A Comprehensive Survey on Word Representation Models: From Classical to State-of-the-Art Word Representation Language Models

USMAN NASEEM, School of Computer Science, The University of Sydney, Australia
IMRAN RAZZAK, School of Information Technology, Deakin University, Australia
SHAH KHALID KHAN, School of Engineering, RMIT University, Australia
MUKESH PRASAD, School of Computer Science, University of Technology Sydney, Australia

Word representation has always been an important research area in the history of natural language processing (NLP). Understanding such complex text data is imperative, given that it is rich in information and can be used widely across various applications. In this survey, we explore different word representation models and its power of expression, from the classical to modern-day state-of-the-art word representation language models (LMS). We describe a variety of text representation methods, and model designs have blossomed in the context of NLP, including SOTA LMs. These models can transform large volumes of text into effective vector representations capturing the same semantic information. Further, such representations can be utilized by various machine learning (ML) algorithms for a variety of NLP-related tasks. In the end, this survey briefly discusses the commonly used ML- and DL-based classifiers, evaluation metrics, and the applications of these word embeddings in different NLP tasks.

CCS Concepts: • **Computing methodologies** → **Artificial intelligence**; **Natural language processing**; **Machine learning**; **Machine learning approaches**; **Information extraction;**

Additional Key Words and Phrases: Text mining, natural language processing, word representation, language models

ACM Reference format:
Usman Naseem, Imran Razzak, Shah Khalid Khan, and Mukesh Prasad. 2021. A Comprehensive Survey on Word Representation Models: From Classical to State-of-the-Art Word Representation Language Models. *ACM Trans. Asian Low-Resour. Lang. Inf. Process.* 20, 5, Article 74 (June 2021), 35 pages.
https://doi.org/10.1145/3434237

1 INTRODUCTION

Text-based data is increasing at a rapid rate, where the low quality of the unstructured text is growing rapidly than structured text. Textual data is very common in many different domains whether social media, online forums, published articles, or clinical notes for patients and online reviews given online where people express their opinions and sentiments to some products or

Authors' addresses: U. Naseem, School of Computer Science, The University of Sydney, Australia; email: usman.naseem@sydney.edu.au; I. Razzak, School of Information Technology, Deakin University, Australia; email: imran.razzak@deakin.edu.au; S. K. Khan, School of Engineering, RMIT University, Australia; email: s3680269@student.rmit.edu.au; M. Prasad, School of Computer Science, University of Technology Sydney, Australia; email: Mukesh.Prasad@uts.edu.au.

2375-4699/2021/06-ART74 $15.00
https://doi.org/10.1145/3434237

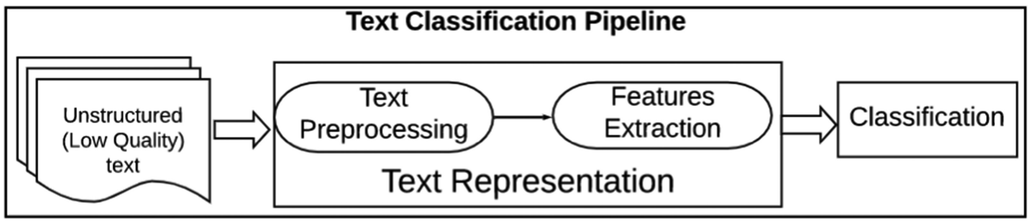

Fig. 1. Text classification pipeline.

businesses [53]. It is a rich source of getting information and gives more opportunity to explore valuable insights that can not be achieved from quantitative data [54]. The main aim of different NLP methods is to get a human-like understanding of the text. It helps to examine the vast amount of unstructured and low-quality text and discover appropriate insights. Coupled with ML, it can formulate different models for the classification of low-quality text to give labels or obtain information based on prior training. For instance, researchers in the past focused on mining the opinion and sentiments of users about a product, restaurant, and movie reviews, and so on, to predict the sentiment of users. Over the years text has been used in various applications such as email filtering [20], irony and sarcasm detection [113] document organization [46], sentiment and opinion mining prediction [107, 115], hate speech detection [105, 106, 110, 114], question answering [44], content mining [2], biomedical text mining [109, 112, 128], and many more.

However, being unstructured content, it adds complexity to the model, deciphers automatically, or uses in conjunction with traditional features for a ML framework [57]. Moreover, even though large of volumes of text information is widely available and can be leveraged for interesting applications, it is rife with problems. Like most data, it suffers from traditional problems such as class imbalance and lack of class labels, but besides, there are some inherent issues with text information. Apart from being unstructured, text mining and representation learning become more challenging due to the following discussed factors.

The language on social media is unstructured and informal. Social media users express their emotions and write in different ways, use abbreviations, punctuations, emoticons, slangs, and often use URLs. These language imperfections may cause noise and is a challenging task to handle by applying appropriate pre-processing techniques. Besides, understanding semantics, syntactical information and context is important for text analysis [104, 111].

Much research has been dedicated to addressing each of these concerns individually. However, in this survey, we focus on how text can be represented as numeric\continuous vectors for easier representation, understanding, and applicability to traditional machine-learning frameworks. Text may be seen as a collection of entities such as documents, sentences, words, or characters, and most algorithms leverage the implicit relationship between these entities to infer them as vectors.

Over the years, many methods and algorithms have been used to infer vectors from text be they at character, word, sentence, or document level. All the methods are aimed at better quantifying the richness in the information and making them more suitable for machine learning frameworks such as to perform clustering, dimensionality reduction, or text classification. In this survey, we study how text representation methods have evolved from manually selecting the features called feature engineering to more SOTA representational learning methods that leverage neural networks to discover relevant embeddings.

In any NLP task, first, we should have data that we are interested in analyzing. The next step is to represent the raw unstructured data in a form that ML classification algorithms can understand. Text representation is divided into main two parts: (i) Text pre-processing and (ii) Features Extraction and then classify the learned representations using an appropriate classifier [3, 69].

Contribution and Organization: In this article, we present a comprehensive study of various text representation methods starting from the bag of words approach to more SOTA representational learning methods. We describe various commonly used text representation methods and their variations and discuss various text mining applications they have been used in. We conclude with a discussion about the future of text representation based on our findings. We would like to note that this article strictly focuses on the representation of text for low-quality text Classification and therefore uses content, data, and text interchangeably.

Below, first, we briefly discuss different steps in text classification pipeline illustrated in Figure 1, followed by the details of each step in next sections.

(1) **Unstructured (Low Quality) Text:** Unstructured (Low Quality) text is a form of written text that requires metadata and cannot easily be listed or classified. Usually, it is the information generated by users on social media postings, documents, email, or messages. Raw text is scattered and sparse with less number of features and does not give sufficient word co-occurrence information. It is an important origin of information for businesses, research institutes, and monitoring agencies. Often companies mine it for getting the data to improve their marketing strategies and achieve an edge in the marketplace. It plays a big part in predictive analytics and in analyzing sentiments of users to find out the overall opinion of customers. It helps to discover unique insights by revealing hidden information, discovering trends and recognising relationships between irrelevant bits of the data [45, 161].

(2) **Text Representation:** For text classification, the text should be converted into a form that the computer can understand. First, we need to improve the quality of raw and unstructured text and then extract features from it before classification. Both of these steps are briefly discussed below.
   - **Text pre-processing:** Pre-processing is the crucial step, especially in the classification of short text. Pre-processing techniques are valuable techniques for decreasing the data adequacy, sparsity, and helps to improve the low quality of text especially in the case of short text where everyone writes in their style, and use emoticons, abbreviations, make spelling mistakes, use URLs, and so on. A proper combination of common and advance pre-processing techniques can help to learn good text representation [8, 146]. pre-processing techniques analyzed in our study are briefly discussed in Section 2.
   - **Features Extraction:** Features extractions of the data is the critical step for machines to classify and understand the data like humans. It is the process of transforming raw data into numeric data that machines can understand. Usually, this feature extraction step of transforming a raw data is called a features vector. Extracting robust word representations is not so easy without having a considerable amount of corpus due to diversity of expressing sentiments, emotions, and intentions in the English language. However, due to social media platforms, researchers now have access to get an enormous amount of data. However, assigning labels to this massive amount of data collected from social media platforms is not an easy job. To make this annotation process easy, researchers initially worked on finding a sign of sentiment and emotion within the content of the text like emoticons and hashtags [69, 155, 165]. Some of the famous classical and current feature extraction algorithms are briefly discussed in Section 3.

(3) **Classification:** Selecting the best classifier is the essential part of text classification pipeline. It is hard to find out the most effective and adequate classifier for text classification task without understanding theoretically and conceptually each algorithm. Since the scope of this article is only restricted to present different text representation methods, we will not discuss different text classification algorithms in detail. These classifiers include famous

traditional ML algorithms of text classification such as Logistic Regression, which is used in many data mining areas [23, 123], Naive Bayes, which is computationally not expensive and works well with less amount of memory [73], K-nearest Neighbor, which is non-parametric methods, and Support Vector Machine is a famous classifier that has been widely used in many different areas earlier. Then tree-based algorithms such as random forest and decision tree are discussed followed by **deep learning (DL)**-based classifiers that are a collection of methods and approaches motivated by the working mechanism of the human brain. These methods utilise the extensive amounts of training input data to achieve the high quality of semantically rich text representations that can be given as input to different ML methods that can make better predictions [67, 69].

2 TEXT PRE-PROCESSING

Text datasets contain a lot of unwanted words such as stop-words, punctuation, incorrect spellings, slangs, and so on. This unwanted noise and words may have an negative effect on the performance of the classification task. Below, first, we present the preliminaries where we discuss different methods and techniques related to text pre-processing and cleaning, followed by some literature review where researchers analyzed the effects of text pre-processing techniques.

2.1 Preliminaries Related to Text Pre-processing

- **Tokenization:** A process of transforming a text (sentence) into tokens or words is known as tokenization. Documents can be tokenized into sentences, whereas sentences can be converted into tokens. In tokenization, a sequence to text is divided into the words, symbols, phrases, or tokens [6]. The prime objective of tokenization is to find out the words in a sentence. Usually, tokenization is applied as a first and standard pre-processing step in any NLP task.[40]
- **Removal of Noise, URLs, Hashtag, and User-mentions:** Unwanted strings and Unicode are considered as leftover during the crawling process, which is not useful for the machines and creates noise in the data. Also, almost all of tweets messages posted by users, contains URLs to provide extra information, User-mention/tags (α) and use hashtag symbol "#″ to associate their tweet message with some particular topic and can also express their sentiments in tweets by using hashtags. These give extra information that is useful for human beings, but it does not provide any information to machines and considered as noise that needs to be handled. Researchers have presented different techniques to handle this extra information provided by users such as in the case of URLs; it is replaced with tags [1] whereas User-mentions (α) are removed [13, 65]
- **Word Segmentation:** Word segmentation is the process of separating the phrases, content, and keywords used in the hashtag. Moreover, this step can help in understanding and classifying the content of tweets easily for machines without any human intervention. As mentioned earlier, Twitter users use # (hashtags) in almost all tweets to associate their tweets with some particular topic. The phrase or keyword starting with # is known as hashtags. Various techniques are presented in the literature for word segmentation in References [22, 136].
- **Replacing Emoticons and Emojis:** Twitter users use many different emoticons and emojis such as:), :(, and so on. to express their sentiments and opinions. So it is important to capture this useful information to classify the tweets correctly. There are few tokenizers available that can capture few expressions and emotions and replace them with their associated meanings [41].
- **Replacement of abbreviation and slang:** Character limitations of Twitter enforce online users to use abbreviations, short words, and slangs in their posts online. An abbreviation is

a short or acronym of a word such as MIA, which stands for missing in action. In contrast, slang is an informal way of expressing thoughts or meanings, which is sometimes restricted to some particular group of people, context, and considered as informal. So it is crucial to handle such kind of informal nature of text by replacing them to their actual meaning to get better performance without losing information. Researchers have proposed different methods to handle this kind of issue in a text, but the most useful technique is to convert them to an actual word that is easy for a machine to understand [68, 100].

- **Replacing elongated characters:** Social media users, sometimes intentionally use elongated words in which they purposely write or add more characters repeatedly more times, such as "loooovvveee, greeeeat." Thus, it is important to deal with these words and change them to their base word so that classifier does not treat them different words. In our experiments, we replaced elongated words to their original base words. Detection and Replacement of elongated words have been studied by References [97] and [5].
- **Correction of Spelling mistakes:** Incorrect spellings and grammatical mistakes are very commonly present in the text, especially in the case of social media platforms, especially on Twitter and Facebook. Correction of spelling and grammatical mistakes helps in reducing the same words written indifferently. Textblob is one of the libraries that can be used for this purpose. Norvig's spell correction[1] method is also widely used to correct spelling mistakes.
- **Expanding Contractions:** A contraction is a shortened form of the words that is widely being used by online users. An apostrophe is used in the place of the missing letter(s). Because we want to standardize the text for machines to process easily, in the removal of contractions, shortened words are expanded to their original root /base words. For example, words like "how is," "I'm," "can't," and "don't" are the contractions for words "how is," "I am," "cannot," and "do not," respectively. In the study conducted by Reference [14], contractions were replaced with their original words or by the relevant word. If contractions are not replaced, then the tokenization step will create tokens of the word "can't" into "can" "t."
- **Removing Punctuations:** Social media users use different punctuations to express their sentiments and emotions, which may are useful for humans but not all much useful for machines for the classification of short texts. So removal of punctuation is common practice in classification tasks such as sentiment analysis. However, sometimes some punctuation symbols such as "!" and "?" shows/denotes the sentiments. It is common practice to remove punctuation [82]. Whereas, replacing question mark or sign of exclamation with tags has also been studied by Reference [5].
- **Removing Numbers:** Text corpus usually contains unwanted numbers that are useful for human beings to understand but not much use for machines, which makes lower the results of the classification task. The simple and standard method is to remove them [47, 58]. However, we could lose some useful information if we remove them before transforming slang and abbreviation into their actual words. For example, words such as "2maro," "4 u," "gr8," and so on, should be first converted to actual words, and then we can proceed with this pre-processing step.
- **Lower-casing all words:** A sentence in a corpus has many different words with capitalization. This step of pre-processing helps to avoid different copies of the same words. This diversity of capitalization within the corpus can cause a problem during the classification task and lower the performance. Changing each capital letters into a lower case is the most common method to handle this issue in text data. Although this pre-processing technique projects all tokens in a corpus under the one feature space also causes a bunch of problems

[1] http://norvig.com/spell-correct.html.

in the interpretation of some words such as "US" in the raw corpus. The word "US" could be pronoun and a country name as well, so converting it to a lower case in all cases can be problematic. The study conducted by Reference [33] has lower-cased words in corpus to get clean words.
- **Removing Stop-words:** In-text classification task, there are many words that do not have critical significance and are present in high frequency in a text. It means the words that do not help to improve the performance because they do not have much information for the sentiment classification task, so it is recommended to remove stop words before feature selection step. Words such as (a, the, is, and, am, are, etc.). A popular and straightforward method to handle with such words is to remove them. There are different stop-word libraries available such as NLTK, scikit-learn, and spaCy.
- **Stemming:** One word can turn up in many different forms, whereas the semantic meaning of those words is still the same. Stemming is the techniques to replace and remove the suffixes and affixes to get the root, base, or stem word. The importance of stemming was studied by Reference [92]. There are several types of stemming algorithms that help to consolidate different forms of words into the same feature space such as Porter Stemmer, Lancaster stemmer and Snowball stemmers, and so on, Feature reduction can be achieved by utilizing the stemming technique.
- **Lemmatization:** The purpose of the lemmatization is the same as stemming, which is to cut down the words to its base or root words. However, in lemmatization inflection of words are not just chopped off, but it uses lexical knowledge to transform words into its base forms. There are many libraries available that help to do this lemmatization technique. Few of the famous ones are NLTK (Wordnet lemmatizer), genism, Stanford CoreNLP, spaCy, TextBlob, and so on.
- **Part of Speech (POS) Tagging:** The purpose of Part of speech (POS) tagging is to assign part of speech to text. It clubs together with the words that have the same grammatical with words together.
- **Handling Negations:** For humans, it is simple to get the context if there is any negation present in the sentence, but for machines sometimes it does not help to capture and classify accurately so handling a negation can be a challenging task in the case of word-level text analysis. Replacing negation words with the prefix "NEG" has been studied by Reference [103]. Similarly, handling negations with antonym has been studied by Reference [124].

