Presented By: Mokhtar Ahmed, Mohamed Magdy

Arabic Sentiment Analysis Using Deep learning

supervised By: Dr. Samhaa R. El-Beltagy

* **Abstract**

Opinion mining task “detecting the people opinion about something” is still an open research area especially in Arabic language. Recently, neural network models have been used to learn semantic representations for NLP tasks (Le and Mikolov, 2014; Tang et al., 2015), achieving highly competitive results and we trying to get the advantages from those techniques in the opinion mining task. For example distributed word representations “Word Embedding” (Mikolov et al., 2013) have been used as the basic building block by most models for NLP. Numerous methods have been proposed to learn representations of **phrases and larger text segments** from distributed **word representations.**

Convolutional neural networks have been widely used for semantic composition automatically capturing n-gram information. Sequential models such as recurrent neural network or long short-term memory (**LSTM**) (Li et al., 2015a; Tang et al., 2015) have also been used for recurrent semantic composition. The attention mechanism was first proposed in machine translation (Bahdanau et al., 2014). We explore CNN and recurrent neural networks with attention mechanism to learn document representation for detecting opinion mining in Arabic language.

* **Introduction**

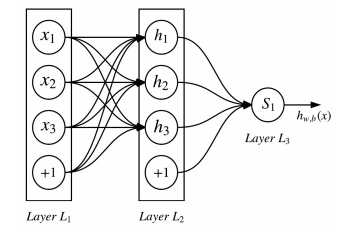
Sentiment analysis or opinion mining is the computational study of people’s opinions, sentiments, emotions, appraisals, and attitudes towards entities such as products, services, organizations, individuals, issues, events, topics, and their attributes. The inception and rapid growth of the field coincide with those of the social media on the Web, for example, reviews, forum discussions, blogs, micro-blogs, Twitter, and social networks, because for the first time in human history, we have a huge volume of opinionated data recorded in digital forms. Since early 2000, sentiment analysis has grown to be one of the most active research areas in natural language processing (NLP). It is also widely studied in data mining, Web mining, text mining, and information retrieval. In fact, it has spread from computer science to management sciences and social sciences such as marketing, finance, political science, communications, health science, and even history, due to its importance to business and society as a whole. This proliferation is due to the fact that opinions are central to almost all human activities and are key influencers of our behaviors. Our beliefs and perceptions of reality, and the choices we make, are, to a considerable degree, conditioned upon how others see and evaluate the world. For this reason, whenever we need to make a decision we often seek out the opinions of others. This is not only true for individuals but also true for organizations. Nowadays, if one wants to buy a consumer product, one is no longer limited to asking one’s friends and family for opinions because there are many user reviews and discussions about the product in public forums on the Web. For an organization, it may no longer be necessary to conduct surveys, opinion polls, and focus groups in order to gather public opinions because there is an abundance of such information publicly available. In recent years, we have witnessed that opinionated postings in social media have helped reshape businesses, and sway public sentiments and emotions, which have profoundly impacted on our social and political systems. Such postings have also mobilized masses for political changes such as those happened in some Arab countries in 2011. It has thus become a necessity to collect and study opinions. However, finding and monitoring opinion sites on the Web and distilling the information contained in them remains a formidable task because of the proliferation of diverse sites. Each site typically contains a huge volume of opinion text that is not always easily deciphered in long blogs and forum postings. The average human reader will have difficulty identifying relevant sites and extracting and summarizing the opinions in them. Automated sentiment analysis systems are thus needed. Because of this, there are many start-ups focusing on providing sentiment analysis services. Many big corporations have also built their own in-house capabilities. These practical applications and industrial interests have provided strong motivations for research in sentiment analysis. Existing research has produced numerous techniques for various tasks of sentiment analysis, which include both supervised and unsupervised methods. In the supervised setting, early papers used all types of supervised machine learning methods (such as Support Vector Machines (SVM), Maximum Entropy, Naïve Bayes, etc.) And feature combinations. Unsupervised methods include various methods that exploit sentiment lexicons, grammatical analysis, and syntactic patterns. Several survey books and papers have been published, which cover those early methods and applications extensively. Since about a decade ago, deep learning has emerged as a powerful machine learning technique and produced state-of-the-art results in many application domains, ranging from computer vision and speech recognition to NLP. Applying deep learning to sentiment analysis has also become very popular recently. This paper first gives an overview of deep learning and then provides a comprehensive survey of the sentiment analysis research based on deep learning.

