final thesis.pdf

by Vasist G N

Submission date: 14-Feb-2021 02:03PM (UTC+0000)
Submission ID: 144777368
File name: final_thesis.pdf (3.48M)
Word count: 32790
Character count: 173400

COMPARATIVE STUDY OF DIFFERENT PREDICTIVE MODELS FOR THE TASK OF TEXT SIMILARITY

Vasist G N

Student ID: **931122**

Under the supervision of

SUVAJIT MUKHOPADHYAY

A thesis submitted in fulfilment of the

requirements for the award of the degree of

MASTER OF SCIENCE IN DATA SCIENCE

Liverpool John Moores University – Master's in Data Science

November 2020

ACKNOWLEDGMENT

I would like to thank my supervisor, Suvajit Mukhopadhyay, for his assistance and guidance provided for completion of this thesis.

I want to thank Prof. Manoj Jayabalan from LJMU for his weekly session's guidance provided by is valuable and informative.

I would like to thank the mentors from UpGrad and IIT-B who were helpful in making this journey pleasant.

I would also like to thank my family for all the support provided.

Abstract

Text similarity is an important consideration in designing systems like design of contextual chatbots, duplicate question detection, web search engines and machine translation. One major challenge in real world applications is tradeoff between computational cost and accuracy while designing a text similarity model. For instance, over 100 million visit Quora month and more than 14 million questions have been asked so far. There is a need to have a system in place which is both accurate and time efficient. This will help systems like Quora and Stack-overflow to identify duplicate questions with lesser computational resources. Overall end users will be benefitted as they would find answers to their questions rather than waiting for someone to answer.

Complex neural network models have proven their advantage over classical models however the training time required, and computational cost associated with them is high. Explosion of data in recent times have necessitated need of an optimal time efficient text similarity model. Transfer Learning has bought about a revolutionary change in natural language processing paradigm, but these models again incur huge computational cost and are less time efficient (Reimers and Gurevych, 2020).

We will evaluate various neural networks models with transfer learning which are simple with lesser trainable parameters also classical shallow learning models with different combination of text pre-processing, text processing and feature engineering along with ensemble approaches. We aim to achieve cost efficient text similarity model which can be used effectively for the task of text similarity.

Keywords: Quora, Stack-overflow, Neural Networks, transfer learning, shallow learning and feature engineering.

Content

List of Figures

List of Tables

List of Abbreviations

Abbreviation	Expansion
LSTM	Long short-term memory
NLP	Natural language processing
GRU	Gated Recurrent Units
ULMFit	Universal Language Model Fine-tuning for Text Classification
SGD	Stochastic gradient descent
NLTK	Natural Language Toolkit
POS	Parts of Speech
NER	Named Entity Recognition
RNN	Recurrent neural network
BERT	Bidirectional Encoder Representations from Transformers
TF-IDF	Term Frequency – Inverse Document Frequency
AUC	Area under curve
CNN	Convolutional Neural Network
AWD-LSTMs	Average SGD Weight Dropped LSTMs
Seq2Seq	Sequence to Sequence
GPT	Generative Pre-Training Model
ELMO	Embeddings from Language Models
ALBert	A Lite Bert
CBOW	Continuous Bag of Words
LEM	Lemmatization
STEM	Stemming
LR	Logistic Regression
LIGHTGBM	Light Gradient Boosting Machine
CatBoost	Category Boosting
NB	Naïve Bayes

1. Introduction

1.1 Background

Finding similarity between documents or texts is a need for both online and offline systems. With the internet and data boom more recently, tasks related to text similarity is facing challenge of a different proportion. Language model which caters the need of text similarity must be both accurate and fast. Handheld devices like mobile phones and tablets although evolving still require applications to be built which are computationally less intensive. Exponential growth of data also necessitates need for optimal text similarity models.

Text similarity has a wide range of applications. Web portals like Quora and StackOverFlow receive numerous questions every day. It is required to have system which could detect if the question asked by the user already has been asked previously. If a language model can detect whether a question asked is duplicate or not it would not only help user to find relevant answer quickly, but this also helps service provider in many ways. There is no need to open a thread for the question asked so no extra storage is involved where even answers maybe duplicated. Making text similarity model to run in real time requires the model to return results fast, this calls for a model which is simple yet powerful.

There are numerous other use cases of text similarity like finding similar documents, web search engines, plagiarism checking applications and many more. All these use cases domain maybe different but the problem at the core remains the same which is data that is growing exponentially.

Initial days of automation text similarity was mostly detected using string-based approaches. They were either character-based or word based. These were extremely basic methods which would compare texts character wise or word wise. They neither were able to capture semantics or syntactic information of the given text. Minor changes in document would be enough to escape the duplication check.

Corpus based approaches wherein one could represent textual data in vector space bought about a change in way text similarity models were built. Various corpus-based approaches came to fore like count vectorizer, TFIDF, bag of words and others. Using these approaches a word embedding comprising of these unit vectors were built which was eventually fed to machine learning

algorithms. These algorithms were trained on these word embeddings and were able to make predictions based on the training. Word embeddings built using these approaches were not able to extract much information about the text and eventually suffered in determining similarity between the texts.

There was a need of an approach which would have the intelligence to learn from the textual data which was basically unstructured. Deep learning approaches based on the neural model bought a wave of change wherein word embedding was part of the network. weights and biases of the embedding layer were also adjusted iteratively while training. This changed the way in which text similarity or in a broader way natural language tasks were modelled. Language model was able to iteratively learn nuances of the underlying data. These neural models heralded an era of intelligent language models which were able to understand semantics and grammar of the natural language in process.

Deep learning approaches helped in bringing pre-trained word embeddings like Glove, Word2Vec and FastText which resulted highly accurate models with little training as they were able to transfer knowledge substantially. These Pre-Trained word embeddings are used with various machine learning and deep learning frameworks to build intelligent language models which can identify semantics and grammar of the underlying language.

Further we saw improved architectures like Bi-Directional RNN's, Seq-Seq and Encoder-decoder mechanisms which enhanced the performance of text similarity models. RNNs traditionally have an issue while modeling long-range dependencies, the problem of vanishing gradients. This issue of long range dependencies and vanishing gradients was addressed with invent of LSTM's (Hochreiter and Schmidhuber, 1997). LSTM's aided with mechanism of gates and cell states was able to retain long-range dependencies. GRU's (Gated Recurrent Units) (Chung et al., 2014) a lighter version of LSTM's which performs on par with LSTM's also started gaining popularity.

Encoding long sequences into vectors still was a problem with long range sequences. Attention mechanism (Chorowski et al., 2015) wherein special attention to input tokens overcame these issues and started to give better performance later (Shaw et al., 2018) self-attention Transformer Models which works work upon Self Attention improved performance further. Self-Attention is a new type of mechanism where a given token looks around its neighboring tokens to form embedding or its meaning. The Attention mechanism with Transformers have been a revolution in

the field of Natural Language modeling. modelling (Howard and Ruder, 2018) Transfer learning and language began the new era of transfer learning in natural language processing tasks, ULMFit built using state-of-art AWD-LSTMs gained prominence in various language models.

(Peters et al., 2018) introduced bidirectionality into natural language modeling tasks which bought ELMo to fore in natural language tasks. ELMO is a feature-based approach where to form word embeddings the context of the sentence is involved. It is done by making the model look from either direction (forward, backward) and concatenating the results. Bi-directional LSTMs is used to capture context from either side. This has allowed one to either use the word embeddings for language modeling or combine with other available embeddings.

Transformer-based approaches have revolutionized the text similarity models, overall natural language tasks. These pre-trained models are trained with a large corpus, which makes these models highly efficient. With very less pre-processing and feature engineering text similarity models with high accuracy can be built. Distillation process has made large pre-trained word embeddings not only smaller but as accurate as the bigger models.

(Devlin et al., 2019) BERT has proven to be an efficient state of art model for natural language processing tasks it has outperformed various state of art. For text similarity models like sentence or document similarity, it requires that both sentences and documents to be fed into the network, which causes a massive computational overhead. There is a necessity to choose language models which can process a document in a simple yet in a powerful manner.

We are also seeing various lighter word embedding which are equally provided state-of-art results. (Reimers and Gurevych, 2020) came up with Sentence-BERT (SBERT) which was able to produce same accuracy as BERT but was able to find the text similarity in about 5 seconds. (Sanh et al., 2019) came up with DistilBERT which is 40% of the size of BERT yet has proven to be as powerful as BERT we could also see improved time efficiency as it lighter than BERT. (Jiao et al., 2019) is as effective as BERT while being 7.5x smaller and 9.4x faster.

CatBoost and LightGBM are not only accurate as XGBoost or Random Forest but also computationally efficient. Time taken to converge also is very less.

1.2 Problem Statement

Although we have seen a lot research being done in determining text similarity there has not been a comprehensive study of these approaches in terms of both accuracy and computational complexity. Text similarity can be measured using fuzzy approaches, string-based approaches, traditional machine learning approaches, neural networks with different models like ANN, RNN, CNN, LSTM, etc. also we have various transfer learning models with attention and transformer mechanisms.

Some of the previously conducted research on detecting text similarity are:

- (Dhakal et al., 2018) have carried out a survey using traditional machine learning approaches and an ANN architecture using on the Quora dataset. Random forest outperformed other machine learning models however ANN outperformed Random forest with an accuracy of 86.09%.
- (Benard Magara et al., 2018) evaluated various machine learning text similarity approaches both for accuracy and time efficiency. RPart algorithm was able to provide accuracy and time efficiency of 80.73 and 2.354628 seconds respectively on a small dataset however random forest and XGBoost took close to 40 seconds.
- (Sharma et al., 2019) conducts extensive exploration of Quora dataset and used various machine learning models, including linear, tree-based model and neural network models. CBOW Neural network model outperformed compared to other models.
- (Reimers and Gurevych, 2020) Sentence BERT is a novel approach which fine-tunes BERT. SBERT is computationally more efficient than BERT and more accurate than BERT. SBERT architecture makes it an apt for usage in sentence similarity systems which has less computational power and still will be able to give better performance.
- (Wang et al., 2020) this paper shows how pre-trained word embedding word2vec is used with different neural architectures to detect duplicate questions. word2vec with LSTM outperformed compared to other models.
- (Imtiaz et al., 2020) three different word embeddings were used with Siamese LSTM. Ensemble method of combining the outputs of three embeddings was done 91% accuracy

was achieved by this ensemble approach. Ensemble approach have proven in various machine and deep learning systems to be delivering better performance compared to individual models.

- (Liu et al., 2017) used ensemble technique wherein a set of learning algorithms will be used rather than constituent learning algorithm alone. Individual language model tends to cause bias in terms of fixed set of parameters, Ensemble approach helps reducing such a bias.

Most of these studies mentioned above try to achieve better accuracy. But real-world applications need to be both time efficient and accurate. In this exploratory research we will evaluate both time efficiency and accuracy. We will explore various text pre-processing steps like stemming, lemming, removal of stop words with and without text processing steps like POS tagging, NER etc. Then we will explore feature engineering approaches like TF-IDF, Count Vectorizer and continuous Bag of words with various machine and simple neural network models.

We shall also explore transfer learning with lighter word embeddings like Distil BERT, Tiny Bert with various machine and simple neural network models. There hasn't been research conducted on these lighter word embeddings for the task text similarity, evaluation of both accuracy and time efficiency will determined on the word embeddings. Finally, we will try out ensemble methods by combining output of one or models.

1.3 Research Question

The central research questions part of this study is listed below:

- Which word embedding gives better performance for the task of text similarity?
- Will lighter models like DistilBert and TinyBert be able to match performance of BERT and be time efficient?
- Will usage of ensemble approach result in better performance?
- Can we have text similarity models with higher accuracy which can be used in real time applications?

- Are traditional machine learning methods with proper usage of feature engineering methodologies be able to match the performance of neural networks or transformer models?
- Usage of NER or POS how effective are they in terms of performance against not using it?
- How different class balancing methods influence on overall stability of the model?

1.4 Aim and Objectives

The main aim of this research to evaluate various text similarity models which can be used in real world use cases. The goal here is to find a language model which can detect similarity between texts accurately and in a time efficient manner.

The research objectives are formulated based for this study is as follows:

- To explore various feature engineering techniques and available pre-trained embeddings.
- To investigate various language models which can determine text similarity.
- To analyze the different language models considered in this study in terms of accuracy and time efficiency.
- To investigate whether ensemble approaches will be able to improve overall performance as compared to individual language models significantly.