2.2 Related Work on Text Pre-processing Methods

Text pre-processing plays a significant role in text classification. Many researchers in the past have made efforts to understand the effectiveness of different pre-processing techniques and their contribution to text classification tasks. Below, we present some studies conducted on the effects of pre-processing techniques on text classification tasks.

Bao et al. [8] study showed the effect of pre-processing techniques on Twitter analysis task. Unigram and bi-grams features were fed to Liblinear classifier for the classification. They showed in their study that reservation of URL features, the transformation of negation (negated words), and normalization of repeated tokens have a positive effect on classification results, whereas lemmatization and stemming have a negative effect on classification results. Singh and Kumari [146] showed the impact of pre-processing on Twitter dataset full of abbreviations, slangs, acronyms for the sentiment classification task. In their study, they showed the importance and significance of slang and correction of spelling mistakes and used **Support Vector Machine (SVM)** classifier to study the role of pre-processing for sentiment classification. Haddi et al. [45] also explored the effect of text pre-processing on movie review dataset. The experimental shows that pre-processing methods such as the transformation of text such as changing abbreviations to actual words and removal of stop word, special characters, and handling of negation with the prefix "NOT" and stemming can significantly improve the classification performance. The SVM classifier was used in the experiments conducted by Uysal and Gunal: Reference [161] analyzed the role of pre-processing on two different languages for sentiment classification. They employed SVM classifier in their studies and showed that performance is improved by selecting an appropriate combination of different techniques such as removal of stop words, the lower casing of text, tokenization, and stemming. They concluded that researchers should choose all possible combinations carefully, because the inappropriate combination may result in degradation of performance. Similarly, Jianqiang and Xiaolin [59] studied the role of six different pre-processing techniques on five datasets in their study, where they used four different classifiers. Their experimental results show that replacing acronyms (abbreviations) with actual words and negations improved the sentiment classification, whereas removing stop-words, special characters, and URLs have an adverse influence on the results of sentiment classification. Role of text pre-processing to reduce the sparsity issue in Twitter sentiment classification is studied by Said et al. [139]. Experimental results demonstrate that choosing a combination of appropriate pre-processing methods can decrease the sparsity and enhance the classification results. Agarwal et al. [1] proposed novel tweets pre-processing approaches in their studies. They replaced URL, user mentions, repeated characters, and negated words with different tags and removed hashtags. Classification results were improved by their proposed pre-processing methods. Other studies were presented by Saloot et al. [140] and Takeda and Takefuiji [168] in the natural language workshop that focuses on noise user-generated text.[2] Noisy nature of Twitter messages is reduced/decreased by normalizing tweets using a maximum entropy model and entity linking. Recently, Symeonidis et al. [156] presented the comparative analysis of different techniques on two datasets for Twitter sentiment analysis. In their study, they studied the effect of each technique on four traditional ML-based classifiers and one neural network-based classifier with only TFIDF (unigram) for words representation method. Their study showed that pre-processing technique such as removing numbers, lemmatization, and expanding contractions to base words performs better, whereas removing punctuation does not perform well in the classification task. Their study also presented the interactions of the limited number of different techniques with others and showed that techniques perform well when interacted with others. However, no work has been done the recommendation of pre-processing techniques to improve the quality of the text.

3 FEATURE EXTRACTION METHODS

In this section, we discuss various popularly used feature extraction models. Different researchers in the past have proposed different features of extraction models to address the problem of losing syntactic and semantic relationships between words. These methods, along with the literature review where different methods have been adopted for different NLP-related tasks. First, we present some classical models, followed by some famous representation learning models.

3.1 Classical Models

This section presents some of the classical models that were commonly used in earlier days for the text classification task. Frequency of words is the basis of this kind of words representation methods. In these methods, a text is transformed into a vector form that contains the number of the words appearing in a document. First, we give a short description of categorical word representation methods and then weighted word representation methods.

[2]http://noisy-text.github.io/.

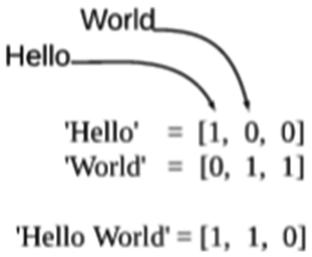

Fig. 2. An illustration of one-hot encoding and BoW models.

(1) **Categorical word representation:** is the simplest way to represent text. In this method, words are represented by a symbolic representation either "1" or "0." One-hot encoding and **Bag-of-words (BoW)** are the two models that come under categorical word representation methods. Both are briefly discussed below.
- **One hot encoding:** The most straightforward method of text representation is one hot encoding. In one hot encoding, the dimension is the same amount of terms present in the vocabulary. Every term in vocabulary is represented as a binary variable such as 0 or 1, which means each word is made up of zeros and ones. Index of the corresponding word is marked with 1, whereas all others are marked as zero (0). Each unique word has a unique dimension and will be represented by a 1 in that dimension with 0s everywhere else.
- **Bag-of-Words (BoW):** BoW is simply an extension of one-hot encoding. It adds up the one-hot representations of words in the sentence. The BOW method is used in many different areas such as NLP, **computer vision (CV)**, **information retrieval (IR)**, and so on. The matrix of words built using BOW ignore the semantic relationship between words and order of word is also ignored along with the grammar.
  As stated, BOW is an extension of one-hot encoding, e.g., encodes token in the vocabulary as a 1-hot-encoded vector. As vocabulary may increase to huge numbers, then vocabulary size would increase and thereby, the length of the vectors would increase, too. Besides, a large number of "0s" that may result in a sparse matrix, containing no order of text as well as information of the grammar used in the sentences.
  An example of "*Hello*" and "*World*" as one-hot encoding and "*Hello World*" as BoW is given in Figure 2.

(2) **Weighted Word representation:** Here, we present the common methods for weighted word representations such as **Term Frequency (TF)** and **Term Frequency-Inverse Document Frequency (TF-IDF)**. These are associated with categorical word representation methods but rather than only counting, weighted models feature numerical representations based on words frequency. Both of them are briefly discussed below.
- **Term Frequency (TF):** Term frequency, is the straightforward method of text feature extraction. TF calculates how often a word occurs in a document. A word can probably appear many times in large documents as compared to small ones. Hence, TF is computed by dividing the length of the document. In other words, TF of a word is computed by dividing it with the total number of words in the document.
- **Term Frequency-Inverse Document Frequency (TF-IDF):** To cut down the impact of common words such as "the," "and," and so on, in the corpus, TF-IDF was presented by Reference [150] for text representation. TF here stands for Term frequency, which is defined in the above section, and IDF denotes inverse document frequency, which is a technique presented to be used with TF to reduce the effect of common words. IDF assigns

a more weight to words with either higher or lower frequencies. This combination of TF and IDF method is known as TF-IDF and is represented mathematically by the below equation.

$$TF - IDF(t, d, D) = TF(t, d) \times \log\left(\frac{D}{df_t}\right),$$

where t denotes the terms; d denotes each document; D represents the collection of documents; and df_t denotes sum of documents with term t in it. TF-IDF is built on the concept of BOW model; therefore, it can not capture the order of words in a document, semantics, and syntactical information of words. Hence, TF-IDF is good to use as a lexical level feature.

3.2 Representation Learning

Since categorical word representations, models fail to capture syntactic and semantic meaning of the words, and these models suffer the curse of high dimensionality. The shortcomings of these models led the researchers to learn the distributed word representation in low dimension space [17]. The limitations of classical feature extraction methods make it use a limited for building a suitable model in ML. Due to this, different models have been presented in the past, which discovers the representations automatically for downstream tasks such as classification. Such methods that discover features itself are called as feature learning or representation learning.

It is very important, because the performance of ML models heavily depends on the representations of the input [10]. DL-based models, which are good at learning important features itself, is changing traditional feature learning methods. Proper representation can be learned either by utilizing supervised learning methods or unsupervised methods.

In the area of NLP, unsupervised text representation methods like word embeddings have replaced categorical text representation methods. These word embeddings turned into very efficient representation methods to improve the performance of various downstream tasks due to having a previous knowledge for different ML models. Classical feature learning methods have been replaced by these neural network-based methods due to their good representation learning capacity. Word embedding is a feature learning method where a word from the vocabulary is mapped to N dimensional vector. Many different words embedding algorithms have been presented, and the famous algorithms, for instance, *Word2vec*, *GloVe,* and *FastText*, are discussed in this study.

First, we briefly present different pre-training methods for learning the word representation of the document. These pre-training methods are classified into three different groups: (i) Supervised learning (SL), (ii) Unsupervised learning (UL), and (iii) Self-supervised learning (SSL). Below, we discuss each of these briefly:

(1) **Supervised learning (SL)** is to learn a feature that translates an input to an output on a basis of input-output pair training data.
(2) **Unsupervised learning (UL)** is to discover some intrinsic information, such as clusters, densities, latent representations, from unlabeled data.
(3) **Self-Supervised learning (SSL)** is a hybrid of SL and UL. SSL's learning model is mostly the same as SL, except the training data labels are automated. SSL's main concept is to predict some aspect of the input in some form from other parts. The **Masked Language Model (MLM)**, for instance, is a self-supervised task that tries to predict the masked words in a sentence provided the remaining words.

3.2.1 Distributed Representations. As previously mentioned, hand-crafted features were primarily used to model tasks in natural language before approaches based on neural networks came around and addressed some of the challenges faced by conventional Ml algorithms, such as the dimensionality curse.

(1) **Continuous Words Representation (Non-Contextual Embeddings):** Word Embedding is NLP technique in which text from the corpus is mapped as the vectors. In other words, it is a type of learned representation that allows same meaning words to have the same representation. It is the distributed representation of a text (words and documents) that is a significant breakthrough for better performance for NLP-related problems. The most significant benefit of word embedding is this that it provides more efficient and expressive representation by keeping the word similarity of context and by low dimensional vectors. Nowadays, word embedding is being used in many different applications such as semantic analysis, philology, psychiatry, cognitive science, social science, and psychology [34].
An automatic feature learning technique in which every token in a vocabulary is indexed into an N dimension vector is known as distributed vectors or Word embedding, which follows the distributional hypothesis. According to this, words that are used and appear in the similar contexts tend to assure the same meanings. So these vectors tend to have the attributes of word's neighbors, and they capture the similarity between words. In the 1990s, several researchers made attempts to lay down the foundation of distributional semantic learning.
Bengio et al. [11] presented a model that learned word representations using distributed representation. Authors presented NNLM model, which obtains word representations as to the product while training **language model (LM)**. Just like traditional LM, NNLM also uses previous $n-1$ words/tokens to predict the nth word/token. Different word embedding models have been proposed, which makes uni-grams useful and understandable to ML algorithms and usually, these models are used in the first layer in a deep neural network-based model. These word embedding are pre-trained by predicting a word based on its context without losing semantic and syntactical information. Thus, using these embedding techniques have demonstrated to be helpful in many NLP tasks, because it does not lose the order of words and captures the meaning of words (syntactic and semantic information of words). However, the popularity of word representation methods are due to two famous models, Word2vec [96] and GloVe [89]. These, along with others, are briefly discussed below.

- **Word2vec:** Word2vec is words representation model developed by Reference [96]. This model uses two hidden layers that are used in a shallow neural network to create a vector of each word. The word vectors captured by **Continuous Bag of words (CBOW)** and Skip-gram models of Word2vec are supposed to have semantic and syntactic information of words. To have a better representation of words, it is recommended to train the corpus with the large corpus. Word2vec has proved to be useful in many NLP-related tasks [28]. Word2vec was developed to build training of embedding more significant, and since then, it has been used as a standard for developing pre-trained word representation. Based on the context, Word2vec predicts by using one of the two neural network models such as CBOW and Skip-gram. A predefined length of the window is moved together with the corpus in both models, and the training is done with words inside the window in each step [4]. This feature presentation algorithm gives a robust tool for unfolding relationships in the corpus and the similarity between token. For instance, this method would regard the two words such as "*small*" and "*smaller*" near to each other in the vector space. Figure 3 shows the working principle of both Word2vec algorithms, *CBOW,* and *Skip-Gram.*
  - **Continuous Bag of words (CBOW):** Continuous Bag of words (CBOW) gives words prediction of current work based on its context. CBOW communicates with the neighboring words in the window. Three layers are used in CBOW process. Context is

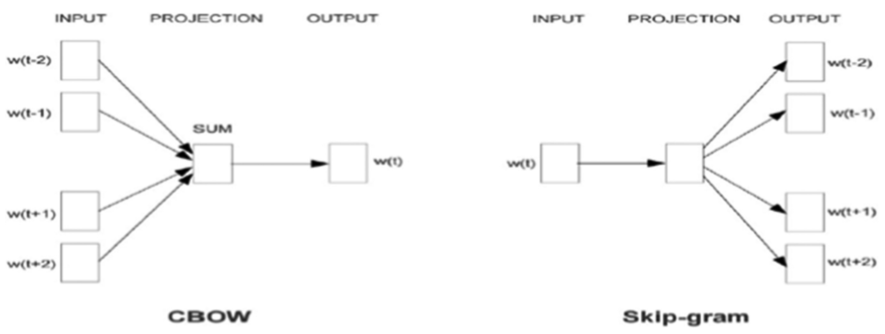

Fig. 3. Working principle of Word2vec (Image taken from [96]).

considered as the first layer, whereas the layer that is hidden matches with the estimation of every word from the input to the weight matrix, which later on is estimated to the output that is considered as the third layer. The last phase of this method is to correlate the output and the work itself to improve the representation based on the back-propagation of the error gradient. In Figure 3, CBOW method predicts the middle word based on its context in skip-gram predicts the context word based on center word [102].

