max_length=50,
vocab_size=8000):
        # loading data
        pos_examples = [clean_str(s.strip()) for s in open_file(pos_file)]
        neg_examples = [clean_str(s.strip()) for s in open_file(neg_file)]
        x_data = pos_examples + neg_examples
        y_data = [0.] * len(pos_examples) + [1.] * len(neg_examples)  # 0 for pos and 1 for neg

        if not os.path.exists(vocab_file):
            build_vocab(x_data, vocab_file, vocab_size)

        self.words, self.word_to_id = read_vocab(vocab_file)

        for i in range(len(x_data)):  # tokenizing and padding
            x_data[i] = process_text(x_data[i], self.word_to_id, max_length, clean=False)

        x_data = np.array(x_data)
        y_data = np.array(y_data)

        # shuffle
        indices = np.random.permutation(np.arange(len(x_data)))
        x_data = x_data[indices]
        y_data = y_data[indices]

        # train/dev split
        num_train = int((1 - dev_split) * len(x_data))
        self.x_train = x_data[:num_train]
        self.y_train = y_data[:num_train]
        self.x_test = x_data[num_train:]
        self.y_test = y_data[num_train:]

    def __str__(self):
        return 'Training: {}, Testing: {}, Vocabulary: {}'.format(len(self.x_train), len(self.x_test),
len(self.words))
```

6.2. CNN CONFIGURATION

The configuration of our models was the same but with difference was regarding the word embeddings. For both the models the same data_helper,py was used for preprocessing.For each model code we have imported Corpus, read_vocab, process_text from the Data_helpers.The python code for both models was as followed:

For Cnn_rand :

```
base_dir = 'data/resto'
pos_file = os.path.join(base_dir, 'resto.pos.txt')
neg_file = os.path.join(base_dir, 'resto.neg.txt')
vocab_file = os.path.join(base_dir,‘resto.vocab.txt‘)

save_path = 'checkpoints'  # model save path
```

```
if not os.path.exists(save_path):
    os.mkdir(save_path)
model_file = os.path.join(save_path, 'cnn_rand.params')
class TCNNConfig(object):
    """
    CNN parameters
    """
    embedding_dim = 128  # embedding vector size
    seq_length = 50  # maximum length of sequence
    vocab_size = 8000  # most common words

    num_filters = 50  # number of the convolution filters (feature maps)
    kernel_sizes = [3, 4, 5]  # three kinds of kernels (windows)

    dropout_prob = 0.2 # dropout rate
    learning_rate = 0.02  # learning rate
    batch_size = 50  # batch size for training
    num_epochs = 10  # total number of epochs

    num_classes = 2  # number of classes

    dev_split = 0.1  # percentage of dev data


class Conv_Max_Pooling(nn.Block):
    """
    Integration of Conv1D and GlobalMaxPool1D layers
    """

    def __init__(self, channels, kernel_size, **kwargs):
        super(Conv_Max_Pooling, self).__init__(**kwargs)

        with self.name_scope():
            self.conv = nn.Conv1D(channels, kernel_size)
            self.pooling = nn.GlobalMaxPool1D()

    def forward(self, x):
        output = self.pooling(self.conv(x))
        return nd.relu(output).flatten()


class TextCNN(nn.Block):
    """
    CNN text classification model
    """
    def __init__(self, config, **kwargs):
        super(TextCNN, self).__init__(**kwargs)

        V = config.vocab_size
        E = config.embedding_dim
        Nf = config.num_filters
        Ks = config.kernel_sizes
        C = config.num_classes
        Dr = config.dropout_prob

        with self.name_scope():
            self.embedding = nn.Embedding(V, E)  # embedding layer