1.5 Significance of the Study

- Explosion of data in recent times have necessitated need of an optimal time efficient text similarity model, (Patrizio, 2019) predicts 175 Zettabytes of data by 2025. This huge data necessitates the need of more computational resources to accomplish text/document similarity tasks. Training time required to build these models also will be significantly high. Simple models with less trainable parameters which can give good performance can make us well prepared to handle data explosion better.
- Language models in the era of big data should be simple yet powerful, it requires to be both accurate and take less to train and predict. Real time applications need the result to be returned in couples of seconds/minutes. Organizations which offer text similarity solutions

require more computational resources to provide seamless service to users as more computational resources are required this in turn results more cost to avail these services. If there is a model which can perform the task with less resources and still be accurate, we can see more real time applications in this domain and overall experience of using web search/ plagiarism check applications would be better.

- This comparative study becomes significant to address the need of the optimal language models.

1.6 Scope of the Study

The scope of this comparative study will be limited to the below items:

- Exploring data pre-processing and feature engineering techniques for the task of text similarity.
- Building language model with simple architecture using both transfer learning and traditional corpus-based methods.
- Models will be tuned for better performance by evaluating with different tunable parameters.
- Evaluating ensemble approach with various combination of models. Each ensemble model will have same feature engineering methodology but will have different combination of learning models.

The following will not be included as a part of the study:

- More than one embedding technique in an ensemble.
- Only Quora duplicate questions data set will be considered for this study no other dataset like this will be evaluated.
- No new method or model will be proposed from this study.

1.7 Structure of Study

This study is divided into 3 parts.

Chapter 1 – Introduction gives an overview to topic considered for this study an efficient text similarity model and emphasizes on the need of computationally efficient and accurate text similarity model. It gives an overview of the problem on hand and provides a brief about existing approaches. Aims, objectives and scope of research outlined in this section which gives birds eye view of the study considered in this paper. Scope of the research particularly describes what this research intends to achieve.

Chapter 2 – Literature survey this chapter initially deals with understanding what is natural language processing, what is text similarity and what is the need of simple text similarity language model. Then it provides an overview of the historical study done in the field of text similarity. Then it emphasizes on natural language process evolution to support the task of text similarity. Brief overview is also provided on how machine learning techniques and deep learning techniques advancements aided NLP in task of text similarity. Finally, evolution of transfer learning is seen in detail.

Chapter 3- Research Methodology this chapter focuses on the methodology to be followed during this study. In this chapter we will start understanding the data by performing exploratory data analysis for understating the data. Post understanding the dataset the next this chapter deals with data pre-processing, in this step we will focus on cleansing the input data by removing stop-words, unwanted characters, lowercasing the texts to avoid duplicates and finally reducing the words in the text to its inflectional form using either stemming or lemming. Next this chapter focuses on building word embeddings using various embedding techniques which create from scratch also pre-trained embeddings like Glove or Word2vec. Once these embeddings are created next focus would be one building language models using various machine and deep learning frameworks like Logistic regression, Naïve Bayes, Cat-Boost, Light-GBM and simple GRU with ensemble mechanism. The study also explores transformer-based approaches like Tiny-BERT and Distil-BERT. Ensemble technique of combining multiple model outputs will also be evaluated. Aim of this comparative study is to accurate computationally efficient language model for the task of text similarity.

2. Literature Review

2.1 Introduction

This section we will see how text similarity approaches have evolved. (Measuring Sentences Similarity: A Survey) Text similarity in terms of natural language processing has been in place from long time, majorly Word based, structure based, and vector-based techniques are predominantly used. Also, we have seen evolution of models right from Bayesian approach to deep models, Deep learning and transfer learning has revolutionized the field of text similarity. Here we will analyze currently available text similarity techniques with one another. This study will mainly focus on techniques which take less computational resource and provide accurate results. In this section we will discuss machine learning techniques which includes both shallow and deep learning techniques, also transformer models which has become a new normal in this field.

2.1.1 What is text similarity and why is it required?

Text similarity is process in which two texts are compared with one another to measure their similarity. For Example in the dataset we have considered for this study Quora question answer dataset suppose user asks a question "What are the most popular text similarity algorithms?" and if there are answers already available for question "Which is the best algorithm for checking string similarity metric?" user should be returned the answer answered for a similar type of question. Text similarity has a broad variety of use cases across all domains. Web searches, plagiarism checks, contextual chatbots, document similarity use text similarity techniques.

2.1.2 Why do we require a text similarity model which is both fast and accurate?

Data is growing exponentially over the years (Patrizio, 2019) predicts 175 Zettabytes of data by 2025. This necessitates more training to be performed by the underlying text similarity model, which in turn requires more computational resources. This increases cost for the provider providing text similarity. These complex models also will be big in size would again require more computational resources to host them. Having a simple model not reduces training time but also be deployed and consumed with lesser computational resources. The model being both accurate and fast can be used in wide variety of applications across all domains.

(Benard Magara et al., 2018) evaluated approaches which could optimize the task of text similarity. They investigated three machine learning algorithms Recursive Partitioning (RPart), Random Forest and Boosting algorithm. RPart was the best algorithm which not only gave accurate results but also was time efficient.

2.1.3 What is Natural Language processing?

Natural language processing is an artificial intelligence methodology, which gives the computer the ability to interpret and understand human language. Natural language processing is used in wide variety of applications in our day to day life like AI assistants like Alexa, Siri, google assistant or in cars which take speech as input. Machine or the processor accepting inputs does not understand the human language in the native form it requires a machine understandable input and again it must give response in a human understandable form. Here is where natural language techniques are used which will act as an interpreter between the machine and humans.

Natural language processing techniques can broadly be categorized as

- Syntactic Analysis**:** Syntactic Analysis mainly focuses on syntactic alignment of words in each sentence. Syntactic analyzing checks whether given sentence is following grammatical rules.
- Semantic Analysis: Semantic analysis looks for the meaning conveyed by the sentence. It more focuses the intent of the sentence.

2.2 Related Works on text similarity

In this section first will get started with SemVal (**Sem**antic **Eval**uation) an ongoing series to evaluate computational semantic analysis, here text similarity challenges have been a topic since 2012 until 2017.

(Aggarwal et al., 2012) came with up novel approach of combining corpus-based and knowledge-based semantics to determine text similarity. The combined scores obtained were fed to a linear regressor and a bagging model, this experiment also showed how combining learnings from different language models resulted in better performance compared to the base models. This study still needs to be extended to understand syntactic information.

2017 SemVal challenge used benchmark Pearson's r * 100 dataset below table has details of top performing architectures.(Tian et al., 2018) ECNU average scores of four deep learning models and three well performing feature engineered models. Three feature engineered models use Random Forest (RF), Gradient Boosting (GB) and XGBoost (XGB). (Maharjan et al., 2020) another top performing language model which came second in English language task, it used various feature engineering techniques like unigram overlap, summed word alignments scores, fraction of unaligned words, difference in word counts by type (all, adj, adverbs, nouns, verbs), and min to max ratios of words by type. Different deep learning models were built on it researchers used a unique ensemble technique Genre-Attention which was able to learn from different base models. Overall resulting ensemble was more powerful in terms of accuracy. Both these SemVal models emphasizes the power of combining different model outputs to make it a power ensemble. This benefit of having ensemble approach further necessitates need of having ensemble approaches in the future research to be done.

STS 2017 Participants on STS Benchmark			
Name	Description	Dev	Test
ECNU	Ensemble (Tian et al., 2017)	84.7	81.0
BIT	WordNet+Embeddings (Wu et al., 2017)	82.9	80.9
DT_TEAM	Ensemble (Maharjan et al., 2017)	83.0	79.2
HCTI	CNN (Shao, 2017)	83.4	78.4
SEF@UHH	Doc2Vec (Duma and Menzel, 2017)	61.6	59.2
Sentence Level Baselines			
sent2vec	Sentence spanning CBOW with words & bigrams (Pagliardini et al., 2017)	78.7	75.5
SIF	Word embedding weighting & principle component removal (Arora et al., 2017)	80.1	72.0
InferSent	Sentence embedding from bi-directional LSTM trained on SNLI (Conneau et al., 2017)	80.1	75.8
C-PHRASE	Prediction of syntactic constituent context words (Pham et al., 2015)	74.3	63.9
PV-DBOW	Paragraph vectors, Doc2Vec DBOW (Le and Mikolov, 2014; Lau and Baldwin, 2016)	72.2	64.9
Averaged Word Embedding Baselines			
LexVec	Weighted matrix factorization of PPMI (Salle et al., 2016a,b)	68.9	55.8
FastText	Skip-gram with sub-word character n-grams (Joulin et al., 2016)	65.3	53.6
Paragram	Paraphrase Database (PPDB) fit word embeddings (Wieting et al., 2015)	63.0	50.1
GloVe	Word co-occurrence count fit embeddings (Pennington et al., 2014)	52.4	40.6
Word2vec	Skip-gram prediction of words in a context window (Mikolov et al., 2013a,b)	70.0	56.5

Table 2-1 STS Benchmark for text similarity models 2017

(Dhakal et al., 2018) have carried out a survey using traditional machine learning approaches and an ANN architecture using on the Quora dataset. Data was initially cleansed by removing stop words later words were tokenized and stemmed. Researchers used Word2Vec pre-trained

embedding and experimented on various machine learning algorithms like Logistic Regression, Decision Tree, Naïve Bayes, KNN and Random Forest. Random forest outperformed other machine learning models. The study also used ANN model with a simple architecture which outperformed all machine learning approaches. As this study was limited to handful of approaches none of the algorithms used in this study supports long term dependencies also there is a need to explore more language modelling techniques.

(Benard Magara et al., 2018) evaluated various non-linear machine learning algorithms for the task of text similarity. Main aim of this study was to determine text similarity models which were efficient and accurate. Researchers used TF-IDF for feature engineering and three non-linear modelling algorithms Random Forest, Recursive Partitioning (RPart) and Boosted Tree were used. Boosted Tree was the most efficient model, RPart was as accurate as Random Forest but in terms of time efficiency RPart was 20 times faster than other models. As concluded by researchers this study did not consider any advanced models and was not evaluated on a larger data set. Sample considered for this study is small, performance metrics mentioned may vary if a large or an alien data is validated against this model. However, this study gave a direction for text similarity tasks for using partitioning methodologies while training

(Sharma et al., 2019) conducted a comparative study of various language modelling approaches. Feature engineering techniques like Unigrams, Bigrams, Trigrams and continuous bag of words were used. For building language models linear models like SVM and Logistic Regression, Tree models like Decision Tree, Random Forest and Gradient Boosting finally deep learning frameworks like LSTM, Bi-LSTM and attention framework were considered. Continuous bag of words with a neural model outperformed all other models. This comparative study displayed how neural networks were apt in language modelling tasks. Usage of architectures like LSTM and Bi-LSTM would have addressed the problem with long term dependencies in a big way. As researchers have suggested this study has to further explore ensemble approaches. Also, newer advancements in the language modelling like attention models or transformers must be explored with various feature engineering techniques.

(Wang et al., 2017) a bilateral multi-perspective matching (BiMPM) novel technique for matching texts, sentences are matched in both directions on Quora duplicate questions data set.

Suppose we have sentence A and B, matched will be done both A against B also with B against A. In this study three different architectures are being used apart from the proposed BiMPM model. First architecture uses a Siamese network were two Siamese models Siamese-CNN and Siamese-LSTM is used, both these models encode both the sentence vectors and decision is made taking the cosine similarity between them. Second uses a two baseline models Multi-Perspective CNN and Multi-Perspective LSTM rather than cosine similarity model used here multiple perspective cosine matching function is used. Third model used is matching aggregation framework proposed by (Wang et al., 2016). BiMPM model outperformed all the benchmark models considered in this study. In the same study BiMPM model was evaluated on SNLI dataset wherein ensemble-BiMPM model outperformed all other benchmark language models. This study showed how training sentences in both directions improves the stability of the model but for long term dependencies this solution may not be suited. Also, models considered are sequential in nature which trained from scratch considering the complexity in the model it will computationally inefficient. This approach of training sentences in both the directions will have it be researched upon further.