* **Background**
  + **Deep Learning**

The research community lost interests in neural networks in late 1990s mainly because they were regarded as only practical for **shallow** neural networks (neural networks with one or two layers) as training a **deep** neural network (neural networks with more layers) is complicated and computationally very expensive. However, in the past 10 years, deep learning made breakthrough and produced state-of-the-art results in many application domains, starting from computer vision, then speech recognition, and more recently, NLP. The renaissance of neural networks can be attributed to many factors. Most important ones include:

1. The availability of computing power due to the advances in hardware (e.g., GPUs)
2. The availability of huge amounts of training data.
3. The power and flexibility of learning intermediate representations.

In a nutshell, deep learning uses a cascade of multiple layers of nonlinear processing units for feature extraction and transformation. The lower layers close to the data input learn simple features, while higher layers learn more complex features derived from lower layer features. The architecture forms a hierarchical and powerful feature representation.

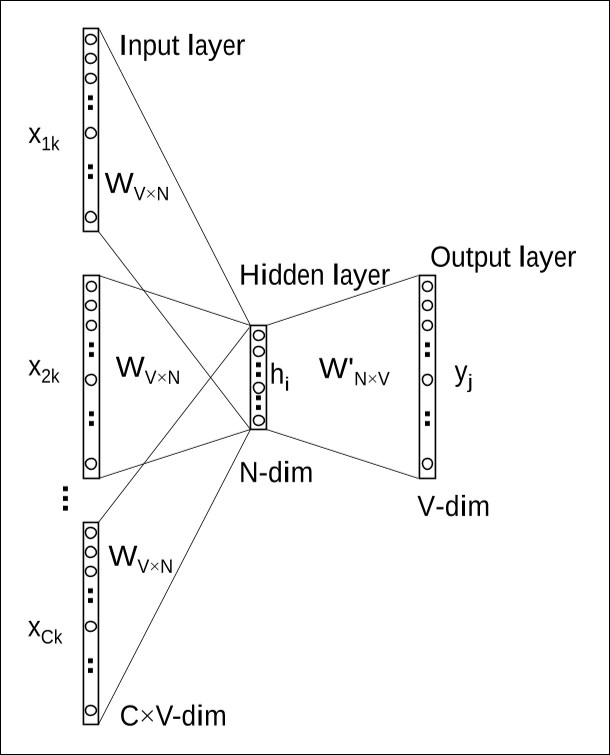
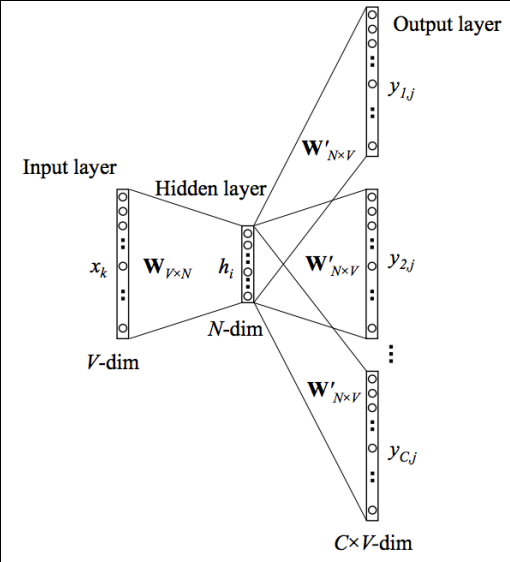


**Figure1: Feed forward neural network**

* **WORD EMBEDDING**

Many deep learning models in NLP need word embedding results as input features. Word embedding is a technique for language modelling and feature learning, which transforms words in a vocabulary to vectors of continuous real numbers (e.g., 𝐵𝐵 "ℎ𝐵"→ (…,0.15,…,0.23,…,0.41,…)). The technique normally involves a mathematic embedding from a high-dimensional sparse vector space (e.g., one-hot encoding vector space, in which each word takes a dimension) to a lower-dimensional dense vector space. Each dimension of the embedding vector represents a latent feature of a word. The vectors may encode linguistic regularities and patterns. The learning of word embeddings can be done using neural networks or matrix factorization.