            # three different convolutional layers
            self.conv1 = Conv_Max_Pooling(Nf, Ks[0])
            self.conv2 = Conv_Max_Pooling(Nf, Ks[1])
            self.conv3 = Conv_Max_Pooling(Nf, Ks[2])
            self.dropout = nn.Dropout(Dr)  # a dropout layer
            self.fc1 = nn.Dense(C)  # a dense layer for classification

    def forward(self, x):
        x = self.embedding(x).transpose((0, 2, 1))  # Conv1D takes in NCW as input
        o1, o2, o3 = self.conv1(x), self.conv2(x), self.conv3(x)
        outputs = self.fc1(self.dropout(nd.concat(o1, o2, o3)))

        return outputs


def get_time_dif(start_time):
    """
    Return the time used since start_time.
    """
    end_time = time.time()
    time_dif = end_time - start_time
    return timedelta(seconds=int(round(time_dif)))


class Dataset(Dataset):
    """
    An implementation of the Abstracted gluon.data.Dataset, used for loading data in batch
    """

    def __init__(self, x, y):
        super(Dataset, self).__init__()
        self.x = x
        self.y = y
    def __getitem__(self, index):
        return self.x[index].astype(np.float32), self.y[index].astype(np.float32)

    def __len__(self):
        return len(self.x)
```

For Cnn_static:

```
base_dir = 'data/resto'
pos_file = os.path.join(base_dir, 'resto.pos.txt')
neg_file = os.path.join(base_dir, 'resto.neg.txt')
vocab_file = os.path.join(base_dir, frWac2Vec.bin")

save_path = 'checkpoints'  # model save path
if not os.path.exists(save_path):
    os.mkdir(save_path)
model_file = os.path.join(save_path, 'cnn_static.params')


def try_gpu():
    """If GPU is available, return mx.gpu(0); else return mx.cpu()"""
    try:
        ctx = mx.gpu()
        _ = nd.array([0], ctx=ctx)
    except:
        ctx = mx.cpu()
    return ctx

class TCNNConfig(object):
    """
    CNN parameters
    """
    embedding_dim = 128  # embedding vector size
    seq_length = 50  # maximum length of sequence

    num_filters = 50  # number of the convolution filters (feature maps)
    kernel_sizes = [3, 4, 5]  # three kinds of kernels (windows)

    dropout_prob = 0.5  # dropout rate
    learning_rate = 0.01  # learning rate
    batch_size = 50  # batch size for training
    num_epochs = 10  # total number of epochs
    num_classes = 2  # number of classes

    dev_split = 0.1  # percentage of dev data


class Conv_Max_Pooling(nn.Block):
    """
    Integration of Conv1D and GlobalMaxPool1D layers
    """

    def __init__(self, channels, kernel_size, **kwargs):
        super(Conv_Max_Pooling, self).__init__(**kwargs)

        with self.name_scope():
            self.conv = nn.Conv1D(channels, kernel_size)
            self.pooling = nn.GlobalMaxPool1D()

    def forward(self, x):
        output = self.pooling(self.conv(x))
        return nd.relu(output).flatten()


class TextCNN(nn.Block):
    """
    CNN text classification model, based on the paper.
    """

    def __init__(self, config, **kwargs):
        super(TextCNN, self).__init__(**kwargs)

        V = config.vocab_size
        E = config.embedding_dim
        Nf = config.num_filters
        Ks = config.kernel_sizes
        C = config.num_classes
        Dr = config.dropout_prob

        with self.name_scope():
            self.embedding = nn.Embedding(V, E)  # embedding layer

            # three different convolutional layers
            self.conv1 = Conv_Max_Pooling(Nf, Ks[0])
            self.conv2 = Conv_Max_Pooling(Nf, Ks[1])
            self.conv3 = Conv_Max_Pooling(Nf, Ks[2])
            self.dropout = nn.Dropout(Dr)  # a dropout layer
        self.fc1 = nn.Dense(C)  # a dense layer for classification

    def forward(self, x):
        x = self.embedding(x).transpose((0, 2, 1))  # Conv1D takes in NCW as input
        o1, o2, o3 = self.conv1(x), self.conv2(x), self.conv3(x)
        outputs = self.fc1(self.dropout(nd.concat(o1, o2, o3)))

        return outputs


def get_time_dif(start_time):
    """
    Return the time used since start_time.
    """
    end_time = time.time()
    time_dif = end_time - start_time
    return timedelta(seconds=int(round(time_dif)))


class Dataset(Dataset):
    """
    An implementation of the Abstracted gluon.data.Dataset, used for loading data in batch
    """

    def __init__(self, x, y):
        super(Dataset, self).__init__()
        self.x = x
        self.y = y

    def __getitem__(self, index):
        return self.x[index].astype(np.float32), self.y[index].astype(np.float32)

    def __len__(self):
        return len(self.x)
```