(Addair, 2017) compared three language modelling approaches to measure text similarity for Quora duplicate questions dataset. Siamese network framework was used with CNN and LSTM architectures. Researchers also combined both models CNN+LSTM results to make it combined language model. LSTM outperformed other language models in this study with a high F1 score. Major take way from this study is LSTMs are enormously powerful in modelling tasks, drawback of LSTM is it is very computationally intensive so including GRU in research a lighter version compared to LSTM would yield better results. Researcher feels using a word embedding built from Quora dataset would have yielded better results, usage of transformer based transfer learning which are trained on large corpus or training the word embedding from the dataset considered is somewhat this research aims to explore.

(Liu et al., 2019) Novel technique Concept Interaction Graph was introduced to transform a document to weighted undirected graph. Deep neural models suffered from long term dependencies so address that researchers used a graph representation of document which was later fed to a Siamese encoded Graph Convolutional network, this approach outperformed existing state-of-art approaches.

(Reimers and Gurevych, 2020) Sentence Bert modification of BERT, this architecture adds a pooling layer at the output of BERT. Pooling strategy was experimented with both Mean and Max strategies. Sentence BERT architecture used Siamese triple network to fine tune BERT, this not only boosted performance. Computational efficiency was significant, for the same task BERT took close to 65 hours SBERT reduced it to 5 seconds.

(Yao et al., 2019) proposed a new approach where unlabeled text can be sent directly to the network based on a LSTM autoencoder. Word vector is normalized so that algorithm converges faster, it has also helped in lowering chances of vanishing gradient. Word vectors trained from Quora dataset is better compared to all other approaches. Word vectors from Wikipedia gave a good enough result. What this study indicates is accuracy of a model is dependent on how model learns the underlying dataset. As suggested by researcher wanting to explore attention would also help in parallel execution.

(Shahmirzadi et al., 2019) An extensive comparative analysis for text similarity was conducted. Researchers compared TF-IDF approach with LSI topic model and D2V neural model. TF-IDF without any hyperparameter tuning outperformed other two models, However, with hyper parameter tuning researchers were able to improve performance of LSI and DSV models in comparison to TF-IDF. It also must be noted that TF-IDF is quite simple and takes less computational time on the other hand both LSI and D2v are not only requires more computational power also requires hyperparameter tuning. Considering the performance benefit also there is not a significant change in terms of accuracy TF-IDF can be explored more.

(Patro et al., 2018) Novel architecture comprising of 3 layers, 1 LSTM encoder layer 2nd LSTM decoder and a 3rd LSTM discriminator. Training happens end to end across three modules. Weights are shared between 3 layers embedding learns both from global and local loss. Researchers propose 4 distinct models Local loss, global Loss, Local-Global Loss and Local-Global Loss shared. For Quora duplicate question dataset Local-Global Loss shared model outperformed. This sentence embedding using sequential encoder-decoder with a discriminator bettered previous state-of-art models for various benchmark datasets in terms of BLUE and TER scores.

(Imtiaz et al., 2020) Siamese-MaLSTM network architecture was used with three different word embeddings GoogleNewsVector, FastText and FastText sub word was used. Researchers also blended pre-trained embeddings and the same Siamese-MaLSTM was used. Blended embedding performed the best with F1-Score of 95 percent. As mentioned by researchers techniques like attention mechanism had to explored.

(He et al., 2015) Convolutional neural network-based similarity model which has primarily two components sentence model and similarity measurement model. In the sentence model convolutional network is used to capture different granularities of input. Similarity measurement layer uses multiple similarity measurement techniques to compare given sentences. Researchers evaluated on various benchmark datasets like SICK and MSRVID and were able to achieve state-of-art results.

(Peng et al., 2019) Bio-Medical Language Understanding evaluation was introduced to facilitate research language modelling research in the field of biomedicine domain. Researchers evaluated transformer approaches with fine tuning ELMO and BERT using PubMed and MIMIC-III dataset. Four Bert models were used in the research Bert base (PubMed), Bert base (PubMed + MIMIC-III), Bert large (PubMed) and Bert large (PubMed + MIMIC-III). Bert base (PubMed + MIMIC-III) outperformed other models when evaluated on benchmark MedSTS and BIOSSES datasets.

(Wang et al., 2020) have used Word2Vec embedding to build word embedding on StackOverFlow duplicate question dataset. Researcher has used CNN architecture for language modelling. CNN architecture has outperformed machine learning algorithms like Logistic Regression, SVM, Random Forest and XGBoost. Also, CNN has better accuracy than the RNN architecture considered.

This research will focus on building a stable language model, survey of the related work has showed how natural language along with machine/deep learning techniques have evolved to address the challenges faced. Every study we saw above brings a new dimension to further research it can either be addressing the drawbacks of the study or using the method used to further refine the model. Natural language modelling tasks have evolved with advancements in deep learning methodologies. Deep learning approaches bought pre-trained embeddings like Glove, Word2vec and other pre-trained embeddings to fore, which coupled with machine and

deep learning models are used for building various task specific language models. Deep learning architectures like encoder-decoder model, Seq2Seq architecture and Siamese network have been used widely in natural language tasks. Attention mechanism which helped language tasks mostly sequential to be executed in parallel. Further the self-attention mechanism led to various state-of-art transformer architecture models which changed the course of language modelling. Further in this chapter we will see how language models evolved for the task of text similarity.

2.3 Language Modelling Evolution

Field of text similarity is a branch on natural language processing. Over the years text similarity approaches have evolved with advances in natural language processing. Natural language models can be divided into two operations at a higher level first building a word embedding and second would be building a similarity model. (Farouk, 2019) has shown how text similarity has evolved with natural language processing. Broadly text similarity approaches into String Based, Corpus Based and Knowledge based.

String Based approaches which can further be categorized character based and term based was prevalent before word embeddings came to fore. String based approaches compared given text lexically however to build intelligent systems semantic understanding is essential here is where word embeddings with corpus based and knowledge-based approaches came to fore.

2.3.1 Word Embeddings

Word embeddings is a feature learning and a language modelling technique wherein words are mapped to vectors of real numbers. There are various available methods to build word embeddings broadly we can divide into word embeddings built from pre-trained word vectors and ones built by without using pre-trained vectors.

(A Neural Probabilistic Language Model) distribution representation of words refers to a vector representation which is nothing but a word embedding was used to train a neural model, This approach resulted in a model which performed better than the state of art. Distributed representation of words was able to overcome the curse of dimensionality which eventually

contributed to improvements in the model.(A unified architecture for natural language processing) showed the utility of using pre-trained word embeddings with neural networks.

Corpus based approaches can be categorized into bag-of-words model, Shallow window-based methods and Matrix factorization methods.

Bag-of-Words Model

In this order of the words is not considered, it will not capture the semantics of the given text.

One hot encoding: is a basic type of word embedding available wherein length of vector depends on the length of the input string. All vectors are orthogonal to one another measure of similarity cannot be done between two strings.

Bag-of-words: which has a fixed vector size for all the inputs in a training set, it does not take lexical information into consideration. In both bag of words and one hot encoding there is no way to discriminate between words one with more significance and with lesser significance.

Count vectorizer: is a type of word embedding which provides a simple way to build a word vector embedding of token counts. Word vector prioritizes number of appearance of words and discards words which appear in lesser frequency.

TF-IDF: model frequently occurring words are penalized over lesser frequently appearing words, words with lesser frequency is prioritized.

Shallow window-based methods

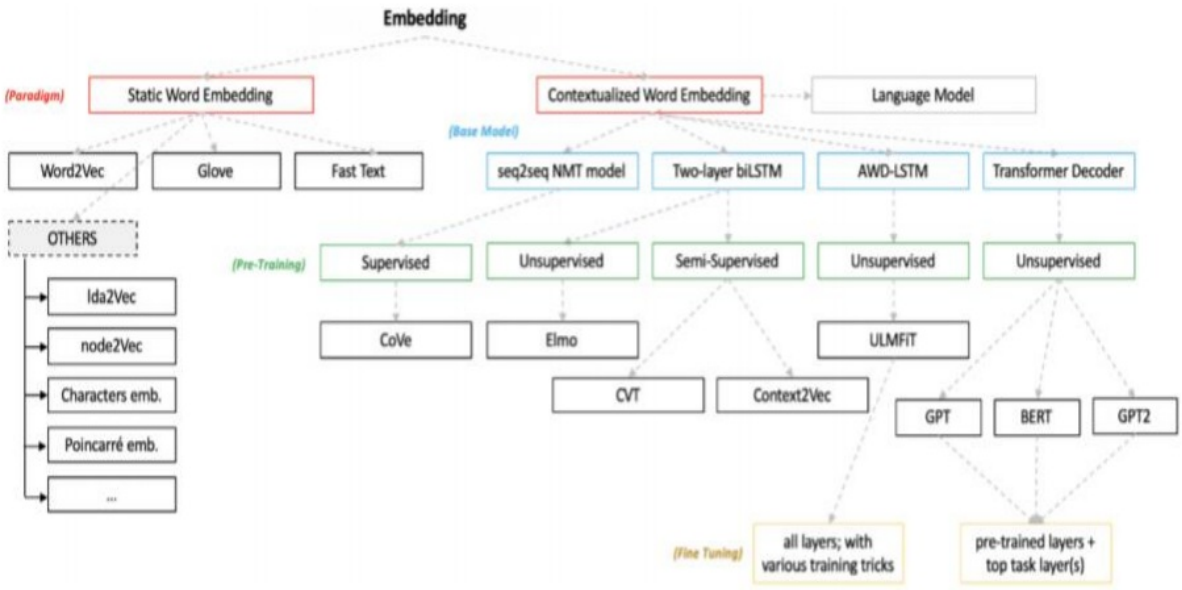


Figure 2-1 Pre-Trained Word Embeddings

Word2vec

(Efficient Estimation of Word Representations in Vector Space) introduced Word2vec which was a combination of two model's continuous bag of words and Skip-Gram. Word2Vec was trained with unsupervised approach wherein the words appearing in similar contexts have similar vector representations.

Architecture of Word2Vec was a simple neural network comprising of input and output layers with only one hidden layer which had 300 dimensions. Word2vec transformed natural language tasks as it could learn semantically and with grammar information based on the context. Major advantage of word embeddings built from word2Vec was it resource efficient did not require training from scratch. Word2vec suffered in tasks which had polysemy words and was difficult to optimize specifically.

Glove

(Pennington et al., 2014), (George et al., 2018) came up with Glove word embedding which is a combination of global matrix factorization and local context windows. Glove word embedding generate smaller dimension of a word embedding which helps save computational resources. Glove embeddings are optimized in such a way that dot products of input vectors are logarithm of word-occurrence. Glove embeddings gives importance to global corpus while learning the semantics of an input text. Glove embedding also does not work when polysemy words are present in data to be learned from.

Poincare embeddings

(Nickel and Kiela, 2017), started using hyperbolic space rather than the traditional distance mechanisms to determine the similarity between words. Hierarchical properties of words were captured using this technique however for non-hierarchical data this method may not be suitable.

FastText

(Joulin et al., 2017) introduced FastText which is built using character level representation. FastText can provide better results as it is at character level this makes more suitable for handling rare words and differentiating words with similar characteristics efficiently. However, it can be computationally inefficient.

All the embeddings seen so far are not contextual that is for example “Shopkeepers near the river bank have their accounts in this bank”, here word bank occurs twice but the meaning of the word bank is different in both the places. Word embeddings should be intelligent enough to understand the context of the words in each sentence.

Transfer learning models like BERT, ELMO, GPT and others fall in this category, this will be discussed in detail in further sections of this chapter.

Matrix Factorization Methods

These methods capture statistical information of a corpus by decomposing large matrices. LSA (Latent Semantic Analysis) and LDA (Latent Dirichlet allocation) are the most common methods.

For building text similarity language model, a machine learning modelling technique is required. As we have seen in various research conducted previously mainly machine learning algorithm Tree models and deep learning approaches have been used. Briefly will evaluate each of the modelling techniques used.

2.3.2 Machine Learning Models

Machine learning techniques bought about a change in way text similarity models were built. Models were trained with a certain corpus and was able to predict based on the learnt data. This heralded a new era in natural language processing.

Naïve Bayes is a simple but were very efficient algorithm for modelling language models. For smaller datasets naïve bayes performed extremely being robust, fast and easy to implement naïve bayes started getting used in many NLP tasks. Strong violation of independence of assumptions made it perform badly and on large datasets naïve bayes proved to be less efficient.

Logistic regression is one more simple to implement machine learning. Also there has been a wide usage of Logistic Regression in language modelling tasks as it is very simple to model.