– **Skip-Gram:** Skip-Gram is the reverse of CBOW model: Prediction is given based on the central word after the training of context in skip-gram. Input layer correlates with the targeted word, and the output layer corresponds with the context. This model looks for the estimation of the context given the word, unlike CBOW. The last phase of this model is the correlation between output and every word of the context to adjust representation based on back-propagation [34, 102].
Skip-gram is efficient when we have less training data and not frequent words are well presented. In comparison, CBOW is quicker and performs better with repeated words. To address the issues of learning the final vectors, two algorithms are proposed. First one is negative sampling in which we restrict the sum of output vectors that needs to be updated, so only a sample of the vectors is updated based on a noise distribution that is a probabilistic distribution used in the sample step. Moreover, the other method is Hierarchical softmax, which is developed based on the Huffman tree. It is a binary tree that gives all words depending on their counts. Then normalization is done for each step from the root to the target. Negative sampling is efficient when the dimension of vectors is less and works well with repeated words. In comparison, hierarchical softmax works well when we have less frequent words [102].

• **Global Vectors (GloVe):** Word2vec-trained word embedding will better capture the semantics of words and manipulate the connectivity of words. However, Word2vec mainly focuses on the local context window knowledge, whereas the global statistical information is not used well. So the GloVe [89] is presented, which is a famous algorithm based on the global co-occurrence matrix, each element X_{ij} in the matrix depicts the frequency of the word w_i and the word w_j co-occur in a appropriate context window and is widely used for the text classification task.
GloVe is an expansion of the Word2vec for learning word vectors efficiently where the words prediction is made based on surrounding words. GloVe is based on the appearances of a word in the corpus, which is based on two steps. Creation of the co-occurrence matrix from the corpus is the first step, followed by the factorization to get vectors. Like

Word2vec, GloVe also provided pre-trained embeddings in a different dimension (100, 200, 300 dimensions) that are trained over the vast corpus. The objective function of GloVe is given below:
$J = \sum_{k,j=1}^{V} f(X_{kj})(w_k^T w_j^{'} + b_k + b_j - \log X_{kj})$
where;
V : is size of vocabulary,
X : is co-occurrence matrix,
X_{kj} is frequency of word k co-occurring with word j,
X_k total number of occurrences of word k in the corpus,
P_{kj} is the probability of word j occurring within the context of word k,
w is a word embedding of dimension d,
$w^{'}$ is the the context word embedding of dimension d.
Word representation methods such as Word2vec and GloVe are simple, accurate, and on large datasets, they can learn semantic representations of words. They do not, however, learn embedded words from **out-of-vocabulary(OOV)** words. Such words can be defined in two ways: words that are not included in the current vocabulary and words that do not appear in the current training corpus. To solve these, various models are proposed to address this challenge. We briefly describe one of the most famous models below.

- **FastText:** Bojanowski et al. [15] proposed FastText and is based on CBOW. When compared with other algorithms, FastText decreases the training time and maintains the performance. Previously mentioned algorithms assign a distinct representation to every word that introduces a limitation, especially in case of languages with sub-word level information/ OOV.
  FastText model solved the issues mentioned above. FastText breaks a word in n-grams instead of full word for feeding into a neural network, which can acquire the relationship between characters and pick up the semantics of words. FastText gives better results by having better word representation primarily in the case of rare words. Facebook has presented pre-trained word embeddings for 294 different languages, trained on Wikipedia using FastText embedding on 300 dimensions and utilized Word2Vec skip-gram model with its default parameters [63].

Although these models are used to retain syntactic and semantic information of a document, there remains the issue of how to keep the full context-specific representation of a document. Understanding the actual context is required for the most downstream tasks in NLP. Some work recently tried to incorporate the word embedding with the LM to solve the problem of meaning. Below, some of the common context-based models are briefly presented.

(2) **Contextual word representations:**

- **Generic Context word representation (Context2Vec):** Generic Context word representation (Context2Vec) was proposed by Melamud et al. [94] in 2016 to generate context-dependent word representations. Their model is based on Word2vec's CBOW model but replaces its average word representation within a fixed window with better and powerful Bi-directional LSTM neural network. A large text corpus was used to learn neural model that embeds context from a sentence and target words in the same low dimension that later is optimized to reflect the inter-dependencies between target and their entire sentential context as a whole as shown in Figure 4.
- **Contextualized word representations Vectors (CoVe):** McCann et al. [90] presented their model contextualized word representations vectors (CoVe), which is based on context2Vec. They used machine translation to build CoVe instead of the approach used in Word2vec (skip-gram or CBOW) or GloVe (Matric factorization). Their basic approach

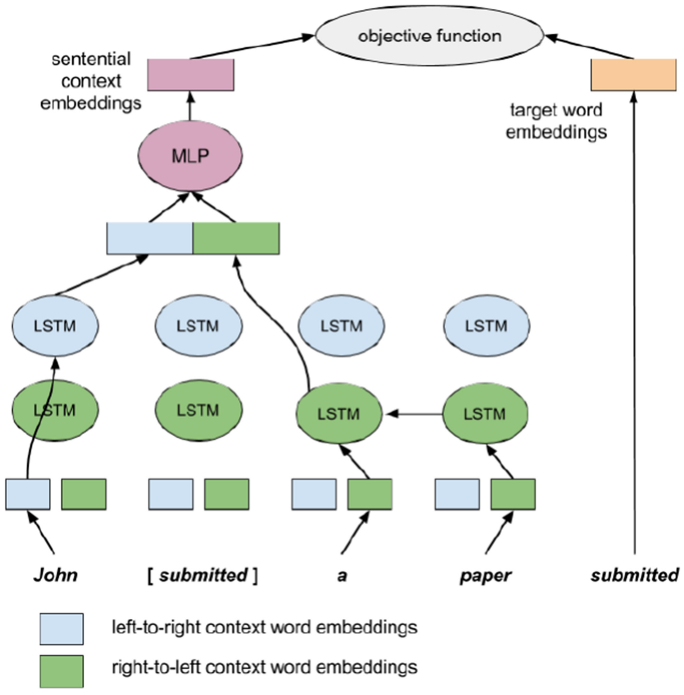
objective function
sentential context embeddings
target word embeddings
MLP
LSTM
John
[submitted]
a
paper
submitted
left-to-right context word embeddings
right-to-left context word embeddings

Fig. 4. Working principle of Context2Vec
(Image taken from [93]).

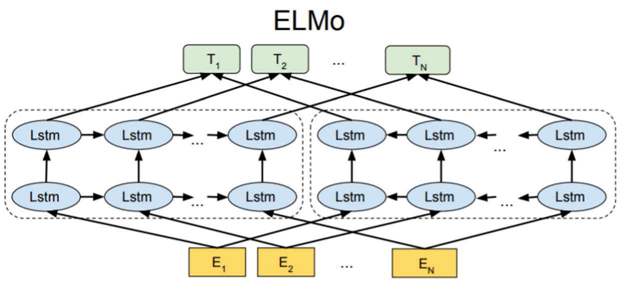

Fig. 5. Working principle of ELMo
(Image taken from [30]).

was to pre-train two-layer BiLSTM for attention sequence to sequence translation, starting from GloVe word vectors, and then they took the output of sequence encoder and called it a CoVe, combined it with GloVe vectors, and used in a downstream task-specific mode using transfer learning.

- **Embedding from language Models (ELMo):** Peters et al. [125] proposed Embedding from Language Models (ELMo), which gives deep contextual word representations. Researchers concur that two problems should be taken into account in a successful word representation model: the dynamic nature of word use in semantics and grammar, and as the language environment evolves, these uses should alter. They therefore introduce a method of deep contextualized word representation to address the two problems above, as seen in Figure 5.

The final word vectors are learned from bi-directional language model (forward and backward LMs). ELMo uses the linear concatenation of representations learned from bidirectional language model instead of only just the final layer representations like other contextual word representations. In different sentences, ELMo provides different word representations for the same word. Word representations learned by ELMo are based on the representation learned from **Bi-language model (BiLM)**. The log-likelihood of sentences is used in the training phase of BiLMs both in forward and backward LMs. The final vector is computed after the concatenation of hidden representations from forwarding LM $\overrightarrow{h}_{n,j}^{LM}$ and backward LM $\overleftarrow{h}_{n,j}^{LM}$, where $j = 1, \ldots, L$ and is given below:

$$\begin{aligned} BiLM = \sum_{n=1}^{k} (\log p(t_n | t_1, \ldots, t_{n-1}; \Theta_x, \overrightarrow{\Theta}_{LSTM}, \Theta_s) \\ + \log p(t_n | t_{n+1}, \ldots, t_n; \Theta_x, \overleftarrow{\Theta}_{LSTM}, \Theta_s), \end{aligned}$$

where the token representation parameters and softmax parameters θx and θs are shared between the forward and backward directions, respectively. And $\overrightarrow{\Theta}_{LSTM}$ and $\overleftarrow{\Theta}_{LSTM}$ are then forward and backward LSTM parameters, respectively. In a downstream task, ELMo extracts the representations learned from BiLM from an intermediate layer and executes a linear combination for each token. BiLM contains *2L+1* set representations as given below:

$$\begin{aligned} R_n &= (X_x^{LM}, \overrightarrow{h}_{n,j}^{LM}, \overleftarrow{h}_{n,j}^{LM} | \quad j = 1, \ldots, L) \\ &= (h_{n,j}^{LM} | \quad j = 0, \ldots, L), \end{aligned}$$

where $h_{n,0}^{LM} = x_n^{LM}$ is the layer of token and $h_{n,j}^{LM} = [\overrightarrow{h}_{n,j}^{LM}, \overleftarrow{h}_{n,j}^{LM}]$ for each BiLSTM layer. ELMo is a combination of these characteristics unique to the task where all layers in M are flattened to a single vector and are given below:

$$ELMo_n^{task} = E(M_n; \Theta^{task}) = \gamma^{task} \sum_{j=0}^{L} s_j^{task} h_{h,j}^{LM}, \tag{1}$$

where s^{task} are weights that are softmax normalized for the combination of representations from different layers and γ^{task} is a hyper-parameter for optimization and scaling of representations.

Table 1 presents the comparison of Classical, non-contextual and contextual (Context2Vec, CoVe, and ELMo) LMs with their Pros and cons.

Summary: Text representation embeds textual data into a vector space, which significantly affects the performance of downstream learning tasks. Better representation of text is more likely to facilitate better performance if it can efficiently capture intrinsic data attributes. Below, we briefly highlight the limitations of categorical and continuous word representation models.

Classical word representation methods such as categorical and weighted word representations are the most naive and most straightforward representation of textual data. These legacy word representation models have been used widely in early days for different classification tasks such as document classification, **Natural language processing (NLP)**, information retrieval, and **computer vision (CV)**. The categorical word representation models are simple and not difficult to implement, but have limitations, such as they do not consider capture semantics and syntactic information, because they do not consider the

Table 1. Comparison of Classical, Non-contextual, and Contextual (Context2Vec, CoVe, ELMo) Word Representation Models

Word Representation Models	Model	Architecture	Type	Pros	Cons
Classic words Representations	One Hot Encoding and BoW	-	Count based	(i) Easy to compute (ii) Works with the unknown word (iii) Fundamental metric to extract terms	(i) It does not capture the semantics & syntactic info (ii) Common words effect on the results (iii) Can not capture sentiment of words
	TF and TF-IDF	-		(i) Easy to compute (ii) Fundamental metric to extract the descriptive terms (iii) Because of IDF, common terms do not impact results	(i) It does not capture the semantics & syntactic info (ii) Can not capture the sentiment of words
Continuous (Non-Contextual) words Representations	Word2vec	Log Bilinear	Prediction based	(i) It captures the text semantics & syntactic (ii) Trained on huge corpus (Pre-trained)	(i) Fails to capture contextual information (ii) It fails to capture OOV words (iii) Need huge corpus to learn
	GloVe	Log Bilinear	Count based	(i) Enforce vectors in the vector space to identify sub-linear relationships (ii) Smaller weight will not affect the training progress for common words pairs such as stop words	(i) It fails to capture contextual information (ii) Memory utilization for storage (iii) It fails to capture OOV words (iv) Need huge corpus to learn (Pre-trained)
	FastText	Log Bilinear	Prediction based	(i) Works for rare words (ii) Addresses OOV words issue	(i) It fails to capture contextual information (ii) Memory consumption for storage (iii) Compared to GloVe and Word2vec, it is more costly computationally
Contextual words Representations	Context2Vec CoVe & ELMo	BiLSTM	Prediction based	(i) It solves the contextual information issue	(i) Improves performance (ii) Computationally more expensive (iii) Requires another word embedding for all LSTM and feed-forward layer

order of words and do not consider any relationship between words. Further, the size of the input vector is proportional to vocabulary size, which makes them computationally expensive, which results in poor performance.

Representation learning methods have helped the research community to build powerful models. However, its drawback is that the features need to be selected manually so to solve this shortcoming there was a need to present some methods that can discover and learn these representations automatically for any downstream task. This automatic extraction of features without human intervention is known as representation learning that has improved results drastically over the past few years in many areas such as image detection, speech recognition, NLP, and so on [76]. Continuous word representation models such as Word2vec, GloVe, FastText [16, 63, 89, 96] and so on, have drastically improved the classification results and overcome shortcomings of categorical representations. It is found that having these continuous word representation of words is more affected as compared to traditional linguistic features because of their ability to capture more semantic and syntactic information of the textual data without losing much information. Despite their success,

Table 2. Gap Analysis of Classic, Non-contextual, Contextual (Context2Vec, Cove, and ELMo) LMs

Language Models	**Semantics**	**Syntactical**	**Context**	**Out of Vocabulary**
1-Hot encoding	[×]	[×]	[×]	[×]
BoW	[×]	[×]	[×]	[×]
TF	[×]	[×]	[×]	[×]
TF-IDF	[×]	[×]	[×]	[×]
Word2vec	[✓]	[✓]	[×]	[×]
GloVe	[✓]	[✓]	[×]	[×]
FastText	[✓]	[✓]	[×]	[✓]
Context2Vec	[✓]	[✓]	[✓]	[✓]
CoVe	[✓]	[✓]	[✓]	[×]
ELMo	[✓]	[✓]	[✓]	[✓]

there are still some limitations that they are not capable of addressing such as they are unable to handle polysemy issues, because they assign the same vector to word and ignores its context. Also, models such as Word2vec and GloVe assign a random vector to a word that they did not encounter during training, which means they are unable to handle OOV words that were solved by FastText, which breaks words into n-grams. All of these limitations degrade the performance of text classification. Moreover, all of the current SOTA methods do not perform well in the case of the low-quality text.