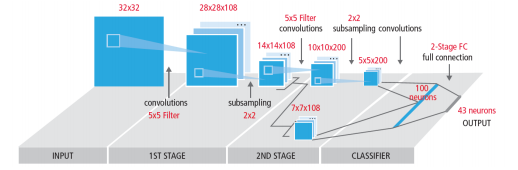
One commonly used word embedding system is Word2Vec, which is essentially a computationally efficient neural network prediction model that learns word embeddings from text. It contains Continuous Bag-of-Words model (CBOW), and Skip-Gram model (SG). The CBOW model predicts the target word (e.g., “wearing”) from its context words (“the boy is \_ a hat”, where “\_” denotes the target word), while the SG model does the inverse, predicting the context words given the target word. Statistically, the CBOW model smoothens over a great deal of distributional information by treating the entire context as one observation. It is effective for smaller datasets. However, the SG model treats each context-target pair as a new observation and is better for larger datasets. Another frequently used learning approach is Global Vector (GloVe), which is trained on the nonzero entries of a global word-word co-occurrence matrix.



**Figure 2: Skip Gram Model Figure 3: CBOW Model**

* **Convolutional Neural Network**

Convolutional Neural Network (CNN) is a special type of feed forward neural network originally employed in the field of computer vision. Its design is inspired by the human visual cortex, a visual mechanism in animal brain. The visual cortex contains a lot of cells that are responsible for detecting light in small and overlapping sub-regions of the visual fields, which are called receptive fields. These cells act as local filters over the input space. CNN consists of multiple convolutional layers, each of which performs the function that is processed by the cells in the visual cortex. Figure 4 shows a CNN for recognizing traffic signs. The input is a 32x32x1 pixel image (32 x 32) represents image width x height; 1 represents input channel). In this first stage, the filter (size 5x5x1) is used to scan the image. Each region in the input image that the filter projects on is a receptive field. The filter is actually an array of numbers (called weights or parameters). As the filter is sliding (or convolving), it is multiplying its weight values with the original pixel values of the image (element wise multiplications). The multiplications are all summed up to a single number, which is a representative of the receptive field. Every receptive field produces a number. After the filter finishes scanning over the image, we can get an array (size 28x28x1), which is called the activation map or feature map. In CNN, we need to use different filters to scan the input. In Figure 4, we apply 108 kinds of filters and thus have 108 stacked feature maps in the first stage, which consists of the first convolutional layer. Following the convolutional layer, a subsampling (or pooling) layer is usually used to progressively reduce the spatial size of the representation, thus to reduce the number of features and the computational complexity of the network. For example, after subsampling in the first stage, the convolutional layer reduces its dimensions to (14x14x108). Note that while the dimensionality of each feature map is reduced, the subsampling step retains the most important information, with a commonly used subsampling operation being the max pooling. Afterwards, the output from the first stage becomes input to the second stage and the new filters are employed. The new filter size is 5x5x108, where 108 is the feature map size of the last layer. After the second stage, CNN uses a fully connected layer and then a softmax readout layer with output classes for classification.



**Figure 4: Convolutional Neural Network**

* **Problem Definition**

Arabic is one of the six official languages of the United Nations. It is the official language of 27 countries and is spoken by more than 422 million people in the Arab world. On the web, Arabic is ranked the fourth mostly used language and the fastest growing during the last five years with a growth rate of 6091.9% in the number of Internet users.