Support vector machines will create a hyperplane in N-dimensional space for each N-class. SVM however suffers with performance as the time complexity involved in that is more and will also require more computational resources.

Unlike Tree models or deep learning models, machine learning algorithms do not learn much from the underlying data. It most relies on feature engineering and independent input features, NLP training data will mostly be unstructured and requires the model to mostly learn from the underlying data. Tree models or deep learning techniques would be mor appropriate.

2.3.3 Tree Models

Tree models are extensively used in NLP tasks and are known to give good accurate results which are also interpretable. Major advantage with tree models is interpretable and users can visualize the tree. Tree models can also be executed in parallel.

Random Forest is an ensemble learning technique which is multiple individual decision trees. Multiple decision tree outputs are merged to get a stable model. Random forests can work in parallel however requires more computational resources but is computational less intensive compared to various neural models. Overfitting is a general issue faced by random forest, if hyperparameters are tuned optimally stable model can be achieved.

XGBoost is one more popular technique which is a boosting ensemble as opposed to random forest which is bagging ensemble. XGBoost is scalable and can be run in parallel and is much faster and accurate compared to Random Forest. XGBoost is used widely in language modelling but is computationally intensive.

Deep learning techniques have obtained higher acceptability in Natural language tasks lately. Often hand crafting features is both time consuming and difficult as there would be limited generalization ability. Deep learning addresses this issue by learning at multiple levels. As natural language data is mostly unstructured deep learning methods with self-learning is best suited for these tasks Convolutional Neural networks (CNN) came to fore and this architecture has been widely used and is also base architecture for various state of art models.

2.3.4 Deep Learning Models

Deep learning models mimics a human brain has the ability to learn from large amount of data. It is best suited for tasks such language modelling as these tasks are made up of largely unstructured data. In the literature review conducted above most of them use deep learning models here we will see most popular architectures used in text similarity models

2.3.4.1 Convolutional Neural Networks

(Collobert and Weston, 2008) used CNN architecture for natural language tasks. Here fixed length word embeddings are fed to a CNN network as input. Weights and biases were computed while training on the data and were assigned to respective neurons.

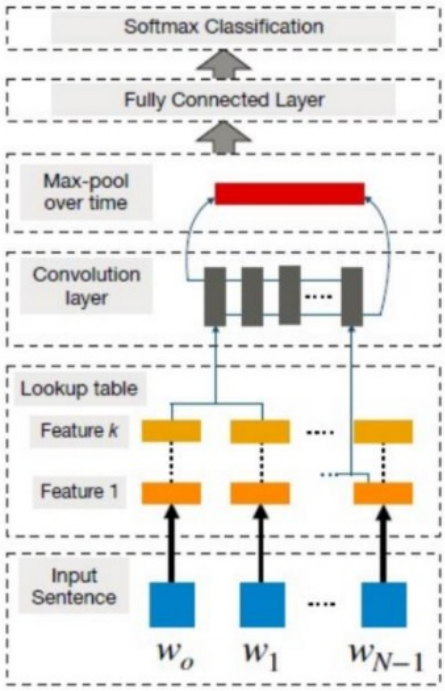


Figure 2-2 CNN Language Model architecture **(BENGIO ET AL., 2003)**

In the above figure input data is converted into word embedding either by using pre-trained word embeddings or manual word embedding mechanism. These word embeddings go through a Convolutional filter layer which produces feature maps. In the next layer we would have max pooling layer which reduce the complexity and a hidden convolutional layer, we can have as many convolutional layers as much the complexity required in the model. This output is given to fully connected layer which then is fed to an output function. CNNs architectures are widely used in many state-of-art models but CNNs performance degrades with long term dependencies. CNNs does not perform well when data is sequential as they do not preserve the sequential order.

Natural language data is mostly sequential but CNNs does not perform well with sequential data. (Wan et al., 2015) introduced Recurrent Neural Networks (RNN) which had a mechanism to retain information.

2.3.4.2 Recurrent Neural Networks

RNN were introduced by (Hopfield, 1982) but started gaining popularity with word embeddings coming to fore. RNN's were able to retain information and received previously, previous received sequence is processed and given to the next step in which the new sequence is available. RNN's became remarkably successful and started getting used in wide variety of tasks not only natural language but also in tasks like computer vision, times series and many other machine learning domains. RNN's introduced mechanism called back propagation (Werbos, 1990) which resulted in higher accuracy and started becoming a go to model for tasks like natural language processing.

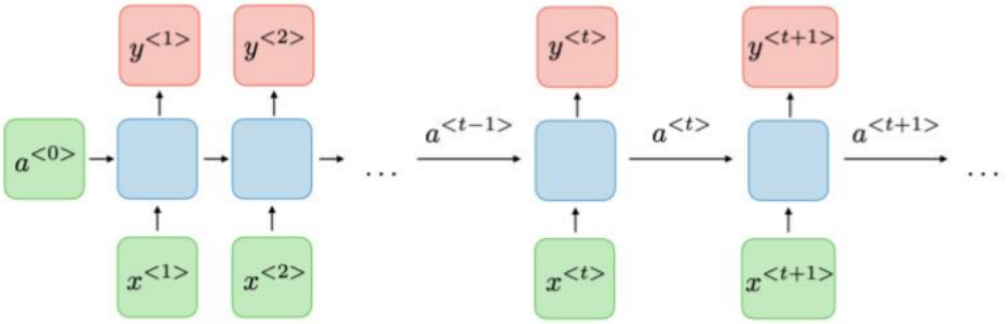


Figure 2-3 RNN architecture for Language Model **(OLAH, 2015)**

With longer sequences or large documents RNN's suffered providing better performance due to vanishing and exploding gradients. (Hochreiter and Schmidhuber, 1997) Came up with Long Short Term Memory (LSTM) to address the issues faced by RNN made up of traditional neural networks.

LSTMs were able to forget and update the information and was able to remember information for long time. LSTMs are even today used in most of the state-of-art language models. LSTMs is a powerful architecture, it suffered due the need of huge computational resources.

(Chung et al., 2014) came up with GRU simpler version of LSTM which has proven to be as powerful as LSTM. Both LSTM and GRU would run sequentially instead of having availability of GPU/TPU/Cloud environment parallel execution could not be achieved.

2.3.4.3 Seq2Seq architecture

(Wan et al., 2015) proposed Seq2Seq architecture in combination with RNN architecture. This architecture is widely used in NLP tasks and has seen wide acceptance. Primarily Seq2Seq architecture comprises of two components encoder and decoder. Both encoder and decoder are RNNs made of up of GRUs and LSTMs. Input sequence is first given to encoder which gives us a vector which is further fed to a decoder. However, for very long sequences holding it in fixed width vector proved costly, this also effected performance badly. (Chorowski et al., 2015) came up with attention mechanism to address this issue.

2.3.4.4 Attention Mechanism

Traditionally NLP tasks work in encoder-decoder mechanism. An encoder-decoder mechanism involves two separate network architectures one to encode the given input sequence which is the encoder. Another to decode the encoded sequence the decoder.

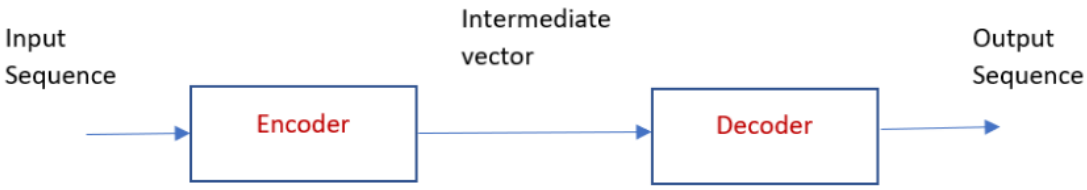


Figure 2-4 Encoder-Decoder Mechanism

In this architecture it requires it requires all elements of an input sequence to be processed before it gets propagated to decoder. Intermediate vector generated after the encoder will be the encoded output of all sequences taken as input. Decoder then makes predictions for the entire sequence. (Chung et al., 2014) study showed how encoder-decoder mechanism performance gets degraded with increase in the length of input sequence. In case of long-term dependencies traditional RNN's were not able to capture dependencies, even LSTMs were not able to capture. In case of long sentences not only the input sequences were gotten but also required exceedingly high computational resources to process them.

(Chorowski et al., 2015) wherein model focuses on certain parts of input to generate output resolved the issue faced in encoder-decoder setup. A new attention layer was introduced between encoder and decoder, encoder consists of Bi-directional RNN (Bi-RNN) which takes the entire input sequence, sequence information of current and past are captured and concatenated this makes it more effective as compared to regular RNN/CNN. Decoder for every target sequence it predicts generates a separate vector. Decoder searches for hidden states of the encoder input to fetch relevant information and weighs them accordingly.

$$c_i = \sum_{j=1}^{T_x} \alpha_{ij} h_j$$

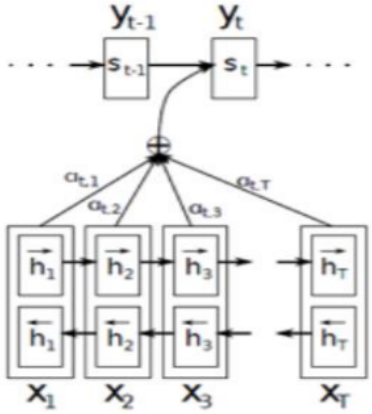


Figure 2-5 Attention Mechanism **(CHOROWSKI ET AL., 2015)**

Attention mechanism achieved state-of-art results in NLP tasks, this architecture became base for other advancements in the field of language modelling.

2.3.4.5 Siamese Architecture

Siamese network where two inputs which two patterns used to compare one another, and the output would be value which corresponds to similarity between the given patterns. (Mueller and Thyagarajan, 2016) proposed new MaLSTM approach wherein Siamese was adapted with LSTM architecture. Here 2 sequences are given to two identical network models one for LSTMa and another for LSTMb. Features from the sequences are extracted and are compared for similarity. Siamese network has a binary classifier at the output which checks if the given inputs belong to

same class or not. If inputs are of same class, then the both the inputs are deemed similar if not they are considered not similar.

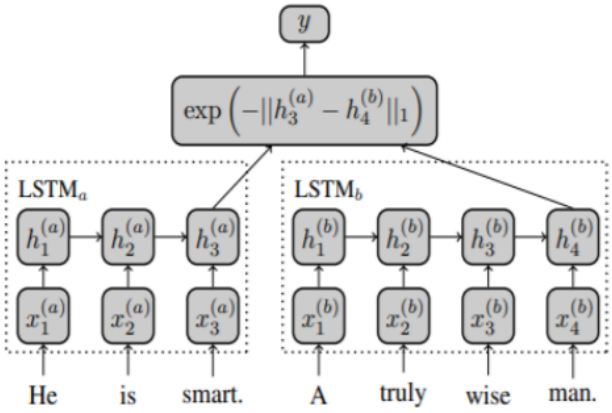


Figure 2-6 Siamese Architecture (**MUELLER AND THYAGARAJAN, 2016**)

Siamese architecture has found wider acceptability in text similarity tasks, it was also widely used in computer vision. Siamese network architecture involves two similar networks this would require more computational resources. Instead of giving state-of-art results Siamese architecture started losing popularity mainly due to the complexity involved.

2.3.4.6 Transformer Architecture

(Shaw et al., 2018) came up with transformer architecture which has revolutionized NLP modelling tasks. This architecture used self-attention mechanism which could be trained in parallel. This was a paradigm shift as all previous models were sequential in nature instead of availability of cloud environment which had massive computational power with modern GPUs and TPUs previous models were not able to make use of it in totality.

Like Seq2Seq models we have an encoding and decoding layer, but they are not sequential as they were in the traditional encoder-decoder models. Special attention mechanism called attention mechanism is present here. Main components which make up transformer model are position encoder, self-attention, encoder and decoder.