- **Universal Language Model Fine-Tuning (ULMFiT):** Presented by Jeremy Howard of fast.ai and Sebastian Ruder of the NUI Galway Insight Center, Universal Language Model Fine-tuning (ULMFiT) [52] is basically a method to allow transfer learning and achieve excellent performance for any NLP task, without training models from scratch. ULMFiT proposed two new methods within the network layers, **Discriminative Fine-tuning (Discr)** and **Slanted Triangular Learning Rates (STLR)** to enhance the Transfer Learning process. This approach includes fine-tuning a pre-trained LM, trained on the dataset of Wikitext 103, to a new dataset in such a way that it does not neglect what it has learned before. UMFiT was based on the SOTA LM at that time, which is LSTM-based model. The architecture and training method, ULMFiT, builds on similar approaches of CoVE and ELMo. In CoVe and ELMo, the encoder layers are frozen. ULMFiT instead describes a way to train all layers, and does so without over-fitting or running into "catastrophic forgetting," which has been more of a problem for NLP (vs. Computer vision) transfer learning in part, because NLP models tend to be relatively shallow. Table 2 presents the gap analysis of Classic, Non-contextual, Contextual (Context2Vec, Cove and ELMo) LMs.
ULMFiT follows three steps to get the good results on downstream tasks, i.e., (i) General LM pre-training, (ii) Target task LM fine-tuning, and (iii) Target task classifier fine-tuning. Three training stages of ULMFiT are shown in Figure 6.
The LM pre-training is unsupervised, as the unlabeled text datasets are numerous, and the pre-training can be expanded up as much as possible. It still has, however, a reliance on task-customized models. Therefore, the enhancement is only gradual as looking for a better model architecture for each role remains non-trivial until the transformer-based models that are discussed below come into being.
- **Transformer-based Pre-trained Language Models:** Transformer [162] has been proven to be more efficient and faster than LSTM or CNN for language modelling, and thus the following advances in this domain will rely on this architecture.

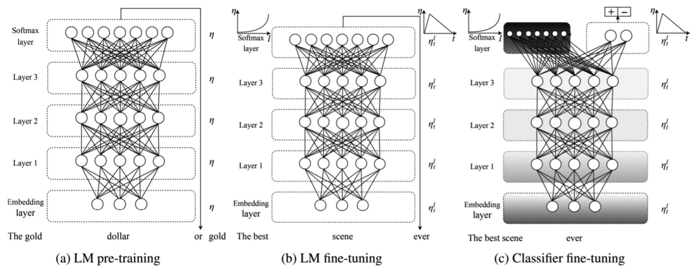

Fig. 6. Working principle of ULMFiT
(Image taken from [52]).

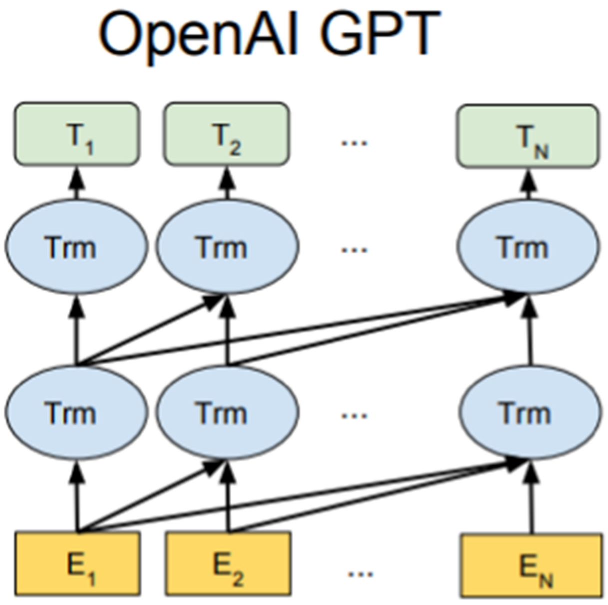

Fig. 7. Working principle of GPT
(Image taken from [132]).

- **GPT (OpenAI Transformer):** Generative Pre-Training, GPT [132], is the first Transformer-based pre-trained LM that can effectively manipulate the semantics of words in terms of context. By learning on a massive set of free text corporas, GPT extends the unsupervised LM to a much larger scale. Unlike ELMo, GPT uses the decoder of the transformer to model the language, as it is an auto-regressive model where the model predicts the next word according to its previous context. GPT has shown good performance on many downstream tasks. One drawback of GPT is it is uni-directional, i.e., the model is only trained to predict the future left-to-right context. The overall model of GPT is shown in Figure 7.
- **Bidirectional Encoder Representations from Transformers (BERT):** As seen in Figure 8, Bidirectional Encoder Representations from Transformers (BERT) is a direct descendant of GPT: train a huge LM on free text and then fine-tune individual tasks without custom network architectures. BERT [30] is another contextualized word representation

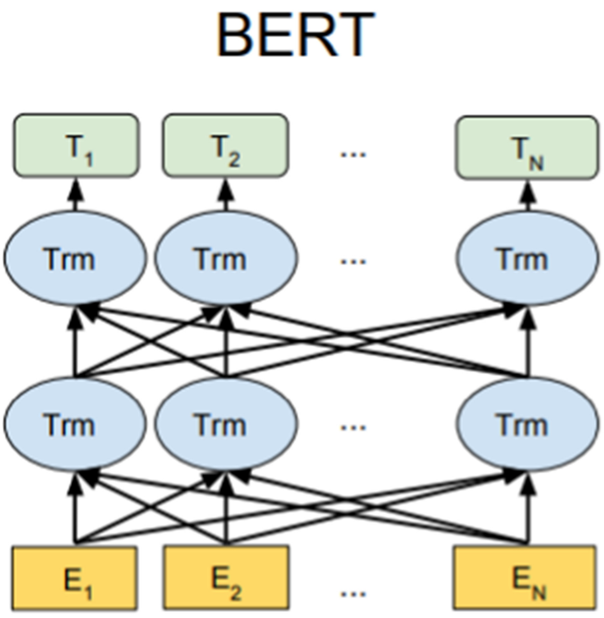

Fig. 8. Working principle of BERT
(Image taken from [30]).

LM, where the transformer NN uses parallel attention layers rather than sequential recurrence [162].

Instead of the basic language task, BERT is trained with two tasks to encourage bi-directional prediction and sentence-level understanding. BERT is trained on two unsupervised tasks: (1) a **masked language model (MLM)**, where 15% of the tokens are arbitrarily masked (i.e., replaced with the "[MASK]" token), and the model is trained to predict the masked tokens, (2) a **next sentence prediction (NSP)** task, where a pair of sentences are provided to the model and trained to identify when the second one follows the first. This second task is intended to collect additional information that is long-term or pragmatic.

BERT is trained in the dataset of Books Corpus [170] and English Wikipedia text passages. There are two BERT pre-trained model available: BERT-Base and BERT-Large. BERT can be used on un-annotated data or fine-tuned on one's task-specific data straight from the pre-trained model. The publicly accessible pre-trained model and fine-tuning code are available online.[3]

- **BERT Variants:** Recent research also explores and strengthens the goal and architecture of BERT. Some of them are briefly discussed below:
- **GPT2:** The OpenAI team released a scaled-up variant of GPT in 2019 with GPT2 [132]. It incorporates some slight improvements compared to the previous concerning the position of layer normalization and residual relations. Overall, there are four distinct GPT2 variants with the smallest being identical to GPT, the medium one being similar in size to BERT-LARGE and the xlarge one being released with 1.5B parameters as the actual GPT2 standard.

[3]https://github.com/google-research/bert.

- **XLNet:** XLNet, also known as Generalized Auto-regressive Pre-training for Language Understanding [169], which proposes a new task to predict the bidirectional context instead of the masked Language task in BERT, is a permutation language in which we make some permutations of each sentence so the two contexts will be taken into consideration. To maintain the position information of the token to be expected, authors employed two-stream self-attention. XLNET was presented to overcome the issue of pre-training fine-tune discrepancy and to include bidirectional contexts simultaneously.
- **RoBERTa:** RoBERTa: A Robustly Optimized BERT Pre-training Approach was implemented in July 2019 [86]; it is like a light version of BERT, but it has fewer parameters and better performance as it removes the training on the sentence classification task. RoBERTa made following changes to the BERT model: (1) Longer training of the model with larger batches and more data; (2) Eliminating the NSP goal; (3) Longer sequence training; (4) Dynamically during pre-training, the masked roles will change. All these changes boost the model's efficiency and make it efficient with XLNet 's previous SOTA results.
- **ALBERT:** Despite this success, BERT has some limitations, such as a huge number of parameters, which is the cause for problems such as degraded pre-training time, memory management issues, model degradation, and so on [72]. These issues are very well addressed in ALBERT, which is modified based on the architecture of BERT and proposed by Lan et al. [72]. In scaling pre-trained models, ALBERT implements two-parameter reduction methods that lift the essential barriers: (i) factorized embedding parameterization, which decomposes the big vocabulary embedding matrix to two small matrices and (ii) replaces the NSP loss by SOP loss; and (iii) cross-layer parameter sharing, which stops the parameter from prospering with the network depth. These methods significantly lower the number of parameters used when compared with BERT without significantly affecting the performance of the model, thus increasing parameter-efficiency. An ALBERT configuration is the same as BERT (large) has 18 times less parameters and can be trained about 1.7 times faster. ALBERT establishes new SOTA results while having fewer parameters compared to BERT.
- **Other Models:** Some of the other recently proposed LMs are a cross-lingual LM Pre-training (**XLM**) [71] from Facebook enhanced BERT for Cross-lingual LM. Two unsupervised training objectives that only include monolingual corporations were introduced by XLM authors: **Causal Language Modeling (CLM)** and **Masked Language Modeling (MLM)** and demonstrated that both the CLM and MLM approaches have powerful cross-lingual features that can be used for pre-training models. Similarly, **StructBERT** [164] implemented a structural objective word that randomly permits the reconstruction order of 3-grams—and a structural objective sentence that predicts the ordering of two consecutive segments.
  DistilBERT [141], a distilled version of BERT, reduces the size of a BERT LM by 40% while retaining 97% of its language understanding proficiency and being 60% quicker. **MegatronLM** [145], a scaled-up transform-based model, 24 times larger than BERT, training multi-billion parameter LMs using model parallelism. **CRTL** [64], a Conditional Transformer Language Model for Controllable Generation, is a 1.63 billion-parameter conditional transformer LM, and it is a conditional generative model. Another recently proposed model, **ERNIE** [152], enhanced representation through knowledge integration, used knowledge masking techniques including entity-level masking and phrase-level masking instead of randomly masking tokens. Authors of ERNIE extended their work and presented **ERNIE 2.0** [153], further incorporating more pre-training tasks, such as semantic

closeness and discourse relations. **SpanBERT** [62] generalized ERNIE to mask random spans, without indicating to external knowledge.
Other prominent LM includes **UniLM:** [32], which uses three objective functions: (i) **language modelling (LM)**, (ii) **masked language modelling (MLM)**, and (iii) **sequence-to-sequence language modelling (seq2seq LM)**, for pre-training a transformer model. To monitor what context the prediction conditions are in, UniLM uses special self-attention masks. **ELECTRA** [26] proposed more better pre-training techniques as compared to BERT. Authors of ELECTRA replaced some of the input tokens with their plausible substitute samples from small generator network rather than corrupting some positions of the inputs with [MASK]. ELECTRA trains a discriminator to determine whether or not each token has been substituted by a generator in the corrupted input that can be used for fine-tuning in downstream tasks. MASS [149] is another recently proposed LM. To pre-train sequence-to-sequence models, MASS uses masked sequences and adopts an encoder-decoder system and expands the MLM objective. Without pre-training or with other pre-training approaches, MASS makes substantial improvements over baselines on a range of zero/low-resource language generation tasks, including neural **machine translation (MT)**, text summarization, and conversational response generation.

- **Text-to Text Transfer Transformer (T5):** Authors in [133] unified natural language understand and generation by transforming the data to the format of text-to-text and apply the framework of an encoder-decoder. In terms of pre-training objectives, architectures, pre-training datasets, and transfer techniques, T5 has implemented a novel pre-training corpus and also systematically contrasts previously proposed methods. T5 adopts a text infilling objective, more extended training and multi-task pre-training. For fine-tuning, T5 uses the token vocabulary of the decoder as the prediction labels.
- **BART:**[80]**:** For pre-training sequence-to-sequence models, BART added additional noise functions beyond MLM. First, using an arbitrary noise function, the input sequence is distorted. Then, a transformer network reconstructs the corrupted input. BART explores a broad range of noise functionality, including token functions, masking, deletion of tokens, text infilling, rotation of documents, and shuffling of words. The best performance is attained by using both sentence shuffling and text infilling. BART matches RoBERTa's performance on GLUE and SQuAD and attain SOTA results on a number of tasks for generating text.

Although these models were able to solve context issues, they are trained on general domain corpora such as Wikipedia, which limits their applications to specific domains or tasks. To enhance the performance in sub-domains, domain-specific transformer-based models have been proposed. Some of the most famous in the biomedical domain are Sci-BERT [9], BioBERT [79], and BioALBERT [109]. Recently, other domain-specific models such as BERTTweet [117], COVID Twitter BERT (CT-BERT) [101] have been trained on datasets from Twitter. Domain-specific models were shown to be useful replacements for LMs trained on general corpus for various downstream tasks. In Table 3, we present the architecture, objective function, and dataset used for training in LMs.

3.3 Related Work on Word Representation Methods

Below, we present some relevant studies where different word representation models have been employed for various text classification tasks.

Pang et al. [120] performed that binary classification task on IMDb dataset and employed unigrams, bigrams, and POS tags as features. For classification, they used SVM, Maximum entropy, and NB classifiers in their study and found out that best results were achieved with unigrams as feature

Table 3. A Comparison of Popular Language Models

LMs	Release Date	Architecture	Pre-Training Task	Corpus Used
Word2vec	Jan-13	FCNN	-	Google News
GloVe	Oct-14	FCNN	-	Common Crawl corpus
FastText	Jul-16	FCNN	-	Wikipedia
ELMo	Feb-18	BiLSTM	BiLM	Wiki-Text-103
GPT	Jun-18	Transformer Decoder	LM	Book-Corpus
GPT-2	Jun-18	Transformer Decoder	LM	Web-Text
BERT	Oct-18	Transformer Encoder	MLM & NSP	WikiEn+Book-Corpus
RoBERTa	Jul-19	Transformer Encoder	MLM & NSP	Book-Corpus + CC-News +Open-Web-Text+ STORIES
ALBERT	Sep-19	Transformer Encoder	MLM+SOP	same as BERT
XLNet	Jun-19	Auto-regressive Transformer Encoder	PLM	WikiEn+ Book-Corpus+Giga5 + Clue-Web + Common Crawl
ELECTRA	2020	Transformer Encoder	RTD+MLM	same as XLNet
UniLM	2020	Transformer Encoder	MLM+NSP	WikiEn + Book-Corpus
MASS	2020	Transformer	Seq2Seq MLM	*Task-dependent
BART	2020	Transformer	DAE	same as RoBERTa
T5	2020	Transformer	Seq2Seq MLM	Colossal Clean Crawled Corpus (C4)

and SVM as classifier. Kwok and Wang [70] used n-grams features along with NB classifier tweets classification. These legacy-based word representation methods such as n-grams, BoW, TF, and TF-IDF have been widely used in different studies for various text classification tasks [29, 43, 68, 84]. These traditional methods for text classification are simple, computationally economical. However, their limitations such as ignoring word order, unable to capture semantic information and high dimensionality, and so on, restrict their use for efficient text classification tasks.