Arabic has three main varieties: Classical Arabic; which is the language of the Qur’an (Islam’s Holy Book); Modern Standard Arabic (MSA) and dialectical Arabic. MSA the most eloquent Arabic language variety used in writing and in most formal speech. Dialectical or colloquial Arabic refers to all oral varieties spoken in daily communication. These vary from one Arab country to another and from one region of the same country to another.

Unlike Latin languages, Arabic is written from right to left and is distinguished by the absence upper or lower cases. Its alphabet includes 28 letters: 25 consonants and only 3 vowels. But in addition to these vocal segments, the Arabic script uses diacritical marks as short vowels. These are placed either above or below the letters to provide the correct pronunciation and clarify the meaning of the word. The majority of MSA texts are written without short vowels. This is so because proficient speakers do not need diacritical marks in order to understand a given text. However, diacritical marks are often used in children’s books as well as books for Arabic learners. The absence of diacritical marks in the majority of texts presents a lexical ambiguity problem that challenges computational systems. For example the undiacritized word شعر may mean (شِعْرٌ poetry), (شَعْرٌ hair) or (شَعَرَ to feel). The target of our work is to explore the opinion mining problem in Arabic. So we go built a framework the following framework to automate and evaluate the opinion mining on tweets crawled from twitter.

* As shown in figure 5 the solution will be:

1. Crawl the tweets from twitter and store them into intermediate data storage.
2. Apply some data cleansing rules like removing numbers, removing elongation, lemmatization, POS and NER “Named Entity Recognition”.
3. Create some features that will contribute in the model building step.
4. Run the model as depict in figure 2 which is a hybrid of neural network feature and discrete feature as attention where red nodes represent neural features, and blue nodes represent discrete features the first layer for the model will detect the word embeddings, the second layer we use the CNN to learn the sentence embedding trying to detect the n-gram and the final layer to classify the tweets + the discrete features in the attention mechanism.