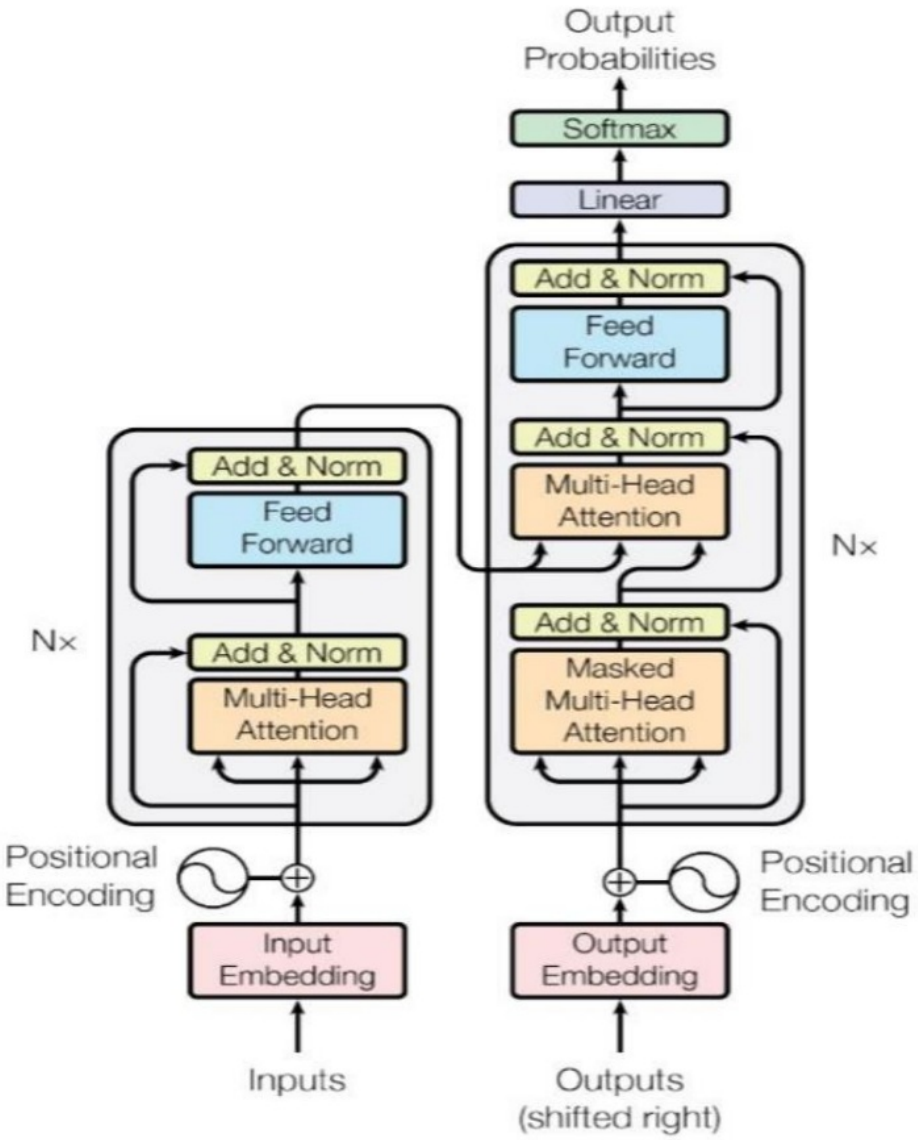


Figure 2-7 Transformer Architecture **(SHAW ET AL., 2018)**

Positional Encoding

Transformers as opposed to Seq2Seq architecture do not process the sentence sequentially, If a sentence is not processed sequentially then we lose out on capturing the structure or the grammar of the input sentence. To address this a d-dimensional vector having positional information of word embeddings with same size as that of the input embedding is added. Every odd step we have vector

created with cosine function and for every even step a vector with sine function is created. This vector will be on the same size as the input embedding and will passed with input embedding.

Encoder

This layer consists of Multi-Headed attention layer made up of eight self-attention blocks each and a Feed-Forward neural network. There would be six such encoding blocks in the encoding layer and each encoding block would have normalization layer. Input embedding is fed through this layer output of which will be fed to the decoder.

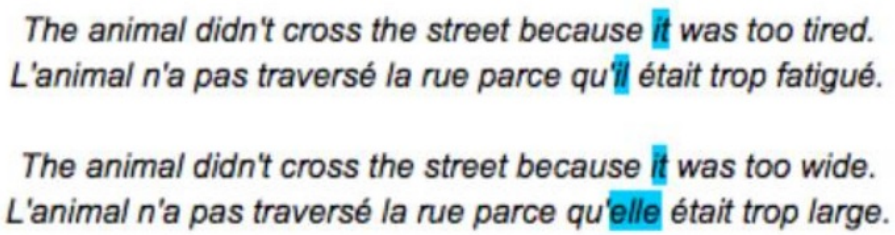
Decoder

Similar to encoder, decoder also has 6 blocks made up of Masked Multi-head Attention layer, as opposed to Multi-Headed attention layer in encoder. Also has one Multi-Headed attention block which takes the input from encoder. In the output layer Softmax function is used to predict the outcome.

Self-Attention

Self-Attention is a mechanism where each input token is attended with all other remaining tokens. This would help in understanding how much the contribution surrounding words make while calculating embedding for a given word.

In the below English to French translation, usage of word “it” infers to a animal in one sentence and street in the other. Transformer as seen translates both the sentences to French correctly. Transformer was intelligent enough to understand the context in which word “it” was used. In the self-Attention mechanism word “it” along with other words in the sentence would have been computed multiple times and concatenated the results.



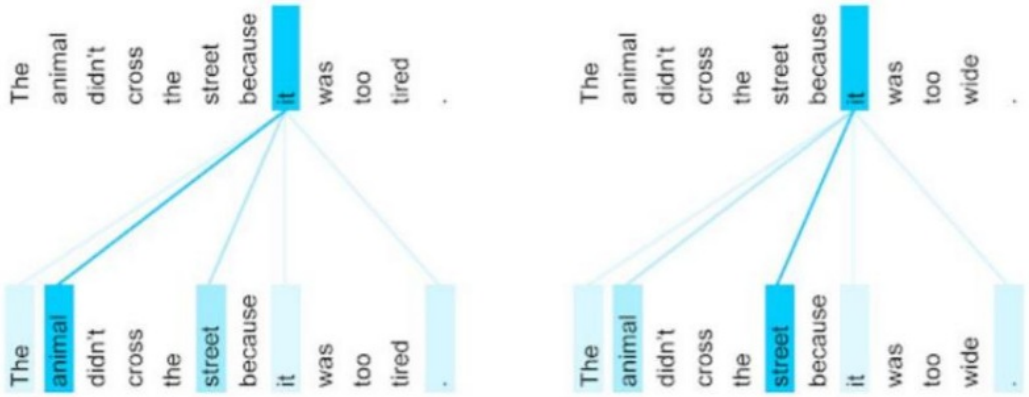


Figure 2-8 Attention Text Processing Mechanism **(SHAW ET AL., 2018)**

Multi-Head Attention

Model as proposed by (Shaw et al., 2018) computes attention multiple times this process is called Multi-Head attention. Final vector which the model outputs is from all the attention heads which gets concatenated.

Masked Multi-Head Attention

While in the token is given to the decoder, token which is not decoded yet will be masked so that decoders will not compute attention. This process is called Masked Multi-Head Attention.

Scaled dot-Product Attention

Scaled dot-Product is used to calculate attention as shown in the below formulae. Values of Q,K and V from the previous encoding block.

$$Attention(Q,K,V) = softmax(\frac{QK^T}{\sqrt{d_k}})V$$

Query-Q is the information required from the other token (Key).

Key-K is the information tokens have to give to another token (Query).

Values-V is the weighted value obtained from key and query.

Values of Q, K and V are multiplied with corresponding weight metrics from previous encoding block and the embedding E from the previous encoder block.

Attention score is obtained for a given vector from the Softmax function by multiplying these weights with Value vector.

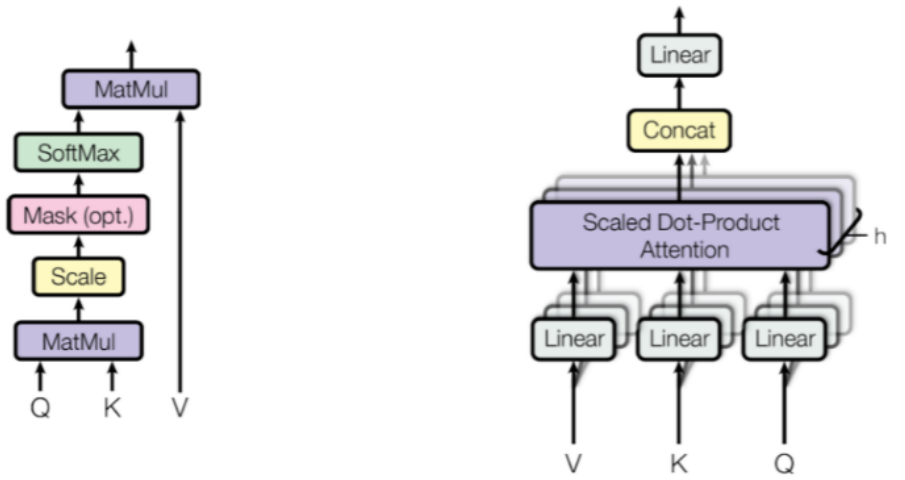


Figure 2-9 Multi-Headed and Scaled Dot-Product Attention **(SHAW ET AL., 2018)**

Transformer Data Flow

Each word is transformed as embedding matrix and with positional encoding is fed into both the encoder and decoder of the transformer. Decoder input will have mask for the words for which attention is not yet computed. Word once processed by encoder will be fed to the Multi-headed attention block in decoder, even the decoder output of six blocks is given to Multi-headed attention block. Finally, through a feed forward network after normalization and concatenation it is given to output SoftMax layer for prediction.

Usage of Self-Attention made it possible to eliminate back propagation which made transformer architecture to be executed in parallel. This architecture has achieved state-of-art of results in translation tasks and is getting adapted to various other NLP and computer vision tasks. It is also available in most of the open source packages and can be imported and trained with very minimal coding effort.

2.4 Transfer Learning

Pre-trained language models have bought in a revolutionary change in natural language processing tasks. These language models understood the natural language on which it was trained on, it was able to understand nuances in the language but also was able to learn interrelations between the words in that language. These pre-trained models were trained on large datasets (Zhou et al., 2018) Wikipedia and BookCorpus and (Proceedings of the Annual Conference of the International Speech Communication Association, INTERSPEECH, 2014) One Billion Word Benchmark.

These language models with a large knowledge base could be leveraged by downstream NLP tasks. Rather than training from scratch one could leverage already trained language models which would significantly bring down the number of trainable parameters in deep learning models. Transfer learning techniques have over the years proven to be an efficient and effective approach. For NLP tasks having less training corpus transfer learning is a boon. With transfer learning one could fine tune the language model with relatively small data and still achieve high accuracy. Even for large datasets transfer learning has proved as an effective modelling technique as there is no need to start training from scratch. (Deng et al., 2010) ImageNet dataset made transfer learning popular wherein ImageNet started getting used extensively in computer vision related tasks.

Transfer learning has been used extensively in NLP tasks nowadays. There are various state-of-art models available for NLP tasks like BERT, ALBert, ELMO and others. We will go through various language modelling, transfer learning techniques in the forthcoming sections.

2.4.1 ULMFIT

(Howard and Ruder, 2018) implemented ULMFIT (The Universal Language Model Fine-tuning) a transfer learning LM (Language Model) for NLP which provided state-of-art results. ULMFIT uses 3-Layer AWD-LSTM language Model, embedding size of input is 400. Each hidden layer has 1150 hidden activations and dropout is applied to all layers for regularization.

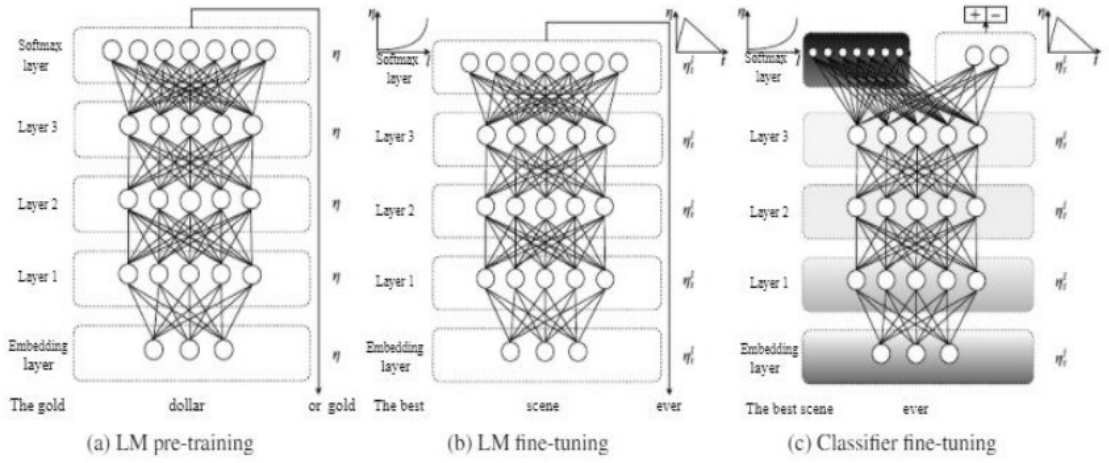


Figure 2-10 ULMFIT **(HOWARD AND RUDER, 2018)**

Three main components of ULMFIT are:

Language Model Pre-Training

WikiText-103 dataset consisting of 28,595 pre-processed articles from WikiPedia which had 103 million words was used for training. Training of this massive dataset is required only once further this pre-trained model can be leveraged by downstream applications. Advantage of training with such a large corpus makes it easy for datasets with lesser data in downstream, as it improves convergence of model and accuracy even with less data.