Later, representation learning methods of learning text representation directly using neural network [27] was adopted, which improved classification results. Word embeddings from continuous word representation models such as Word2vec and GloVe are the most famous and widely used ones among these methods because of low dimensionality of vectors and capture semantic relationships. Word representation models have also been used for sentence-level classification task by averaging word vectors as feature representation that is utilized later on as input for sentence-level classification [21].

Word embeddings that are created based on unigrams and by averaging embeddings are not able to capture the issue of syntactic dependencies such as "but" and "negations" can change the complete meaning of a sentence and long dependencies within a sentence [21]. Sochet et al. [147] proposed a recursive neural network that can capture and model long semantic and sentiment dependencies of words and sentence at different stages. The disadvantage of this method is that it depends on parsing, which makes it challenging to use on Twitter-related text [37]. A paragraph representation model solved this issue learns word vectors and does not reply on parsing [75]. Both of these, recursive neural and paragraph representation models, have assessed on IMDb dataset used by Pang et al. [121], and both models improved the classification results obtained by using BoW features.

Deep neural network-based methods have also been used for Text classification tasks. Tang et al. [158] proposed sentiment-specific word representation model, which are achieved from emoticons labelled tweet messages with the help of the neural network. Severyn and Moschitti [144] presented another neural network-based model where they used Word2vec to learn embedding. Tweets are presented as a matrix wherein which columns compare with words, thus retaining the position they appear in a tweet. Emoticons annotated data was utilized to pre-train the weights and then trained by hand-annotated from SemEval contestant. The experimental results tell that pretraining step enables for a better initialization of the networks' weights and therefore, has a positive role in classification results. In another study conducted by Fu et al. [38], Word2vec was employed to get word representation that was forwarded to the recursive encoder as an input for text classification. Ren et al. [135] also used Word2vec to generate word representations and proposed a new model for the Twitter classification task. Lauren et al. [74] presented a different document representation model where they used the skip-gram algorithm for generating word representations for the classification of clinical texts.

Due to the limitations and restrictions in a few corpora, pre-trained word embeddings are preferred by researchers as an input of ML models. Naseem et al. [128] used pre-trained Word2vec embeddings and forwarded these word embeddings to CNN. Similarly, Kim [66] utilized pre-trained embeddings of Word2vec and forwarded to CNN neural network, which increased the classification results. Camacho et al. [18] for concept representation in their work. Jianqiang and Xiolin [60] have initialized word embeddings using pre-trained GloVe embeddings in their DCNN model. Similarly, Wallace [163] applied GloVe, and Word2vec pre-trained word embedding in deep neural network-based algorithms and enhanced the classification results. Similarly, a study conducted by Wang et al. [166] used pre-trained GloVe embeddings as an input to LSTM with attention model for aspect-based classification, and Liu et al. [85] employed pre-trained word embeddings Word2vec for recommending idioms in essay writing. Recently, Ilic et al. [56] have used contextual word embeddings (ELMo) for word representation for the detection of sarcasm and irony and shown that using ELMo word representations have improved the classification results. The research community has made limited efforts for solving the above mention limitations of continuous word representation models by proposing different models. For example, for handling OOV words that are not seen in the training and they are assigned UNK token and same vector for all words and degrades results if the number of OOV is large. Different methods to handle OOV words have been proposed in different studies [31, 48, 127]. But still these models does not capture the polysemy issues. This issue of words with different meanings (polysemy) is addressed in different models presented by References [55, 116]. In recent days, researchers has presented more robust models to handle OOV words and polysemy issues [83, 91, 93, 126].

To handle domain-specific problems, different studies have been conducted where researchers used existing knowledge encoded in semantic lexicons to these word embeddings to improve the downsides of using the pre-trained embedding that is trained on news data that is usually different from the data we use in our tasks. Some of the models presented are proposed in the following studies that inject external knowledge in existing word embedding and improves the results [36, 99, 118, 143, 151]. Word embeddings are beneficial in different areas beyond NLP, such as link prediction, information retrieval, and recommendation systems. Ledell et al. [78] proposed a model that is suitable for many of the applications mentioned above and acted as a baseline. None of the above mentioned is robust enough and fails to integrate sentiment of words in the representations and does not work well in domain-specific tasks such as sentiment analysis and so on.

Studies show that adding sentiment information into conventional word representation models improves performance. To integrate the sentiment information into word embeddings, researchers have proposed different hybrid word representations by changing existing skip-gram model [160]. Tang et al. [159] proposed several **hybrid ranking models (HyRank)** and developed sentiment embeddings based on C&W, that considers context and sentiment polarity of tweets. Similarly, several other models are presented, which considers context and sentiment polarity of words for sentiment analysis [81, 137, 157]. Yu et al. [81] proposed sentiment embeddings by refining pre-trained embeddings Re(*) using the intensity score of external knowledge resource. Rezaeinia et al. [138] proposed **improved word vectors (IWV)** by combining word embeddings, **part of speech (POS)** and combination of lexicons for sentiment analysis. Recently, Cambria et al. [19] proposed context embeddings for sentiment analysis by conceptual primitives from text and linked with commonsense concepts and named entities.

Recent studies have used these contextual and transformer-based LMs in their model in various NLP tasks. Furthermore, various studies have been presented that use the domain-specific LMs for different NLP tasks. These hybrid and domain-specific LMs have improved the performance and ability to capture complex word attributes, such as semantics, OOV, context, and syntax, into account in various NLP tasks.

4 CLASSIFICATION TECHNIQUES

Choosing an appropriate classifier is one of the main steps in the text classification task. Without having a comprehensive knowledge of every algorithm, we cannot find out the most effective model for the text classification task. Out of many ML algorithms used in text classification, we will present some famous and commonly used classification algorithms. These are used for sentiment classification tasks such as **Naive Bayes (NB)**, **Support vector machine (SVM)**, **logistic regression (LR)**, Tree-based classifiers such as **decision tree (DT)** and **random forest (RF)** and neural network-based (DL) algorithms. Table 4 presents the pros and cons of classification algorithms.

4.1 ML-based Classifiers

- **Naive Bayes (NB) classifiers:** The Naive Bayes (NB) classifiers are a group of different classification algorithms that are based on Bayes theorem, presented by Thomas Bayes [49]. All Naive Bayes algorithms have the same assumption, i.e., each pair of features being classified is independent of others. The NB classification algorithms are widely used for information retrieval [129] and many text classification tasks [95, 119]. Naive Bayes classifiers are called "Naive" because they consider that every feature is independent of other features in the input. Whereas, in reality, words and phrases in the sentences are highly interrelated. The meaning and sentiment depend on the position of words in the sentence, which can change if the position is changed.
  NB classifiers are derived from Bayes' theorem, which states that given the number of documents (n) to be classified into z classes where $z \in \{x_1, x_2, \ldots x_z\}$ the predicted label out is $x \in X$. The Naive Bayes theorem is given as follows:
  $$P(x|y) = \frac{P(y|x)P(x)}{P(y)},$$
  where y denotes a document and x refers to the classes. In simple words, the NB algorithm will take each word in the training data and will calculate the probability of that word being classified. Once the probabilities of every word are calculated, then classifier is read to classify new data by utilizing the prior calculated probabilities during the training phase. Advantages of NB classifiers are: They are scalable, more suitable when the dimension of input is high, its implementation is simple, less computationally expensive, works well when less training data is available, and can often outperform other classification algorithms. Whereas the disadvantages are: NB classifiers make a solid and reliable hypothesis on the shape of data distribution, i.e., any two features are independent given the output class, which gives bad results [148, 167]. Another limitation of NB classifiers is due to data scarcity. For any value of the feature, we have to approximate the likelihood value by a frequentist.

Table 4. Comparison of Classification Algorithms

Classifiers	Pros	Cons
NB	(i) Less computational time. (ii) Easy to understand & implement. (iii) Can easily be trained on less data.	(i) Relies strongly on the class features independence. and does not perform well if the condition is not met. (ii) Issue of zero conditional probability for zero. frequency features that makes total probability zero.
SVM	(i) Effective in higher dimension. (ii) Can model non-linear decision boundary. (iii) Robust to the issue of overfitting.	(i) More computational time for large datasets. (ii) Kernel selection is difficult. (iii) Does not perform well in case of overlapped classes.
LR	(i) Easy and simple to implement. (ii) Less computationally expensive. (iii) Does not need tuning and features to be uniformly distributed.	(i) Fails in case of non-linear problems. (ii) Need large datasets. (iii) Predict results on the basis of independent variables.
DT	(i) Interpretable and easy to understand. (ii) Less pre-processing required. (iii) Fast and almost zero hyper-parameters to tune.	(i) High chances of overfitting. (ii) Less prediction accuracy as compared to others. (iii) Complex calculation in large number of classes.
RF	(i) Fast to train, flexible, and gives high results . (ii) Less variance than single DT. (iii) Less pre-processing required.	(i) Not easy and simple to interpret. (ii) Require more computational resources. (iii) Require more time to predict as compared to others.
DL	(i) Fast predictions once training is complete. (ii) Works well in case of huge data. (iii) Flexible architecture, can be utilized for classification and regression tasks.	(i) Require a large amount of data. (ii) Computationally expensive and time-consuming. (iii) DL-based classifiers are like black-box (issue of model interpretability exists).

- **Support vector machine (SVM):** The support vector machine (SVM) classifiers are some of the famous and common used algorithms used for text classification due to their good performance. SVM is a non-probabilistic binary linear classification algorithm that performs by plotting the training data in multi-dimensional space. Then SVM categorizes the classes with a hyper-plane. The algorithm will add a new dimension if the classes can not be separated linearly in multi-dimensional space to separate the classes. This process will continue until a training data can be categorized into two different classes.
  The advantage of SVM classifiers is that results are obtained by using SVM are usually better. The disadvantage of SVM algorithms is that they are not easy to choose a suitable kernel, long training time in case of extensive data and more computational resources are required, and so on.
- **Logistic Regression (LR) classifier:** Logistic regression (LR) is a statistical model and is one of the earliest techniques used for classification. LR predicts probabilities rather than classes [35, 39] or existence of an event such as a win/lose or healthy/sick, and so on. This can be expanded to model many classes of events like deciding whether an image consists of a cat, duck, cow, and so on. Every object being identified in the image would be given a probability between 0 and 1, and the sum adding to one. LR predicts the results based on the set of independent values. However, if the wrong independent values are added, then the model will not predict good results. It works well in the case of categorical results but fails in the case of continuous results. Also, LR wants that every data point to be independent of all others, but if the findings are interlinked to one another, then classifier will not predict good results.
- **Decision Tree (DT) classifier:** Decision tree (DT) was presented by Magerman [87] and developed by Quinlan [131]. It is one of the earliest classification models for text and data mining and is employed successfully in different areas for classification task [98]. The main intuition behind this idea was to create tree-based attributes for data points, but the major

question is which feature could be a parent and which will be a child's level. DT classifier design contains a root, decision, and leaf nodes that denote dataset, carry out the computation, and performs classification, respectively. During the training phase, the classifier learns the decision need to be executed to separate labelled categories. To classify the unknown instance, the data is passed through the tree. A particular feature from the input text is matched with the fixed that was known during the training stage. The calculation at each decision node compares the chosen features with this fixed feature earlier; the decision relies on whether the feature is more prominent than or less than the fixed, which creates two-way division in the tree. The text will eventually go over these decision nodes until it reaches the leaf node that describes it assigned class.
The advantages of DT classifier are: the amount of hyper-parameters that require tuning is nearly zero, easy to describe, can be understood easily by its visualizations, whereas the significant disadvantages of DT classifier are: it is sensitive to a minor change in the data [42] and has a probability of overfitting [130], complex computations in case of a large number of class labels, and has difficulties with-out-of sample prediction.

- **Random Forest (RF) Classifier:** Random forest, which is also called an ensembles learning technique for text classification that concentrates on methods to compare the results of several trained models in line to give better classifier and performance than a single model. Ho [50] proposed RF classifier, which is simple to understand and also gives better results in classification. RF classifier is composed of the number of DT classifiers where every tree is trained by a bootstrapped subset of the training text. An arbitrary subset of the characteristics is selected at every decision node, and the model will only examine part of these features. The primary issue with utilizing the single tree is that it has massive variation so that the arrangement of the training data and features can impact its results.
  This classifier is quick to train for textual data but slow in giving predictions when trained [7]. It performs well with both categorical and continuous variables, can automatically handle missing values, is robust to outliers and less affected by noise, whereas training a vast number of trees can be computationally expensive, require more training time and utilize much memory.

4.2 Deep learning-based classifiers

DL-based models are motivated by the working of the human brain. It has attained SOTA results in many different areas [108, 134] including NLP. It requires a large number of training data to achieve a semantically good representation of textual data. DL models have attained excellent results compared to models on different classification tasks. Main architectures of DL that are commonly used in any text classification task are briefly discussed below.

- **Recurrent Neural Network (RNN):** RNN is one of the popular neural network-based models that is widely used for different text classification tasks [88, 154]. Previous data points of a sequence are assigned more weights in an RNN model, which makes it more useful and better for any text, string, or sequential data classification. RNN models deal with data from previous layers/nodes in such a good way that makes them superior for semantic analysis of a corpus. **Gated recurrent unit (GRU)** and **long short term memory (LSTM)** are the most common types of RNNs that are used of text classification.
  The one drawback of RNNs is that they are sensitive to gradient vanishing problem and exploding gradient when gradient descent's error is back-propagated [12].
- **Long Short-Term Memory (LSTM):** LSTM was presented by Hochreiter and Schmidhuber [51]. LSTM was presented to address the gradient descent issues of RNN by keeping the long-term dependency in a better way as compared to RNNs. It is more effective to overcome the

issues of vanishing gradient [122]. Even though LSTMs have an architecture like a chain that is same as RNNs but it uses different gates, which handles the volume of information carefully, which is allowed from each node state. The role of each gate and node in a basic LSTM cell is explained below:
$i_t = \sigma(W_i[x_t, ht - 1] + bi)$,
$\widehat{C}_t = tanh(W_c[x_t, ht - 1] + b_c)$,
$f_t = \sigma(W_f[x_t, h_{t-1}] + b_f)$,
$C_t = i_t * \widehat{C}_t + f_t C_{t-1}$,
$o_t = \sigma(W_o)[x_t, h_{t-1}] + b_o$,
$h_t = o_t tanh(C_t)$,
where i_t, $\widehat{C}_t$, and f_t denote input gate, candid memory cell, and forget gate activation, respectively. Whereas C_t computes new memory cell value and o_t andh_t represents the final output gate. b is bias vector, W denotes weight matrix and x_t denotes input to the memory cell at time t.