**Crawl Reviews**

**Review Database**

**Lemmatization**

**Remove numbers**

**Remove elongation**

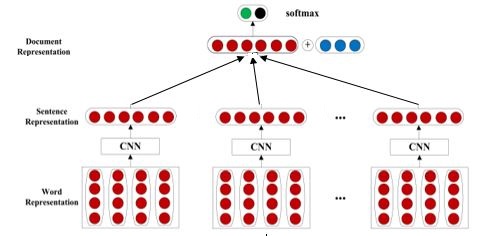
**Data cleaning**

**Discrete Features**

**Run The Model**

**Output**

**Figure 5: Opinion spamming framework architecture**



**Figure 6: Neural network model structure for opinion mining detection**

* **Experiment & Results**
* **Dataset**

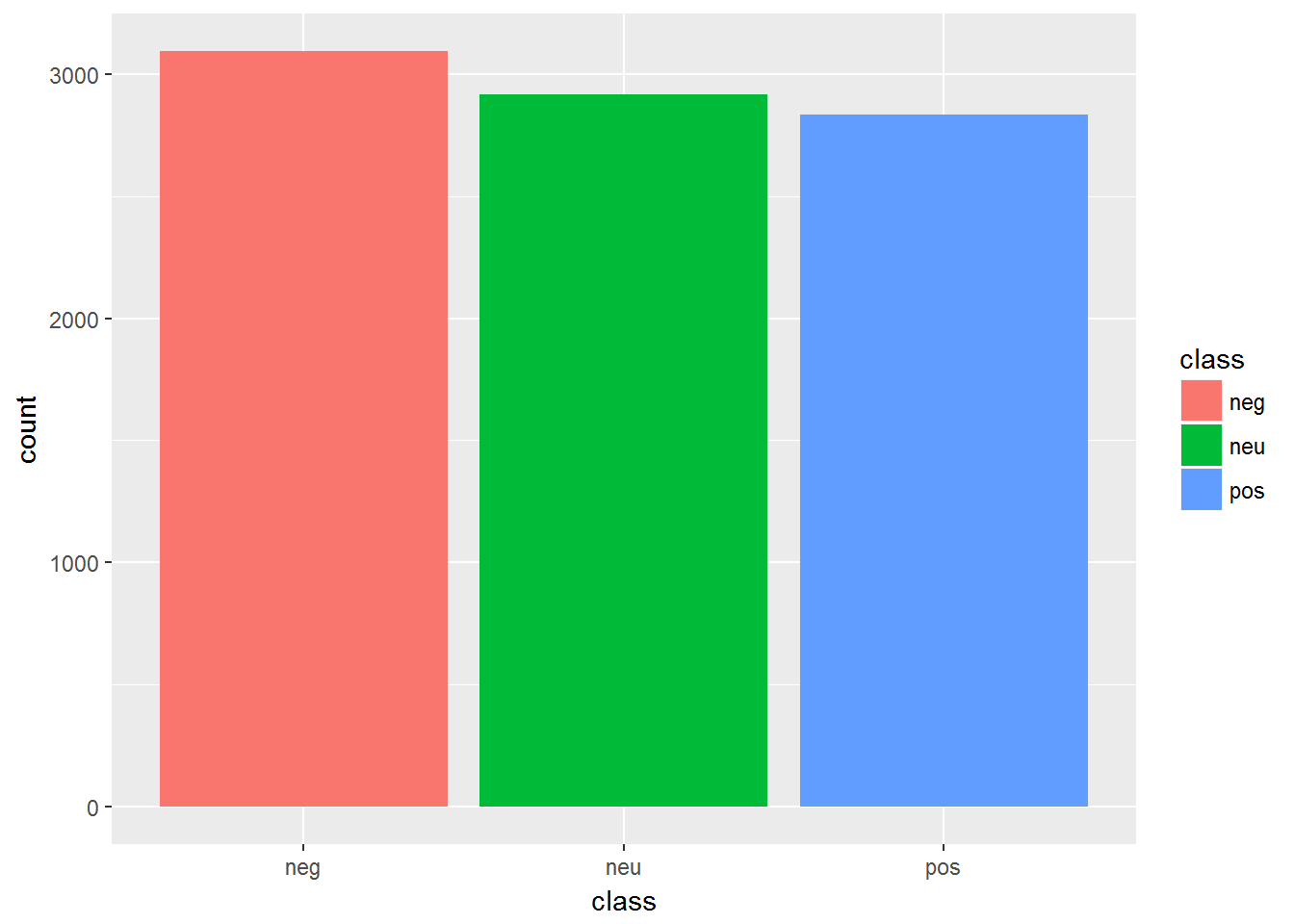
Tweets dataset crawled form twitter and saved into CSV file with the following attributes

1. Integer to represent the uniqueness of the tweet.
2. String represent the tweet body.
3. String to represent the polarity of the tweet (negative, positive or neutral).

The dataset contains 8845 tweets divided into 3 classes as the following

* + **3093** negative tweets
  + **2918** neutral tweets
  + **2834** positive tweets

And excluded the neutral tweets from the analysis.



* **Pre Processing**

In this step we applying the following cleansing rules to the dataset:

1. Removing the Arabic stop words.
2. Removing each mention in the tweet and replace it with a reserved word. ex. "@Mokhtar" to “كلمهمنشن”.
3. Removing each hashtag in the tweet and replace it with a reserved word. ex. “#We\_Love\_Deep\_Learning” to “كلمههشتاج”.
4. Removing each links in the tweet and replace it with a reserved word. ex. “Http://www.google.com” to “كلمهلينك”.
5. Removing each punctuation in the tweet. Ex. “?[]{}” .
6. In this step, words that have been elongated, are reduced to their normal standard form. An example of an elongated word is “yesssssss”" will be transformed to “yes” and “راااااااااااااااااائع”.to “رائع”
7. Replace the negative words in the lexicon with preserved word ex. “زفت” to “كلمهسالب”
8. Replace the positive words in the lexicon with preserved word ex. “حلو” to “كلمهموجب”
9. Remove white spaces.