Language Model Fine Tuning

Specific to the task one will have to fine tune the pre-trained model to make use of the learning of the transferred model. Authors have come up with novel methods to achieve fine tuning.

Discriminative fine tuning: Each layer learns capture information of different type one can use different learning rates for each layer rather than using the same.

Slanted Triangular Learning Rates: To adapt to task specific features model should converge quickly to refine its parameters. Authors propose this method where initially learning rate increases and subsequently delays.

Classifier Fine Tuning

Authors have added two linear layers which can used for task-specific implementation. These layers have a RELU activation function with batch normalization and dropouts. Parameters in these layers can be learnt from scratch. Concatenated output of hidden layers is fed to first layer this is called Concat pooling. Gradual Unfreezing is novel approach wherein some layers are unfrozen gradually rather than training from scratch.

ULMFIT heralded a new era in NLP by introducing transfer learning which helped various NLP tasks with lesser data to provide good results. As this was trained with a large corpus most of the NLP tasks did not require to train from scratch which eventually resulted in high performance with very less training.

2.4.2 GPT

Language models requires large amount of labeled data for training this limits the applicability I various domains as they lack data which is annotated. (Radford and Salimans, 2018) came up with language model which uses a semi supervised approach for understanding the natural language using a combination of supervised fine-tuning and unsupervised pre-training. This model uses transformer architecture and has outperformed in various NLP tasks like natural language inference, question answering, semantic similarity, and text classification.

2 main components of GPT are:

Unsupervised pre-training: In this step model tends learn a large unsupervised corpus text.

Supervised fine-tuning: Here the model trained in the unsupervised mechanism will be adapted with task specific labelled data.

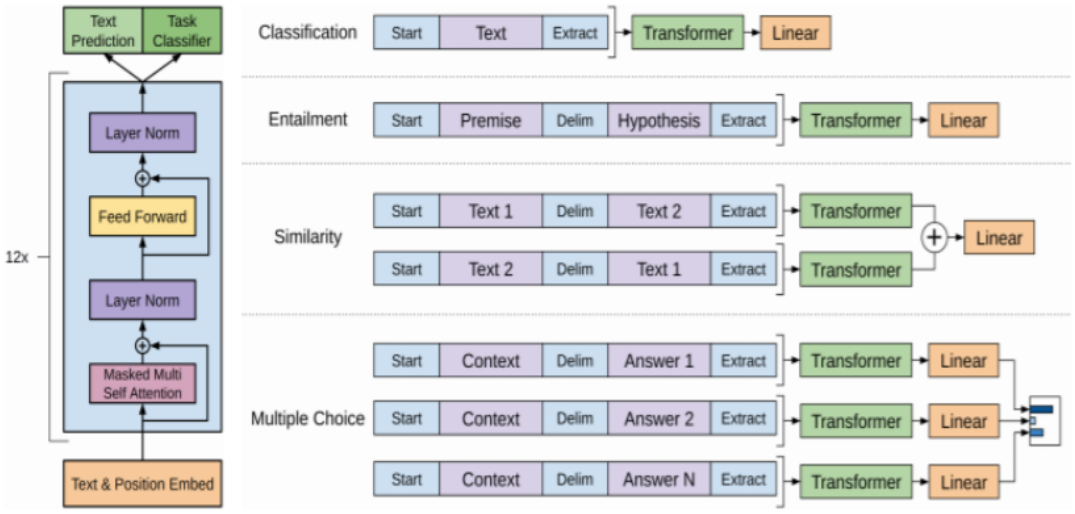


Figure 2-11 GPT **(RADFORD AND SALIMANS, 2018)**

Pre-training on a large corpus model was able to acquaint significant knowledge and ability to process long range dependencies. Down streams tasks training converged quickly there by reducing time for training. Study has shown this architecture was able to outperform state-of-art results of 9 datasets out of 12 considered for the study.

2.4.3 BERT

(Devlin et al., 2019) came up with Bidirectional Encoder Representations from Transformers (BERT) completed changed the course of NLP tasks. What makes BERT different from its predecessors like GPT, ELMO and ULMFIT is BERT is deeply Bi-directional, first Bi-directional unsupervised language pre-trained with plain text corpus. Existing pre-trained models are all Uni-directional, which cannot capture context from both directions. Author proposes two models BERT Base and Bert Large.

- BERT Base has a total of 110 Million parameters, architecture includes 12 Hidden Transformer Layers with hidden size of 768 and 12 attention heads.
- BERT Large has a total of 340 Million parameters, architecture includes 24 Hidden Transformer Layers with hidden size of 1024 and 16 attention heads.

BERT Base model architecture is similar to GPT architecture however BERT takes longer time to converge owing to the Bi-Directional complexity involved n the architecture.

In the transformer architecture the word which is not yet predicted will be masked before it is given to decoder, however if the model is Bi-directional, decoder could get token without mask. Authors came up with masked language modelling and next sentence prediction to overcome this issue.

Masked Language Model

Masked Language Model (MLM) is a technique wherein 15% of the input is masked in each and every batch provided for training. In these 15% not all records are masked only 80% among them is masked, 10% is not masked same word will be sent and another 10% some other word will be sent. Percentage of Masked words is kept at 15% as it will create a mismatch between pre-training and fine-tuning.

Next sentence Prediction

Next sentence prediction task where next sentence has to be predicted, was trained during the pre-training phase of model. This will make model to learn about the relationship between the sentences.

Masked Language Modelling Along Next Sentence Prediction happens the pre-training phase as opposed to other models where it happens separately.

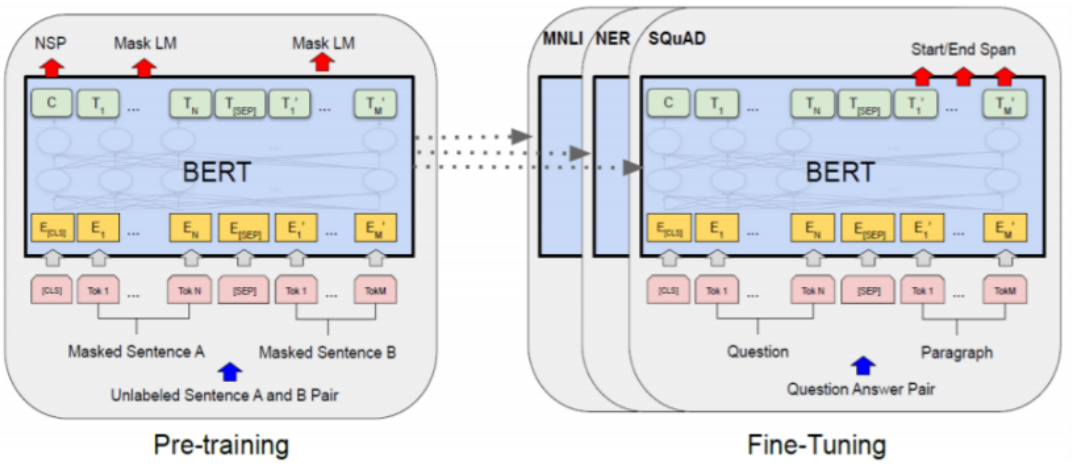


Figure 2-12 Overall pre-training and fine-tuning procedures for BERT **(DEVLIN ET AL., 2019)**

Main components of BERT

Pre-Training: In this Model is trained with large corpus of BookCorpus dataset (800 million words) and Wikipedia (250 million words) is used. Activation function used is GELU (Gaussian Error Linear units) and for regularization dropout is used.

Fine-Tuning: Task specific inputs and outputs are plugged into BERT, where all parameters are fine tuned.

BERT has achieved state-of-art results on 11 datasets. It has been very consistent model which has proven to give good results. BERT introduced Bi-Directional paradigm in NLP transfer learning and has been widely adapted in various NLP tasks.

(Andreas Chandra) have used BERT on the same dataset "Quora Question pair dataset", Accuracy of 97.08% was achieved which is more than any other method used in find text similarity on this dataset. However, BERT being accurate and providing state-of-art results on many benchmark datasets. Sheer size of BERT made it computationally very heavy.

2.4.4 Sentence BERT

(Reimers and Gurevych, 2020) proposed a variation of BERT, BERT which is used with Siamese architecture. Authors have shown for the use case of text similarity for corpus of 10,000 sentences BERT takes close to 65 hours whereas the proposed Siamese BERT architecture takes only 5 seconds.

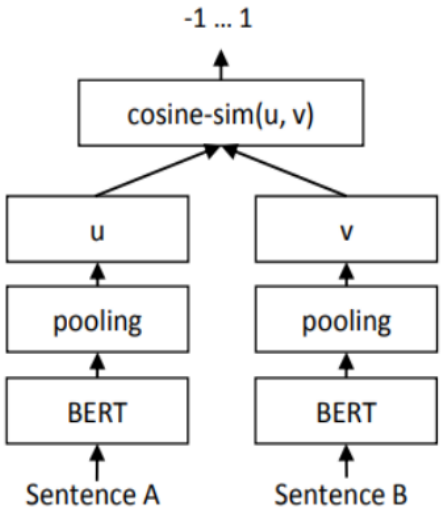


Figure 2-13 Sentence Bert **(REIMERS AND GUREVYCH, 2020)**

SBERT adds one extra pooling layer after the BERT layer to provide fixed length sentence embedding. u and v are the fixed length sentence embeddings obtained from pooling which in turn are computed to find similarity using cosine similarity function.

BERT maps sentence to vector space which is not suitable for text similarity operations like cosine-similarity. SBERT overcomes this by using SBERT architecture fine tunes BERT with Siamese architecture. The proposed approach was not only faster but also provided better results.

2.4.5 ALBERT

(Lan et al., 2019) came up with A LITE BERT FOR SELF-SUPERVISED LEARNING OF LANGUAGE REPRESENTATIONS (ALBERT) which has 18x lesser parameters as compared to BERT. ALBERT has 4 architectures base, large, xlarge and xxlarge. Backbone of ALBERT is BERT it uses same activation function GELU and transformer encoder model.

Model		Parameters	Layers	Hidden	Embedding	Parameter-sharing
BERT	base	108M	12	768	768	False
	large	334M	24	1024	1024	False
ALBERT	base	12M	12	768	128	True
	large	18M	24	1024	128	True
	xlarge	60M	24	2048	128	True
	xxlarge	235M	12	4096	128	True

Table 2-2 Bert and AlBert performance metrics **(LAN ET AL., 2019)**

Three main design choices ALBERT has made over BERT

Factorized embedding parameterization: ALBERT achieves factorization of embedding parameters by decomposing embedding matrices into two smaller matrices.

Cross Layer Parameter Sharing: ALBERT shares parameters across all layers.

Inter-sentence coherence loss: A new loss is proposed based primarily on coherence.

Design choices have made ALBERT to be a smaller model as compared to BERT. However instead of being smaller ALBERT-xxlarge has outperformed BERT-Large.

2.4.6 Distil BERT

(Hinton et al., 2015) came up with new compression technique called Knowledge distillation. Distillation is a technique wherein behavior of large model is mimicked by a smaller model. (Sanh et al., 2019) introduced Distill BERT based on the concept of distillation of knowledge.

Knowledge Distillation: Is compression technique where a student model is trained to have the same behavior of the part model.

DistilBert uses the same kind of architecture as used in BERT. Number of layers are decreased by factor of 2 and token type embeddings along with pooler are removed. All operations used are highly optimized using linear algebra. Number of layers are reduced as authors taught reducing in last layers did not impact much in terms of computational efficiency.