- **Gated Recurrent Unit (GRU):** GRU is another type of RNN that is presented by Chung et al. [25] and Cho et al. [24]. GRU is the simplest form of LSTM architecture. However, it includes two gates and does not contain internal memory, which makes it different from LSTM. Also, in GRU, a second non-linearity (tanh) is not applied on a network. The working of a GRU cell is given below:
$z_t = \sigma_g(W_z x_t + U_z h_{t-1} + b_z)$,
$\widehat{r}_t = \sigma_g(W_r x_t + U_r h_{t-1} + b_r)$,
$h_t = z_t h_{t-1} + (1 + z_t)$,
$\sigma_h(W_h x_t + U_h(r_t h_{t-1}) + b_h)$,
where z_t denotes to the update gate of t, x_t represents input vector, W, U, and b denote parameter matrices. σ_g, which is an activation function can be ReLU or sigmoid, $\widehat{r}_t$ represents reset gate of t, h_t is the output gate of vector t, and σ_h denotes the hyperbolic tangent function.
- **Convolutional Neural Networks (CNN):** Another famous architecture of DL is CNN, which is mostly used for hierarchical classification in a DL [57]. CNN was built and used for image classification in early days, but over the period, it has shown excellent results for text classification as well [77]. In image classification, an image tensor is convolved with a set of kernels of size dxd. The convolution layers in the CNN are known as feature maps that can be stacked to have multiple filters. To overcome the computational issue due to the size of dimensionality, CNN uses pooling layer to reduce the size from one layer to the other one. Different pooling methods have been proposed by researchers to decrease the output without losing features [142].
Max pooling is the most common pooling technique where maximum elements in the pooling window are selected. Features are flattened into one column to feed the pooled output from the stacked features map to the next one. Usually, the last layer of CNNs is fully connected. Weights and feature filters are adjusted during the backpropagation step of CNN. The number of channels is the major issue with CNN's for text classification, which is very few in case of image classification. Three channels form RGB. For text, it can be a vast number that makes dimensions very high for text classification [61].

5 EVALUATION METRICS

In terms of evaluating text classification models, accuracy, precision, recall, and F1 score are the most used to assess the text classification methods. Below, we briefly discuss each of these.

Table 5. Confusion Matrix

Actual Class			
Predicted Class		Positive	Negative
	Positive	*True Positive (TP)*	*False Negative (FN)*
	Negative	*False Positive (FP)*	*True Negative (TN)*

Confusion matrix: Confusion matrix is a unique table or a method that is used to present the efficiency of the classification algorithm. In Table 5, we present the confusion matrix. Details are given below:

- **True Positives (TP):** TP are the accurately predicted positive instances.
- **True Negatives (TN):** TN are the accurately predicted negative instances.
- **False Positives (FP):** FP are wrongly predicted positive instances.
- **False Negatives (FN):** FN are wrongly predicted negative instances.

Once we understand the confusion matrix and its parameters, then we can define and understand evaluation metrics easily, briefly explained below:

- **Accuracy:** Accuracy is the simple ratio of observations predicted correctly to the total observations and is given by

$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN}.$$

- **Precision:** Precision is the ratio of **true positive (TP)** observations to the overall positive predicted values (TP+FP) and is given by

$$Precision = \frac{TP}{TP + FP}$$

- **Recall:** Recall is the ration of true positive (TP) observations to the overall observations (TP+FN) and is given by

$$Recall = \frac{TP}{TP + FN}$$

- **F1 score** - Weighted average of Recall and Precision is known as F1 score, which means F1-score consists of both FPs and FNs and is given by

$$F1 = 2 * \frac{Recall * Precision}{Recall + Precision}$$

6 APPLICATIONS

In the earliest history of ML and AI, these LMs have been widely used to extract features for text classification tasks, especially in the area of information retrieval systems. However, as technological advances have emerged over time, these have been globally used in many domains such as medicine, social sciences, healthcare, psychology, law, engineering, and so on. These LMs have been used in many different areas of text classification tasks such as Information Retrieval, Sentiment Analysis, Recommender Systems, Summarization, Question Answering, Machine Translation, Named Entity Recognition, Adversarial Attacks and Defenses, and so on, in different areas.

7 CONCLUSION

In this survey, we have introduced various algorithms that enable us to capture rich information in text data and represent them as vectors for traditional frameworks. We first discussed classical methods of text representation, which mostly involved feature engineering followed by DL-based model. DL techniques have been attracting much attention in these years, which are well known especially for their capability of addressing problems in computer vision and speech recognition areas. The great success DL achieved stems from its use of multiple layers of nonlinear processing units for learning multiple layers of feature representations of data; different layers correspond to different abstraction levels. DL methods not only show powerful capability in semantic analysis applications on text data but can be successfully used in a number of tasks of text classification and natural language processing. We discussed different word-embedding methods such as Word2vec, GloVe, FastText, and contextual words vectors such as Context2Vec, CoVe, and ELMO. Finally, in the end, we presented current different SOTA models based on the transformer trained on general corpus as well as domain-specific transformer-based LMs. These LMs are still in their developing phase, but we expect in-depth learning-based NLP research to be driven in the direction of making better use of unlabeled data. We expect such a trend to continue with more and better model designs. We expect to see more NLP applications that employ reinforcement learning methods, e.g., dialogue systems. We also expect to see more research on multi-modal learning as, in the real world, language is often grounded on (or correlated with) other signals.

REFERENCES

[1] Apoorv Agarwal, Boyi Xie, Ilia Vovsha, Owen Rambow, and Rebecca J. Passonneau. 2011. Sentiment analysis of Twitter data. https://www.aclweb.org/anthology/W11-0705.pdf.

[2] Charu C. Aggarwal and Chandan K. Reddy. 2013. *Data Clustering: Algorithms and Applications*. CRC Press.

[3] Charu C. Aggarwal and ChengXiang Zhai. 2012. A survey of text classification algorithms. In *Mining Text Data*. Springer, Boston, MA, 163–222.

[4] Edgar Altszyler, Mariano Sigman, and Diego Fernández Slezak. 2016. Comparative study of LSA vs Word2vec embeddings in small corpora: A case study in dreams database. (2016). arXiv:abs/1610.01520.

[5] Alexandra Balahur. 2013. Sentiment analysis in social media texts. In *WASSA@NAACL-HLT*.

[6] Jorge A. Balazs and Juan D. Velásquez. 2016. Opinion mining and information fusion: A survey. *Inf. Fus.* 27 (2016), 95–110.

[7] Himani Bansal, Gulshan Shrivastava, Nguyen Nhu, and L. M. Stanciu (Eds.). 2018. *Social Network Analytics for Contemporary Business Organizations*. IGI Global. DOI: https://doi.org/10.4018/978-1-5225-5097-6

[8] Yanwei Bao, Changqin Quan, Lijuan Wang, and Fuji Ren. 2014. The role of pre-processing in Twitter sentiment analysis. In *Intelligent Computing Methodologies*, De-Shuang Huang, Kang-Hyun Jo, and Ling Wang (Eds.). Springer International Publishing, Cham, 615–624.

[9] Iz Beltagy, Kyle Lo, and Arman Cohan. 2019. SciBERT: A pretrained language model for scientific text. arXiv:cs.CL/1903.10676.

[10] Yoshua Bengio, Aaron Courville, and Pascal Vincent. 2013. Representation learning: A review and new perspectives. *IEEE Trans. Pattern Anal. Mach. Intell.* 35, 8 (2013), 1798–1828.

[11] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. 2003. A neural probabilistic language model. *J. Mach. Learn. Res.* 3 (Mar. 2003), 1137–1155. Retrieved from http://dl.acm.org/citation.cfm?id=944919.944966.

[12] Y. Bengio, P. Simard, and P. Frasconi. 1994. Learning long-term dependencies with gradient descent is difficult. *Trans. Neur. Netw.* 5, 2 (Mar. 1994), 157–166. DOI: https://doi.org/10.1109/72.279181

[13] Adam Bermingham and Alan Smeaton. 2011. On using Twitter to monitor political sentiment and predict election results. In *Proceedings of the Workshop on Sentiment Analysis Where AI Meets Psychology ('11)*. Asian Federation of Natural Language Processing, 2–10. Retrieved from https://www.aclweb.org/anthology/W11-3702.

[14] Marina Boia, Boi Faltings, Claudiu Cristian Musat, and Pearl Pu. 2013. A :) Is worth a thousand words: How people attach sentiment to emoticons and words in tweets. In *International Conference on Social Computing*. 345–350.

[15] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2016. Enriching word vectors with subword information. *arXiv preprint arXiv:1607.04606* (2016).

[16] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2016. Enriching word vectors with subword information. *CoRR* abs/1607.04606 (2016).

[17] Tolga Bolukbasi, Kai-Wei Chang, James Y. Zou, Venkatesh Saligrama, and Adam T. Kalai. 2016. Man is to computer programmer as woman is to homemaker? Debiasing word embeddings. In *Conference on Advances in Neural Information Processing Systems 29*, D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett (Eds.). Curran Associates, Inc., 4349–4357. Retrieved from http://papers.nips.cc/paper/6228-man-is-to-computer-programmer-as-woman-is-to-homemaker-debiasing-word-embeddings.pdf.

[18] Jose Camacho-Collados, Mohammad Taher Pilehvar, and Roberto Navigli. 2016. Nasari: Integrating explicit knowledge and corpus statistics for a multilingual representation of concepts and entities. *Artif. Intell.* 240 (2016), 36–64. DOI: https://doi.org/10.1016/j.artint.2016.07.005

[19] Erik Cambria, Soujanya Poria, Devamanyu Hazarika, and Kenneth Kwok. 2018. SenticNet 5: Discovering conceptual primitives for sentiment analysis by means of context embeddings. In *Association for the Advancement of Artificial Intelligence Conference*.

[20] Xavier Carreras and Lluís Màrquez. 2001. Boosting trees for anti-spam email filtering. *CoRR* cs.CL/0109015 (2001).

[21] Giuseppe Castellucci, Danilo Croce, and Roberto Basili. 2015. Acquiring a large scale polarity lexicon through unsupervised distributional methods. In *Natural Language Processing and Information Systems*, Chris Biemann, Siegfried Handschuh, André Freitas, Farid Meziane, and Elisabeth Métais (Eds.). Springer International Publishing, Cham, 73–86.

[22] Arda Celebi and Arzucan Ozgur. 2016. Segmenting hashtags using automatically created training data. https://www.aclweb.org/anthology/L16-1476.pdf.

[23] Wei James Chen, Xiaoshen Xie, Jiale Wang, Biswajeet Pradhan, Haoyuan Hong, Dieu Tien Bui, Zhao Duan, and Jianquan Ma. 2017. A comparative study of logistic model tree, random forest, and classification and regression tree models for spatial prediction of landslide susceptibility. https://www.sciencedirect.com/science/article/pii/S0341816216305136.

[24] Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Conference on Empirical Methods in Natural Language Processing (EMNLP'14)*. Association for Computational Linguistics, 1724–1734. DOI: https://doi.org/10.3115/v1/D14-1179

[25] Junyoung Chung, Çaglar Gülçehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *CoRR* abs/1412.3555 (2014).

[26] Kevin Clark, Minh-Thang Luong, Quoc V. Le, and Christopher D. Manning. 2020. Electra: Pre-training text encoders as discriminators rather than generators. *arXiv preprint arXiv:2003.10555* (2020).

[27] Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *25th International Conference on Machine Learning (ICML'08)*. ACM, New York, NY, 160–167. DOI: https://doi.org/10.1145/1390156.1390177

[28] Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel P. Kuksa. 2011. Natural language processing (almost) from scratch. *CoRR* abs/1103.0398 (2011).

[29] Thomas Davidson, Dana Warmsley, Michael W. Macy, and Ingmar Weber. 2017. Automated hate speech detection and the problem of offensive language. *CoRR* abs/1703.04009 (2017).

[30] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: Pre-training of deep bidirectional transformers for language understanding. *CoRR* abs/1810.04805 (2018).

[31] Bhuwan Dhingra, Hanxiao Liu, Ruslan Salakhutdinov, and William W. Cohen. 2017. A comparative study of word embeddings for reading comprehension. *CoRR* abs/1703.00993 (2017).

[32] Li Dong, Nan Yang, Wenhui Wang, Furu Wei, Xiaodong Liu, Yu Wang, Jianfeng Gao, Ming Zhou, and Hsiao-Wuen Hon. 2019. Unified language model pre-training for natural language understanding and generation. In *Conference on Advances in Neural Information Processing Systems*. 13063–13075.

[33] Cícero Nogueira dos Santos and Maíra A. de C. Gatti. 2014. Deep convolutional neural networks for sentiment analysis of short texts. In *International Conference on Computational Linguistics*.

[34] Ábel Elekes, Adrian Englhardt, Schäler, and Klemens Böhm. 2018. Toward meaningful notions of similarity in NLP embedding models. *Int. J. Dig. Libr.* (04 2018). DOI: https://doi.org/10.1007/s00799-018-0237-y

[35] Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. 2008. LIBLINEAR: A library for large linear classification. *J. Mach. Learn. Res.* 9 (June 2008), 1871–1874. Retrieved from http://dl.acm.org/citation.cfm?id=1390681.1442794.

[36] Manaal Faruqui, Jesse Dodge, Sujay Kumar Jauhar, Chris Dyer, Eduard H. Hovy, and Noah A. Smith. 2014. Retrofitting word vectors to semantic lexicons. *CoRR* abs/1411.4166 (2014).

[37] Jennifer Foster, Özlem Çetinoğlu, Joachim Wagner, Joseph Le Roux, Joakim Nivre, Deirdre Hogan, and Josef van Genabith. 2011. From news to comment: Resources and benchmarks for parsing the language of Web 2.0. In *Proceedings of 5th International Joint Conference on Natural Language Processing*. Asian Federation of Natural Language Processing, 893–901. Retrieved from https://www.aclweb.org/anthology/I11-1100.

[38] Xianghua Fu, Wangwang Liu, Yingying Xu, and Laizhong Cui. 2017. Combine HowNet lexicon to train phrase recursive autoencoder for sentence-level sentiment analysis. *Neurocomputing* 241 (2017), 18–27.

[39] Alexander Genkin, David D. Lewis, and David Madigan. 2007. Large-scale Bayesian logistic regression for text categorization. *Technometrics* 49, 3 (2007), 291–304. DOI: https://doi.org/10.1198/004017007000000245

[40] Anastasia Giachanou, Julio Gonzalo, Ida Mele, and Fabio Crestani. 2017. Sentiment propagation for predicting reputation polarity. DOI: https://doi.org/10.1007/978-3-319-56608-5_18

[41] Kevin Gimpel, Nathan Schneider, Dipanjan Das, Daniel Mills, Jacob Eisenstein, Michael Heilman, Dani Yogatama, Jeffrey Flanigan, and Noah A. Smith. [n.d.]. Part-of-Speech Tagging for Twitter: Annotation, Features, and Experiments. https://www.aclweb.org/anthology/P11-2008.pdf.