* **Features Extractions:**

We are using deep learning technique for this task and theoretically we don’t need to manually extract feature and this step will be automatically figured out by out deep learning model but we want to examine the effect of the attention mechanism on our model by adding a discrete/ manually extracted features to the classification/final layer of the model and record the results. So we engineered the following features:

* 1. Perc\_Links: This feature is a normalized ratio of count how many link in the tweet.
  2. Perc\_Pos\_Words: This feature is a normalized ratio of count how many positive word in the tweet matching the positive lexicon words in the Lexicon.
  3. Perc\_Neg\_Words: This feature is a normalized ratio of count how many negative word in the tweet matching the postive lexicon words in the Lexicon.
  4. Perc\_Mentions: This feature is a normalized ratio of count how many mention in the tweet.
  5. numHashTags: This feature is a normalized ratio of count how many hashtag in the tweet.
  6. Length: This feature that can take on one of three hot encoded vectors [1, 0, 0], [0, 1, 0] or [0, 0, 1] depending on the length of the tweet. The numbers correspond to very short, short and normal. A tweet is categorized as “very short” if its length is less than 60 characters, “short” if it is less than 100, and normal otherwise.
* **Models**:

In our experiment we build 4 models to try to figure out the best performer model for this task:

* 1. Support Vector Machine model using feature extraction technique.
  2. CNN model and the word embeddings weights are learnt while training.
  3. CNN model and the word embeddings weight are extracted from a predefined word2vec model (fasttext, AraVec)
  4. CNN model and the word embeddings weight are extracted from a predefined word2vec model (fasttext, AraVec) and adding the attention layer (discrete featured) to the classification layer.
* **Results**:

In this section we are comparing the models performance

|  |  |  |
| --- | --- | --- |
| Model | Best Accuracy | Avg. Accuracy |
| Model (a) | 70% | 62% |
| Model (b) | 65% | 55% |
| Model (c) | 70% | 65% |
| Model (d) | 75% | 68% |

Also we used the following parameter for the CNN models (b, c and d)

* 1. Embedding size = **300**
  2. **6** convolutional layers with the following filter’s sizes **[2, 3, 4]** to detect **[2, 3, 4]** gram words.
  3. **6** max pooling layers.
  4. **Adam Optimizer** algorithm with learning rate = 0.001
* **Challenges**:

In our experiments for model c and d which use a pre trained word embeddings model we found that about 20% of the our dataset words are not exist in the word2vec model words “OOV” also we noticed that most of these words actually exist but in a different form like word “اقرا” and “أقرا” or “ابدا” and “بدا” so we calculated the levenshtein / edit distance between the 2 words and if they are small enough we lookup the vector represent this words form the word2vec model.

* **Algorithm**

1. Let oov\_word // the word which not exist in our dataset
2. Let distances [] // array contains all the distance between oov\_word and every word in the word2vec corpus
3. For word in word2Vec\_corpus:
   1. distances[word]=Calculate the levenshtein\_distance\_between(oov\_word, word)
4. min\_distance, word\_min = find\_min\_distances(distances)
5. if min\_distance <= threshold:
   1. retrun word\_embedding[word\_min]
6. else
   * 1. retrun OOV\_embeddings

* **References**

[1] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In Advances in Neural Information Processing Systems.

[2] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. arXiv preprint arXiv:1409.0473.

[3] Abu Bakr Soliman, Kareem Eisa, and Samhaa R. El-Beltagy, “AraVec: A set of Arabic Word Embedding Models for use in Arabic NLP”, in proceedings of the 3rd International Conference on Arabic Computational Linguistics (ACLing 2017), Dubai, UAE, 2017.

[4] Deep Learning for Sentiment Analysis: A Survey by Lei Zhang, LinkedIn Corporation, [lzhang32@gmail.com](mailto:lzhang32@gmail.com) Shuai Wang, University of Illinois at Chicago, [shuaiwanghk@gmail.com](mailto:shuaiwanghk@gmail.com) Bing Liu, University of Illinois a Chicago, [liub@uic.edu](mailto:liub@uic.edu)

[5] <https://en.wikipedia.org/wiki/Levenshtein_distance>