DistilBert has proven as efficient as BERT even though it is 40% smaller and 60% faster. It has proven to be 97% percent as efficient as BERT in many general-purpose language modelling tasks. DistilBert being a smaller and accurate model is very useful as it requires very less computational resources can be used be build language models on lesser hardware and still achieve state-of-art.

2.7.6 Tiny Bert

(Xiaoqi Jiao) came up with TinyBert which is compressed version of BERT. As in DistilBERT, TinyBERT also used Knowledge distillation student-teacher model to compress BERT however they do the distillation at two stages.

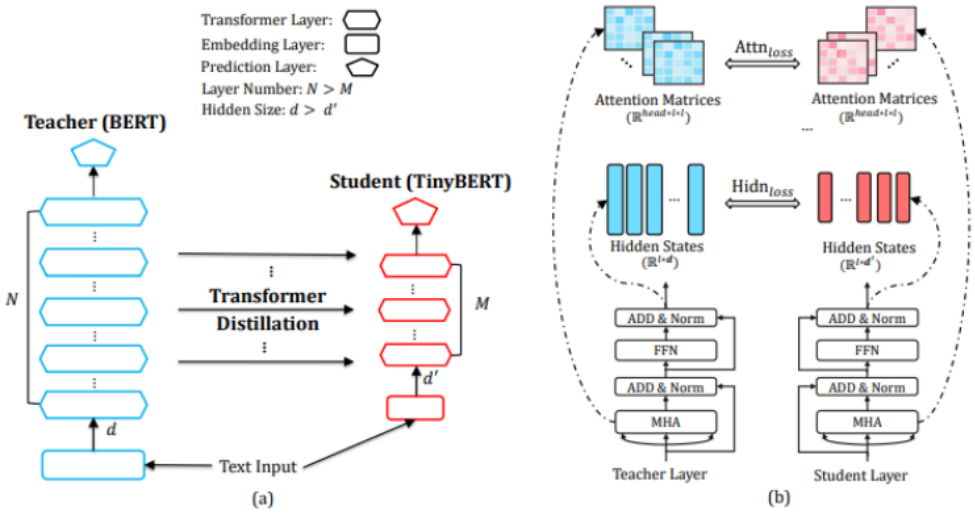


Figure 2-14 Tiny Bert **(JIAO ET AL., 2019)**

Transformer-layer Distillation

The proposed distillation includes distillation (b) both attention based and state-based distillation. (a) is the framework for transformer distillation. Attention based distillation is based on (Clark et al., 2019) findings which showed attention weights learned from BERT can capture linguistic knowledge. The proposed method transfers the linguistic knowledge of teacher BERT to student TinyBERT. Prediction Layer and embedding layer distillation is same as it is doe for attention distillation.

TinyBERT is 96% as effective as BERT being 9.4x faster and 7.5x smaller. TinyBERT ad DistilBERT both based on Knowledge Distillation has provided state-of-art results, as these are smaller models, we can have lesser hardware and achieve results as good as BERT.

2.4.7 XLNet

Transformers can only process fixed length sequences, BERT models are based on transformer models. BERT models neglect the dependency between masked positions and suffer from pretrain-finetune discrepancy. (Yang et al., 2019) proposed model based on Transformer-XL framework was able to overcome these issues. Auto regressive pre-training method used by XLNet enables learning bidirectional contexts by maximizing the expected likelihood over all permutations of the factorization order.

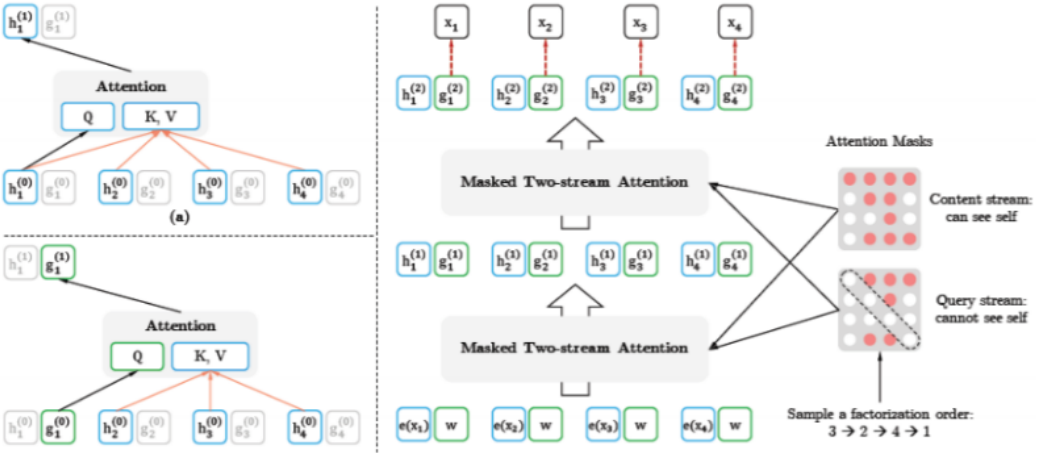


Figure 2-15 XLNet **(YANG ET AL., 2019)**

XLNet outperforms BERT and provides state-of-art results in most of the benchmark datasets.

2.5 Discussion

This section of Literature Review we saw how Language Models has evolved. From the initial string-based approaches to pre-trained embeddings natural language processing has evolved with advancement machine learning techniques, Transfer Learning in recent years have transformed Language models in recent years. Initially text similarity was carried out mostly by string-based approaches like cosine similarity, Jaccard distance and others distance mechanisms which used detect similarity at the character and word level. These string-based methods were very naïve did not have the capability to learn from underlying and predict on unknown datasets.

Machine learning approaches bought word embeddings to fore, initial word embeddings techniques like count vectorizer, TF-IDF, Bag-of-Words and N-Grams were used with machine learning techniques like Naïve Bayes, Logistic regression, SVM, Tree and Boosting models. Major drawback of these approaches where they were not able to understand the semantics of the language. Task of similarity majorly suffered as little difference in the sentence structure could not be captured.

Deep Neural networks revolutionized Natural Language Processing as it was able to learn syntactic and semantic information from given input text. Various pre-trained embeddings like Word2Vec, Glove, FastText and Poincare came to fore. These pre-trained word embeddings provided state-of-art results and was extensively us various language modelling tasks. Drawback with pre-trained embeddings were not able to capture context of the words also required large learning models. Text similarity context of the word plays an important role. Aim of this study is to find optimal model larger models may not suffice the need of this study.

Advancements in deep learning techniques helped addressing the issue of processing sequential data. RNN and CNN architecture were able to mitigate this issue effectively. Both RNN and CNN failed with input having long term dependencies. LSTMs were able to address issue of Long-Term dependencies many different variants of LSTMs like Bi-LSTM started giving state-of-art results. LSTM architecture was way too complex and required a simpler architecture. GRU with a slight modification of LSTM architecture was able to achieve same performance being lighter than LSTM. Sequential models built with LSTM and GRU instead of being powerful and accurate was not able work in parallel this resulted in longer training time. Major drawback in these architectures are they are computationally costly also takes lot of time to be trained.

(Chorowski et al., 2015) came up with attention mechanism in which each token with all remaining tokens are considered while predicting output token. (Shaw et al., 2018) Transformer architecture based on self-attention allowed language models to train in parallel. Transformers and Attention models not only reduced complexity of deep neural models, but it also heralded a new era of transfer learning. Transfer learning has changed the way language models are built in recent years. ELMO, BERT, XLNet and GPT are providing state-of-art results. Even if the data sets are smaller when used with these transfer learning models accuracy of the model so built is high. Knowledge Distillation technique (DistilBERT) wherein a large model (teacher) has been distilled to student

retaining characteristics of the parent model has made transfer learning more viable option than ever. Transfer learning models are high accuracy models which requires very less training efforts, this comparative study should provide an optimal model which both fast and accurate. This novel comparative study should augur further research in this field.

2.6 Summary

We started this section by understanding nuances of natural language processing how it has supported the task of text similarity, advances it has had over the years. We have seen how initial approaches worked and how natural language processing and machine learning models evolved. Briefly we saw the architectural advances of language models over the years till the present day. We have seen every limitation of a particular architecture has led to a new architecture which has addressed the limitation. There have been various milestones over the years be it machine learning techniques, followed by pre-trained word embeddings to the present-day transfer learning. This study intends to explore various methods available to build a language model which is both accurate and fast.

3. Research Methodology

3.1 Introduction

The research undertaken in this study can be categorized as Applied Exploratory Quantitative Research.

Goal Based: Applied Research as this study is to explore existing methods to determine text similarity which is both accurate and fast.

Research Objective: Exploratory Research here we explore different language modelling techniques.

Information Sought: Quantitative as we are using Quora duplicate question dataset.

In this study we will explore various language modelling techniques, specifically the focus would be on language modelling methodologies which are computationally efficient.

As this is a comparative study, we will explore various combinations of language modelling techniques. The key steps to determine text similarity in this study is depicted in the workflow below:

- Understanding the dataset.
- Selecting part of data as a subset if the data volume is high.
- Pre-processing the data.
  - Data will be cleansed by removing stop words and other characters like commas, question marks, etc as they will not ass much value.
  - Will explore methodologies like NER and POS which will categorize the words as nouns, adjectives, etc.
  - Tokenized words will be reduced to inflectional form using stemming or lemming.
- Splitting the data using different splitting techniques. Data will be split into three sets training, testing and validation.

- Balancing data using different sampling techniques if required.
- Embedding using TF-IDF.
- Embedding using pre-trained word vectors Glove and Word2Vec.
- Implementing supervised learning techniques.
- Implementing deep learning techniques.
- Implementing transformer-based transformer learning using DistilBert and ALBert.
- Implementing ensemble techniques.
- Evaluating performance both in terms of accuracy and time taken.

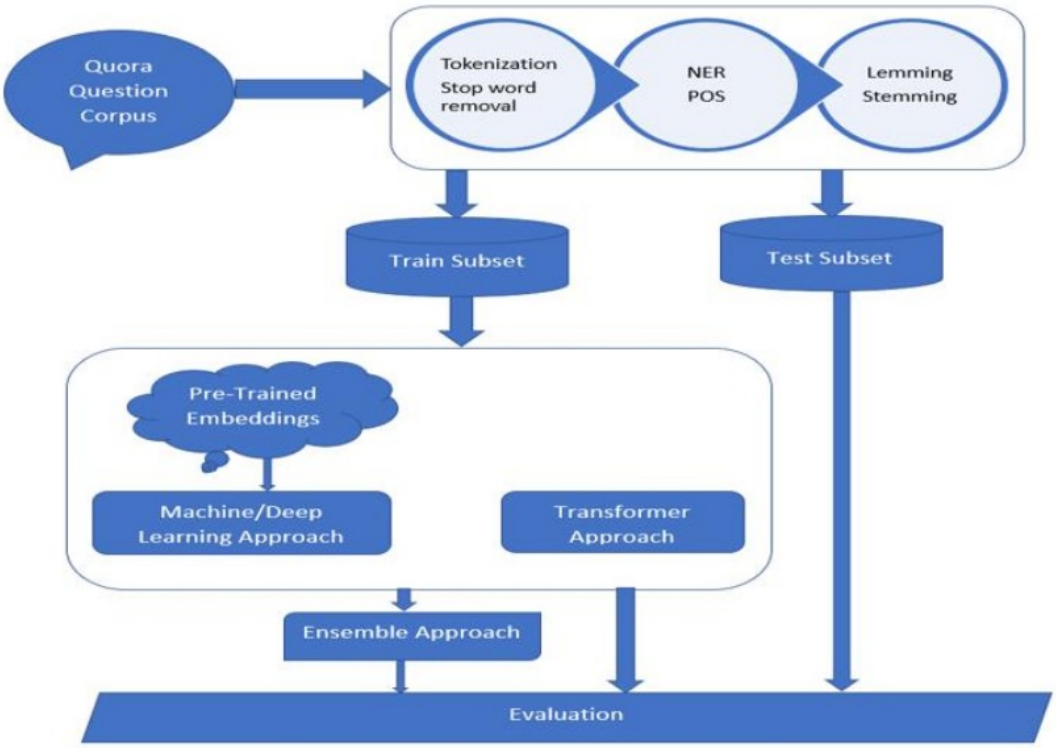


Figure 3-1 High Level Process Flow

3.2 Dataset Description

As Quora is a general-purpose knowledge sharing platform we could get questions in all domains. Each record has a pair of questions (question1 & question2) and a target class(is_duplicate) which indicates whether the question is duplicate or not. Below table has details of data fields in the dataset with its associated description.