[42] Christian Giovanelli, Xin U. Liu, Seppo Antero Sierla, Valeriy Vyatkin, and Ryutaro Ichise. 2017. Towards an aggregator that exploits big data to bid on frequency containment reserve market. In *43rd Conference of the IEEE Industrial Electronics Society*. 7514–7519.

[43] Edel Greevy. 2004. Automatic text categorisation of racist webpages. http://doras.dcu.ie/17275/1/edel_greevy_20120702122736.pdf.

[44] Vishal Gupta and Gurpreet Lehal. 2009. A survey of text mining techniques and applications. *J. Emerg. Technol. Web Intell.* 1 (08 2009). DOI: https://doi.org/10.4304/jetwi.1.1.60-76

[45] Emma Haddi, Xiaohui Liu, and Yong Shi. 2013. The role of text pre-processing in sentiment analysis. In *International Conference on Information Technology and Quantitative Management*.

[46] Khaled M. Hammouda and Mohamed S. Kamel. 2004. Efficient phrase-based document indexing for web document clustering. *IEEE Trans. Knowl. Data Eng.* 16, 10 (Oct. 2004), 1279–1296. DOI: https://doi.org/10.1109/TKDE.2004.58

[47] Yulan He, Chenghua Lin, and Harith Alani. 2011. Automatically extracting polarity-bearing topics for cross-domain sentiment classification. In *49th Meeting of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, 123–131. Retrieved from https://www.aclweb.org/anthology/P11-1013.

[48] Aurélie Herbelot and Marco Baroni. 2017. High-risk learning: Acquiring new word vectors from tiny data. *CoRR* abs/1707.06556 (2017).

[49] Bruce M. Hill. 1968. Posterior distribution of percentiles: Bayes' theorem for sampling from a population. *J. Amer. Statist. Assoc.* 63, 322 (1968), 677–691. Retrieved from http://www.jstor.org/stable/2284038.

[50] Tin Kam Ho. 1998. The random subspace method for constructing decision forests. *IEEE Trans. Pattern Anal. Mach. Intell.* 20, 8 (Aug. 1998), 832–844. DOI: https://doi.org/10.1109/34.709601

[51] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Comput.* 9, 8 (Nov. 1997), 1735–1780. DOI: https://doi.org/10.1162/neco.1997.9.8.1735

[52] Jeremy Howard and Sebastian Ruder. 2018. Universal language model fine-tuning for text classification. In *56th Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, 328–339. DOI: https://doi.org/10.18653/v1/P18-1031

[53] Xia Hu and Huan Liu. 2012. *Text Analytics in Social Media*. Springer US, Boston, MA, 385–414. DOI: https://doi.org/10.1007/978-1-4614-3223-4_12

[54] Ah hwee Tan. 1999. Text mining: The state of the art and the challenges. In *Workshop on Knowledge Discovery from Advanced Databases*. 65–70.

[55] Ignacio Iacobacci, Mohammad Taher Pilehvar, and Roberto Navigli. 2015. SensEmbed: Learning sense embeddings for word and relational similarity. In *Meeting of the Association for Computational Linguistics*.

[56] Suzana Ilic, Edison Marrese-Taylor, Jorge A. Balazs, and Yutaka Matsuo. 2018. Deep contextualized word representations for detecting sarcasm and irony. *CoRR* abs/1809.09795 (2018).

[57] Max Jaderberg, Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. 2014. Reading text in the wild with convolutional neural networks. *CoRR* abs/1412.1842 (2014).

[58] Zhao Jianqiang. 2015. Pre-processing boosting Twitter sentiment analysis? DOI: https://doi.org/10.1109/SmartCity.2015.158

[59] Zhao Jianqiang and Gui Xiaolin. 2017. Comparison research on text pre-processing methods on Twitter sentiment analysis. *IEEE Access* 5 (2017), 2870–2879.

[60] Zhao Jianqiang and Gui Xiaolin. 2018. Deep convolution neural networks for Twitter sentiment analysis. *IEEE Access* PP (01 2018), 1–1. DOI: https://doi.org/10.1109/ACCESS.2017.2776930

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

[62] Mandar Joshi, Danqi Chen, Yinhan Liu, Daniel S. Weld, Luke Zettlemoyer, and Omer Levy. 2020. SpanBERT: Improving pre-training by representing and predicting spans. *Trans. Assoc. Comput. Ling.* 8 (2020), 64–77.

[63] Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2016. Bag of tricks for efficient text classification. *CoRR* abs/1607.01759 (2016).

[64] Nitish Shirish Keskar, Bryan McCann, Lav R. Varshney, Caiming Xiong, and Richard Socher. 2019. CTRL: A conditional transformer language model for controllable generation. arXiv:cs.CL/1909.05858.
[65] Farhan Hassan Khan, Saba Bashir, and Usman Qamar. 2014. TOM: Twitter opinion mining framework using hybrid classification scheme. *Decis. Support Syst.* 57 (Jan. 2014), 245–257. DOI: https://doi.org/10.1016/j.dss.2013.09.004
[66] Yoon Kim. 2014. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882* (2014).
[67] Vandana Korde and C. Namrata Mahender. 2012. Text classification and classifiers: A survey. http://www.airccse.org/journal/ijaia/papers/3212ijaia08.pdf https://www.researchgate.net/publication/276196340_Text_Classification_and_ClassifiersA_Survey.
[68] Efthymios Kouloumpis, Theresa Wilson, and Johanna D. Moore. 2011. Twitter sentiment analysis: The good the bad and the OMG! In *International AAAI Conference on Web and Social Media.*
[69] Kamran Kowsari, Kiana Jafari Meimandi, Mojtaba Heidarysafa, Sanjana Mendu, Laura E. Barnes, and Donald E. Brown. 2019. Text classification algorithms: A survey. *CoRR* abs/1904.08067 (2019).
[70] Irene Kwok and Yuzhou Wang. 2013. Locate the hate: Detecting tweets against blacks. In *27th AAAI Conference on Artificial Intelligence (AAAI'13).* AAAI Press, 1621–1622. Retrieved from http://dl.acm.org/citation.cfm?id=2891460.2891697.
[71] Guillaume Lample and Alexis Conneau. 2019. Cross-lingual language model pretraining. *CoRR* abs/1901.07291 (2019).
[72] Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2019. ALBERT: A lite BERT for self-supervised learning of language representations. arXiv:cs.CL/1909.11942.
[73] Ray R. Larson. 2010. Introduction to information retrieval. *J. Amer. Soc. Inf. Sci. Technol.* 61, 4 (Apr. 2010), 852–853. DOI: https://doi.org/10.1002/asi.v61:4
[74] Paula Lauren, Guangzhi Qu, Feng Zhang, and Amaury Lendasse. 2018. Discriminant document embeddings with an extreme learning machine for classifying clinical narratives. *Neurocomputing* 277 (2018), 129–138. DOI: https://doi.org/10.1016/j.neucom.2017.01.117
[75] Quoc V. Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. *CoRR* abs/1405.4053 (2014).
[76] Yann LeCun, Y. Bengio, and Geoffrey Hinton. 2015. Deep learning. *Nature* 521 (05 2015), 436–44. DOI: https://doi.org/10.1038/nature14539
[77] Yann Lecun, Leon Bottou, Y. Bengio, and Patrick Haffner. 1998. Gradient-based learning applied to document recognition. *Proc. IEEE* 86 (12 1998), 2278–2324. DOI: https://doi.org/10.1109/5.726791
[78] Ledell, Adam Fisch, Sumit Chopra, Keith Adams, Antoine Bordes, and Jason Weston. 2017. StarSpace: Embed all the things! arXiv:cs.CL/1709.03856.
[79] Jinhyuk Lee, Wonjin Yoon, Sungdong Kim, Donghyeon Kim, Sunkyu Kim, Chan Ho So, and Jaewoo Kang. 2019. BioBERT: A pre-trained biomedical language representation model for biomedical text mining. arXiv:cs.CL/1901.08746.
[80] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. 2019. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *arXiv preprint arXiv:1910.13461* (2019).
[81] Liang-Chih, Jin Wang, K. Robert Lai, and Xuejie Zhang. 2018. Refining word embeddings using intensity scores for sentiment analysis. *IEEE/ACM Trans. Audio, Speech Lang. Proc.* 26, 3 (Mar. 2018), 671–681. DOI: https://doi.org/10.1109/TASLP.2017.2788182
[82] Chenghua Lin and Yulan He. 2009. Joint sentiment/topic model for sentiment analysis. In *18th ACM Conference on Information and Knowledge Management (CIKM'09).* ACM, New York, NY, 375–384. DOI: https://doi.org/10.1145/1645953.1646003
[83] Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. 2015. Learning context-sensitive word embeddings with neural tensor skip-gram model. In *24th International Conference on Artificial Intelligence (IJCAI'15).* AAAI Press, 1284–1290. Retrieved from http://dl.acm.org/citation.cfm?id=2832415.2832428.
[84] Shuhua Liu and Thomas Forss. 2014. Combining N-gram based similarity analysis with sentiment analysis in web content classification. In *International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management - Volume 1 (IC3K'14).* SCITEPRESS - Science and Technology Publications, Lda, 530–537. DOI: https://doi.org/10.5220/0005170305300537
[85] Yuanchao Liu, Bingquan Liu, Lili Shan, and Xin Wang. 2018. Modelling context with neural networks for recommending idioms in essay writing. *Neurocomputing* 275 (2018), 2287–2293. DOI: https://doi.org/10.1016/j.neucom.2017.11.005
[86] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. RoBERTa: A robustly optimized BERT pretraining approach. *CoRR* abs/1907.11692 (2019).

[87] David M. Magerman. 1995. Statistical decision-tree models for parsing. In *33rd Meeting on Association for Computational Linguistics (ACL'95)*. Association for Computational Linguistics, 276–283. DOI: https://doi.org/10.3115/981658.981695

[88] Danilo P. Mandic and Jonathon Chambers. 2001. *Recurrent Neural Networks for Prediction: Learning Algorithms, Architectures and Stability*. John Wiley & Sons, Inc., New York, NY.

[89] Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Rose Finkel, Steven Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Meeting of the Association for Computational Linguistics (System Demonstrations)*. 55–60.

[90] Bryan McCann, James Bradbury, Caiming Xiong, and Richard Socher. 2017. Learned in translation: Contextualized word vectors. *CoRR* abs/1708.00107 (2017).

[91] Bryan McCann, James Bradbury, Caiming Xiong, and Richard Socher. 2017. Learned in translation: Contextualized word vectors. In *Conference on Advances in Neural Information Processing Systems*.

[92] Yelena Mejova and Padmini Srinivasan. 2011. Exploring feature definition and selection for sentiment classifiers.

[93] Oren Melamud, Jacob Goldberger, and Ido Dagan. 2016. context2vec: Learning generic context embedding with bidirectional LSTM. In *20th SIGNLL Conference on Computational Natural Language Learning*. Association for Computational Linguistics, 51–61. DOI: https://doi.org/10.18653/v1/K16-1006

[94] Oren Melamud, Jacob Goldberger, and Ido Dagan. 2016. context2vec: Learning generic context embedding with bidirectional LSTM. In *Conference on Computational Natural Language Learning*.

[95] Prem Melville, Wojciech Gryc, and Richard D. Lawrence. 2009. Sentiment analysis of blogs by combining lexical knowledge with text classification. In *15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'09)*. ACM, New York, NY, 1275–1284. DOI: https://doi.org/10.1145/1557019.1557156

[96] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S. Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Conference on Advances in Neural Information Processing Systems*. 3111–9.

[97] Saif Mohammad, Svetlana Kiritchenko, and Xiaodan Zhu. 2013. NRC-Canada: Building the state-of-the-art in sentiment analysis of tweets. In *Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 2: Proceedings of the Seventh International Workshop on Semantic Evaluation (SemEval 2013)*. Association for Computational Linguistics, 321–327. Retrieved from http://aclweb.org/anthology/S13-2053.

[98] James N. Morgan and John A. Sonquist. 1963. Problems in the analysis of survey data, and a proposal. *J. Amer. Statist. Assoc.* 58, 302 (1963), 415–434. Retrieved from http://www.jstor.org/stable/2283276.

[99] Nikola Mrksic, Ivan Vulic, Diarmuid Ó Séaghdha, Ira Leviant, Roi Reichart, Milica Gasic, Anna Korhonen, and Steve J. Young. 2017. Semantic specialisation of distributional word vector spaces using monolingual and cross-lingual constraints. *CoRR* abs/1706.00374 (2017).

[100] T. Mullen and R. Malouf. 2006. A preliminary investigation into sentiment analysis of informal political discourse. *AAAI Spring Symposium - Technical Report* SS-06-03 (2006), 159–162. Retrieved from https://www.scopus.com/inward/record.uri?eid=2-s2.0-33747172751&partnerID=40&md5=6b12793b70eae006102989ed6d398fcb.

[101] Martin Müller, Marcel Salathé, and Per E. Kummervold. 2020. COVID-Twitter-BERT: A natural language processing model to analyse COVID-19 content on Twitter. *arXiv preprint arXiv:2005.07503* (2020).

[102] Marwa Naili, Anja Habacha Chaibi, and Henda Hajjami Ben Ghezala. 2017. Comparative study of word embedding methods in topic segmentation. *Procedia Comput. Sci.* 112 (2017), 340–349.

[103] Vivek Narayanan, Ishan Arora, and Arjun Bhatia. 2013. Fast and accurate sentiment classification using an enhanced Naive Bayes model. *CoRR* abs/1305.6143 (2013).

[104] Usman Naseem. 2020. *Hybrid Words Representation for the Classification of Low Quality Text*. Ph.D. Dissertation. University of Technology Sydney, Australia.

[105] U. Naseem, S. K. Khan, M. Farasat, and F. Ali. 2019. Abusive language detection: A comprehensive review. *Indian J. Sci. Technol.* 12, 45 (2019), 1–13.

[106] U. Naseem, I. Razzak, and P. W. Eklund. 2020. A survey of pre-processing techniques to improve short-text quality: a case study on hate speech detection on twitter. *Multimedia Tools and Applications*, 1–28.

[107] Usman Naseem, Shah Khalid Khan, Imran Razzak, and Ibrahim A. Hameed. 2019. Hybrid words representation for airlines sentiment analysis. In *AI 2019: Advances in Artificial Intelligence*, Jixue Liu and James Bailey (Eds.). Springer International Publishing, Cham, 381–392.

[108] Usman Naseem, Matloob Khushi, Shah Khalid Khan, Nazar Waheed, Adnan Mir, Atika Qazi, Bandar Alshammari, and Simon K. Poon. 2020. Diabetic retinopathy detection using multi-layer neural networks and split attention with focal loss. In *International Conference on Neural Information Processing*. Springer, 1–12.

[109] Usman Naseem, Matloob Khushi, Vinay Reddy, Sakthivel Rajendran, Imran Razzak, and Jinman Kim. 2020. BioALBERT: A simple and effective pre-trained language model for biomedical named entity recognition. *arXiv preprint arXiv:2009.09223* (2020).

[110] Usman Naseem and Katarzyna Musial. 2019. DICE: Deep intelligent contextual embedding for Twitter sentiment analysis. In *International Conference on Document Analysis and Recognition (ICDAR'19)*. IEEE, 953–958.

[111] U. Naseem, I. Razzak, M. Khushi, P. W. Eklund, and J. Kim. 2021. COVIDSenti: A large-scale benchmark Twitter data set for COVID-19 sentiment analysis. *IEEE Transactions on Computational Social Systems.*

[112] Usman Naseem, Katarzyna Musial, Peter Eklund, and Mukesh Prasad. 2020. Biomedical named-entity recognition by hierarchically fusing BioBERT representations and deep contextual-level word-embedding. In *International Joint Conference on Neural Networks (IJCNN'20)*. IEEE, 1–8.

[113] Usman Naseem, Imran Razzak, Peter Eklund, and Katarzyna Musial. 2020. Towards improved deep contextual embedding for the identification of irony and sarcasm. In *International Joint Conference on Neural Networks (IJCNN'20)*. IEEE, 1–7.

[114] Usman Naseem, Imran Razzak, and Ibrahim A. Hameed. 2019. Deep context-aware embedding for abusive and hate speech detection on Twitter. *Aust. J. Intell. Inf. Process. Syst.* 15, 3 (2019), 69–76.

[115] Usman Naseem, Imran Razzak, Katarzyna Musial, and Muhammad Imran. 2020. Transformer based deep intelligent contextual embedding for Twitter sentiment analysis. *Fut. Gen. Comput. Syst.* 113 (2020), 58–69.

[116] Arvind Neelakantan, Jeevan Shankar, Alexandre Passos, and Andrew McCallum. 2015. Efficient non-parametric estimation of multiple embeddings per word in vector space. *CoRR* abs/1504.06654 (2015).

[117] Dat Quoc Nguyen, Thanh Vu, and Anh Tuan Nguyen. 2020. BERTweet: A pre-trained language model for English Tweets. *arXiv preprint arXiv:2005.10200* (2020).

[118] Thomas Niebler, Martin Becker, Christian Pölitz, and Andreas Hotho. 2017. Learning semantic relatedness from human feedback using metric learning. *CoRR* abs/1705.07425 (2017).

[119] Alexander Pak and Patrick Paroubek. 2010. Twitter as a corpus for sentiment analysis and opinion mining. In *International Conference on Language Resources and Evaluation.*

[120] Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. 2002. Thumbs up?: Sentiment classification using machine learning techniques. In *ACL-02 Conference on Empirical Methods in Natural Language Processing - Volume 10 (EMNLP'02)*. Association for Computational Linguistics, 79–86. DOI: https://doi.org/10.3115/1118693.1118704

[121] U. Naseem, M. Khushi, S. K. Khan, K. Shaukat, and M. A. Moni. 2021. A comparative analysis of active learning for biomedical text mining. *Applied System Innovation* 4, 1 (2021), 23.

[122] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. 2013. On the difficulty of training recurrent neural networks. In *Proceedings of the 30th International Conference on International Conference on Machine Learning - Volume 28 (ICML'13)*. JMLR.org, III–1310–III–1318. Retrieved from http://dl.acm.org/citation.cfm?id=3042817.3043083.

[123] Cristian Patriche, Pîrn?u Gabriel, Adrian Grozavu, and Bogdan Ro?ca. 2016. A comparative analysis of binary logistic regression and analytical hierarchy process for landslide susceptibility assessment in the Dobrov River Basin, Romania. *Pedosphere* 26 (06 2016), 335–350. DOI: https://doi.org/10.1016/S1002-0160(15)60047-9

[124] C. V. Patriche, R. Pirnau, A. Grozavu, and B. Rosca. 2016. A comparative analysis of binary logistic regression and analytical hierarchy process for landslide susceptibility assessment in the Dobrov River Basin, Romania. *Pedosphere* 26, 3 (2016), 335–350.

[125] Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*. Association for Computational Linguistics, 2227–2237. DOI: https://doi.org/10.18653/v1/N18-1202

[126] Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. *CoRR* abs/1802.05365 (2018).

[127] Yuval Pinter, Robert Guthrie, and Jacob Eisenstein. 2017. Mimicking word embeddings using subword RNNs. *CoRR* abs/1707.06961 (2017).

[128] Pengda Qin, Weiran Xu, and Jun Guo. 2016. An empirical convolutional neural network approach for semantic relation classification. *Neurocomputing* 190, C (May 2016), 1–9. DOI: https://doi.org/10.1016/j.neucom.2015.12.091

[129] Zhaowei Qu, Xiaomin Song, Shuqiang Zheng, Xiaoru Wang, Xiaohui Song, and Zuquan Li. 2018. Improved Bayes method based on TF-IDF feature and grade factor feature for Chinese information classification. In *IEEE International Conference on Big Data and Smart Computing (BigComp'18)*. 677–680.

[130] J. R. Quinlan. 1987. Simplifying decision trees. *Int. J. Man Mach. Stud.* 27, 3 (1987), 221–234. DOI: https://doi.org/10.1016/S0020-7373(87)80053-6

[131] J. R. Quinlan. 1986. Induction of decision trees. *Mach. Learn.* 1, 1 (Mar. 1986), 81–106. DOI: https://doi.org/10.1023/A:1022643204877

[132] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2018. Language models are unsupervised multitask learners. Retrieved from https://d4mucfpksywv.cloudfront.net/better-language-models/language-models.pdf.

[133] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2019. Exploring the limits of transfer learning with a unified text-to-text transformer. *arXiv preprint arXiv:1910.10683* (2019).

[134] Arshia Rehman, Saeeda Naz, Usman Naseem, Imran Razzak, and Ibrahim A. Hameed. 2019. Deep AutoEncoder-Decoder framework for semantic segmentation of brain tumor. *Aust. J. Intell. Inf. Process. Syst.* 15, 3 (2019), 53–60.

[135] Yafeng Ren, Yue Zhang, Meishan Zhang, and Donghong Ji. 2016. Context-sensitive Twitter sentiment classification using neural network. In *30th AAAI Conference on Artificial Intelligence (AAAI'16)*. AAAI Press, 215–221. Retrieved from http://dl.acm.org/citation.cfm?id=3015812.3015844.

[136] Jack Reuter, Jhonata Pereira-Martins, and Jugal Kalita. 2016. Segmenting Twitter hashtags. *Int. J. Nat. Lang. Comput.* 5 (08 2016), 23–36. DOI: https://doi.org/10.5121/ijnlc.2016.5402

[137] M. Jaggi, P. Mandal, S. Narang, U. Naseem, and M. Khushi. 2021. Text mining of stocktwits data for predicting stock prices. *Applied System Innovation* 4, 1 (2021), 13.

[138] Seyed Mahdi Rezaeinia, Ali Ghodsi, and Rouhollah Rahmani. 2017. Improving the accuracy of pre-trained word embeddings for sentiment analysis. *CoRR* abs/1711.08609 (2017).

[139] Hassan Saif, Marta Fernandez Andres, Yulan He, and Harith Alani. 2013. Evaluation datasets for Twitter sentiment analysis: A survey and a new dataset, the STS-Gold. In *International Workshop on Emotion and Sentiment in Social and Expressive Media: Approaches and Perspectives from AI.*

[140] Mohammad Arshi Saloot, Norisma Idris, Nor Liyana Mohd Shuib, Ram Gopal Raj, and AiTi Aw. 2015. Toward tweets normalization using maximum entropy. In *Workshop on Noisy User-generated Text, NUT@IJCNLP.*

[141] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. DistilBERT, a distilled version of BERT: Smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108* (2019).

[142] Dominik Scherer, Andreas Müller, and Sven Behnke. 2010. Evaluation of pooling operations in convolutional architectures for object recognition. In *Artificial Neural Networks – ICANN 2010*, Konstantinos Diamantaras, Wlodek Duch, and Lazaros S. Iliadis (Eds.). Springer Berlin, 92–101.

[143] Seungil, David Ding, Kevin Canini, Jan Pfeifer, and Maya Gupta. 2017. Deep lattice networks and partial monotonic functions. arXiv:stat.ML/1709.06680.

[144] Aliaksei Severyn and Alessandro Moschitti. 2015. Twitter sentiment analysis with deep convolutional neural networks. In *38th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'15).* ACM, New York, NY, 959–962. DOI: https://doi.org/10.1145/2766462.2767830

[145] Mohammad Shoeybi, Mostofa Patwary, Raul Puri, Patrick LeGresley, Jared Casper, and Bryan Catanzaro. 2019. Megatron-LM: Training multi-billion parameter language models using model parallelism. arXiv:cs.CL/1909.08053.

[146] Tajinder Singh and Madhu Kumari. 2016. Role of text pre-processing in Twitter sentiment analysis. https://www.sciencedirect.com/science/article/pii/S1877050916311607.

[147] R. Socher, A. Perelygin, J. Y. Wu, J. Chuang, C. D. Manning, A. Y. Ng, and C. Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Conference on Empirical Methods in Natural Language Processing*. 1631–1642.

[148] Saeid Soheily-Khah, Pierre-François Marteau, and Nicolas Béchet. 2017. Intrusion detection in network systems through hybrid supervised and unsupervised mining process- a detailed case study on the ISCX benchmark dataset -.

[149] Kaitao Song, Xu Tan, Tao Qin, Jianfeng Lu, and Tie-Yan Liu. 2019. Mass: Masked sequence to sequence pre-training for language generation. *arXiv preprint arXiv:1905.02450* (2019).

[150] Karen Sparck Jones. 1988. A statistical interpretation of term specificity and its application in retrieval. In *Document Retrieval Systems.* Taylor Graham Publishing, London, UK, 132–142. Retrieved from http://dl.acm.org/citation.cfm?id=106765.106782.

[151] Robyn Speer, Joshua Chin, and Catherine Havasi. 2016. ConceptNet 5.5: An open multilingual graph of general knowledge. *CoRR* abs/1612.03975 (2016).

[152] Yu Sun, Shuohuan Wang, Yukun Li, Shikun Feng, Xuyi Chen, Han Zhang, Xin Tian, Danxiang Zhu, Hao Tian, and Hua Wu. 2019. Ernie: Enhanced representation through knowledge integration. *arXiv preprint arXiv:1904.09223* (2019).

[153] Yu Sun, Shuohuan Wang, Yu-Kun Li, Shikun Feng, Hao Tian, Hua Wu, and Haifeng Wang. 2020. ERNIE 2.0: A continual pre-training framework for language understanding. In *AAAI Conference on Artificial Intelligence.* 8968–8975.

[154] Ilya Sutskever, James Martens, and Geoffrey Hinton. 2011. Generating text with recurrent neural networks. In *28th International Conference on International Conference on Machine Learning (ICML'11).* Omnipress, 1017–1024. Retrieved from http://dl.acm.org/citation.cfm?id=3104482.3104610.

[155] Jared Suttles and Nancy Ide. 2013. Distant supervision for emotion classification with discrete binary values. In *Proceedings of the 14th International Conference on Computational Linguistics and Intelligent Text Processing - Volume 2 (CICLing'13).* Springer-Verlag, Berlin, 121–136. DOI: https://doi.org/10.1007/978-3-642-37256-8_11

[156] Symeon Symeonidis, Dimitrios Effrosynidis, and Avi Arampatzis. 2018. A comparative evaluation of pre-processing techniques and their interactions for Twitter sentiment analysis. *Exp. Syst. Applic.* 110 (2018), 298–310. DOI: https://doi.org/10.1016/j.eswa.2018.06.022

[157] Duyu Tang, Bing Qin, Furu Wei, Li Dong, Ting Liu, and Ming Zhou. 2015. A joint segmentation and classification framework for sentence level sentiment classification. *IEEE/ACM Trans. Audio, Speech Lang. Process.* 23, 11 (2015), 1750–61.

[158] Duyu Tang, Furu Wei, Bing Qin, Nan Yang, Ting Liu, and Ming Zhou. 2016. Sentiment embeddings with applications to sentiment analysis. *IEEE Trans. Knowl. Data Eng.* 28, 2 (2016), 496–509.

[159] Duyu Tang, Furu Wei, Bing Qin, Nan Yang, Ting Liu, and Ming Zhou. 2016. Sentiment embeddings with applications to sentiment analysis. *IEEE Trans. Knowl. Data Eng.* 28, 2 (Feb. 2016), 496–509. DOI: https://doi.org/10.1109/TKDE.2015.2489653

[160] Duyu Tang, Furu Wei, Nan Yang, Ming Zhou, Ting Liu, and Bing Qin. 2014. Learning sentiment-specific word embedding for Twitter sentiment classification. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers).* Association for Computational Linguistics, Baltimore, Maryland, 1555–1565. DOI: https://doi.org/10.3115/v1/P14-1146

[161] Alper Kursat Uysal and Serkan Günal. 2014. The impact of preprocessing on text classification. *Inf. Process. Manag.* 50 (2014), 104–112.

[162] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *31st International Conference on Neural Information Processing Systems (NIPS'17).* Curran Associates Inc., 6000–6010. Retrieved from http://dl.acm.org/citation.cfm?id=3295222.3295349.

[163] Byron Wallace. 2017. A sensitivity analysis of (and practitioners' guide to) convolutional neural networks for sentence classification. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers).* Asian Federation of Natural Language Processing, 253–263.

[164] Wei Wang, Bin Bi, Ming Yan, Chen Wu, Zuyi Bao, Jiangnan Xia, Liwei Peng, and Luo Si. 2019. StructBERT: Incorporating language structures into pre-training for deep language understanding. arXiv:cs.CL/1908.04577.

[165] Wenbo Wang, Lu Chen, Krishnaprasad Thirunarayan, and Amit P. Sheth. 2012. Harnessing Twitter "big data" for automatic emotion identification. In *International Conference on Privacy, Security, Risk and Trust and International Confernece on Social Computing*. 587–592.

[166] Yequan Wang, Minlie Huang, Xiaoyan Zhu, and Li Zhao. 2016. Attention-based LSTM for aspect-level sentiment classification. In *Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 606–615. DOI: https://doi.org/10.18653/v1/D16-1058

[167] Yuyang Wang, Roni Khardon, and Pavlos Protopapas. 2012. Nonparametric Bayesian estimation of periodic light curves. *Astrophy. J.* 756, 1 (Aug. 2012), 67. DOI: https://doi.org/10.1088/0004-637x/756/1/67

[168] Ikuya Yamada, Hideaki Takeda, and Yoshiyasu Takefuji. 2015. Enhancing named entity recognition in Twitter messages using entity linking. In *Workshop on Noisy User-generated Text, NUT@IJCNLP*.

[169] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime G. Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. 2019. XLNet: Generalized autoregressive pretraining for language understanding. *CoRR* abs/1906.08237 (2019).

[170] Yukun Zhu, Ryan Kiros, Richard S. Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. *CoRR* abs/1506.06724 (2015).

Received May 2020; revised October 2020; accepted November 2020