Data Field	Description
Id	ID of the question pair.
qid1	ID of question1.
qid2	ID of question2.
question1	Text content of question1.
question2	Text content of question2.
is_duplicate	Flag which indicates the given question is duplicate or not.

TABLE 3-1 QUORA DATASET DESCRIPTION

Dataset contains 2,425,740 question pairs in training set, we will be considering training set for our analysis as the test set does not have labels provided with. There are 1,530,162(63%) question pairs with value 0(indicating no match) and 895,578(37%) with value 1(indicating match).

As there are 63% of values which are tagged to 0 and 37% values tagged to 1, underlying data is imbalanced. We will have to apply data balance techniques to balance the data which would result in a stable language model.

In our study we will only consider columns question1, question2 and is_duplicate columns. Columns Id, qid1 and qid2 are not required as they do not add any value.

3.3 Data Pre-processing

Data Pre-Processing is a step where in data cleansed and treated to make it consumable for modelling mechanism.

3.3.1 Data Cleaning

Check for any NULL/Blanks in fields question1, question2 and is_duplicate. Check count of these columns so there is no row with missing data.

3.3.2 Lower Casing and remove unwanted characters

Will convert the textual data to lowercase it will be helpful in resolving duplicates words. All characters except numbers and letters can be removed as they add no value.

3.3.3 Stop word Elimination

A stop word is a commonly used word (such as “the”, “a”, “an”, “in”) which can be taken out as they add no value to the model.

3.3.4 Stemming

Stemming is a way of reducing inflection in a given word to its root form. Stemming converts the words or a collection of words to its root form stemmed word may be an invalid word in the English dictionary.

3.3.5 Lemmatization

Both lemming and stemming results in reducing the feature space, the key difference between them is that lemming converts words to the root form which will be a valid word in the dictionary.

3.3.6 Text Augmentation

This is a process of supplementing textual data with information by providing semantic insights than what is present in original data. Example: POS tagging, NER.

3.3.7 Class Balancing

Models tend to be biased with the majority class in case data is imbalanced. Data fed to the language model for training has t be balanced. (Padurariu and Breaban, 2019) Data balancing can be performed on dataset either by over sampling or under sampling.

Over Sampling: In case of over sampling minority class gets increased by generating synthetic instances of minority class.

Under Sampling: in case of under sampling majority class gets decreased to match the number of minority class.

Over sampling techniques requires synthetic instances of minority class to be generated and would be a time-consuming process as this research intends to find a language model which can be trained faster, under sampling will be used.

3.4 Transformation

In this step textual data will be converted to vector representation which can be fed to a machine learning model for training. As this study is about comparative analysis of various word embedding approaches will be explored.

3.4.1 Tokenization

As pre-requisite word embeddings needs to be tokenized in words or letters called as tokens, these tokens are converted into numbers which will be represented in vector space.

3.4.2 Word Embeddings

Word embeddings are fixed size vector representation which represents given text input is mapped into vector of real numbers. Word embeddings can either be created from scratch or can be created by using pre-trained word embeddings. Word embeddings can be created from scratch using various available techniques like count vectorizer, TF-IDF vectorizer, continuous bag of words or N-grams.

TF-IDF

Is a frequency based embedding method which takes into account not just the occurrence in the current input but the entire the Corpus. TF-IDF penalizes on words which occur more frequently. Rare words are given higher weightage and words which occur commonly are given lesser weightage.

TF-IDF=TF*IDF

Term Frequency (TF) = (Number of occurrences of word T in document) / (Number of terms in document)

Inverse Document Frequency (IDF) = Log (Number of Documents) / (Number of documents word has appeared)

For Example, consider the below statements:

Text 1: This is a text similarity task.

Text 2: This is a language modelling task.

Let us see TF-IDF values for the words in sentences above. Words (this, is, a and task) are repeated in both the sentences so we will get same TF-IDF score for these, Words (text, similarity, language and modelling) appear only once so will have same score.

TF-IDF score for “This” would be

TF=1/6

IDF = log (2/2) = 0

TF-IDF = (1/6) * 0 = 0

TF-IDF score for “text” would be

TF=1/6

IDF = log (2/1) = 0.301

TF-IDF = (1/6) * 0.301= 0.050

As we see words which are repeated in both the documents has lesser relevance compared to the one which is not repeated.

There are several pre-trained word embeddings from which given text can learn from:

Word2Vec

Word2Vec is a pre-trained word embedding which will convert word to vector preserving its semantic meaning. Word2Vec has the ability to capture context of the words within a given input also can capture relationship between other words. Word2vec is a two layered neural network takes

corpus as input and produces a vector space. Word vectors are positioned in a such a way that words which are related contextually are close to each other. Word2Vec comes in two flavors.

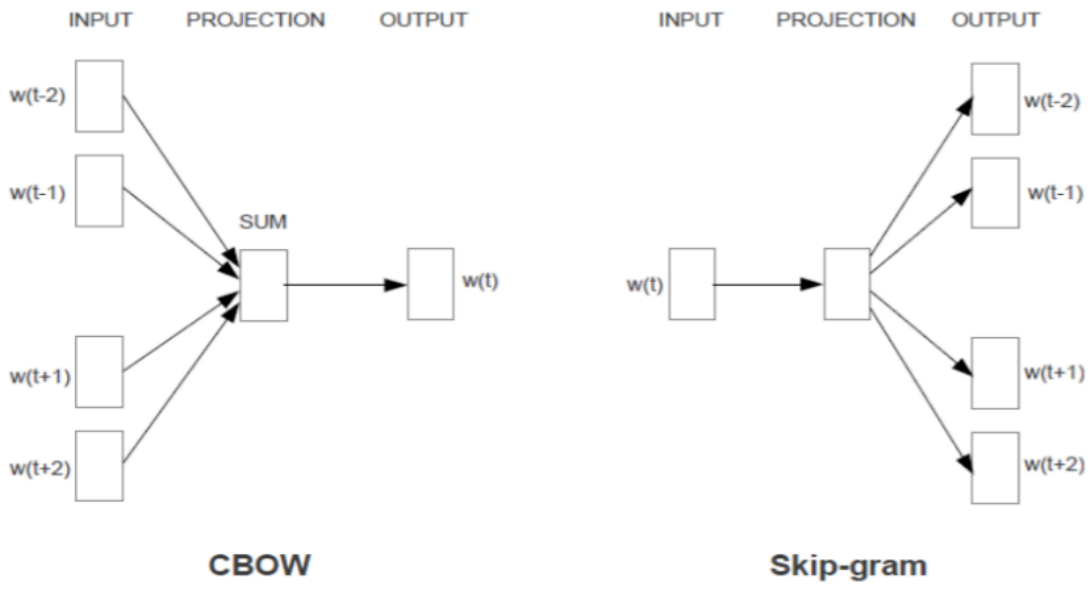


Figure 3-2 Representations of CBOW and Skip-gram **(JOULIN ET AL., 2017)**

Continuous Bag of Words (CBOW)

CBOW predicts words based on the surrounding context of words.

Skip Gram

Skip gram is inverse of CBOW it predicts context words based on target.

Skip gram is more effective with lesser data and less frequent words whereas, CBOW is faster and is more effective with frequently used words.

GloVe

Global Vectors for Word Representation is an extension to the word2vec method for efficiently learning word vectors, developed at Stanford. Glove captures both local and global statistics of a corpus to come up with word vectors. Glove achieves capturing meaning in vector space and takes

advantage of global statistics count. Unlike Word2Vec Glove learns from co-occurrence matrix and trains word vectors. Let us consider below sentence to see how Glove processes it.

“The cat sat on the mat”

With window size of 2 sentence would be converted to below co-occurrence matrix

	the	cat	sat	on	mat
the	2	1	2	1	1
cat	1	1	1	1	0
sat	2	1	1	1	0
on	1	1	1	1	1
mat	1	0	0	1	1

Figure 3-3 Co-Occurrence Matrix

Consider two words “ice” and “steam” differ in state but are form of water. It is expected to appear near to each other.

Probability and Ratio	$k = solid$	$k = gas$	$k = water$	$k = fashion$
$P(k\|ice)$	1.9×10^{-4}	6.6×10^{-5}	3.0×10^{-3}	1.7×10^{-5}
$P(k\|steam)$	2.2×10^{-5}	7.8×10^{-4}	2.2×10^{-3}	1.8×10^{-5}
$P(k\|ice)/P(k\|steam)$	8.9	8.5×10^{-2}	1.36	0.96

Figure 3-4 probability Matrix

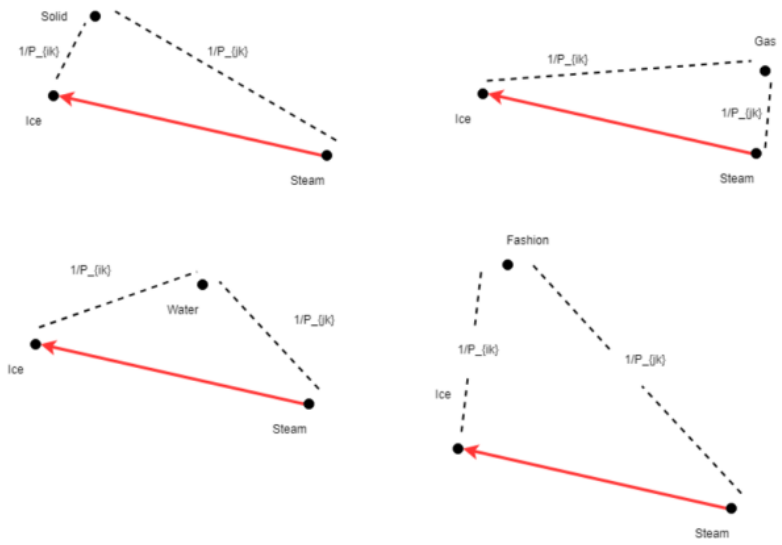


Figure 3-5 Behavior of vector distances to a probe word

Word embedding built from either of the technique is provided as in input to machine learning model which in turn builds a task specific language model. Deep learning models have an embedding layer which is the input layer of the model. Embedding layer also has weights and bias associated with it, deep learning models assign weights and bias appropriately during the process of back propagation when training the model.

Transformer models are unique language models which takes text input and has the ability to convert text into word embedding using the fine-tuning model.

3.4.3 Padding

Sentences given for training or the sentences which come as input to the model for prediction is usually of variable length. These sentences will generate tokens which are nit uniform, these tokens must be normalized before it is fed into the language model. Padding is a mechanism were input vectors of varied length are padded to make it a vector of similar length as the other vectors.

Especially in deep learning models which are more prevalent now, embedding layer to which embedding is provided is part of the network it is a must to provide embeddings of same length. Also it is important to have embeddings with appropriate size, embedding size should not be to

small as it will miss information in case of long term sequences and also it should not be too big as it will make model complex.

3.4.4 Splitting data

Any machine learning model requires a dataset which it can train on and dataset which it can validate trained model on. It is important while building a model to see how model predicts on unseen data set, so it is general practice wherein dataset is split before it is trained. Usually data is split in train, validation and test sets, train set is used for training the model validation set is used by the training model to evaluate its training finally test set is used to validate how model predicts after training.

Splitting of data also is an important decision in modelling data below are the different data splitting techniques.

Random Split: Here data is split into train, test and validation sets using a rando sample. That is in case of classification data, data of different classes are split randomly.

Stratified Split: Here the data is split into test, train and validation and distribution of data on a class variable would be of same proportion.

K-Fold Split: Here data is split into K different sets, Language model is applied on the K sets average of K-sets is taken. Here the split can random or stratified.

3.5 Modelling

As main objective of this study is comparative analysis, we will consider various language processing techniques with language modelling techniques. We also explore ensemble approach which considers learning from various models. Below flow diagram depicts the end to end modeling approach carried out in this study.

- Different feature engineering techniques will be explored.
- Different Word embedding methods will be implemented with both machine and deep learning approaches.

2 different types of embeddings one pre-trained and one frequency based embedding we implemented

Steps to be followed while building TF-IDF embedding

- Use a SKLearn tokenizer to tokenize the text into tokens from the training set.
- SKLearn TF-IDF vectorizer must be fit on tokenized words.
- Test also needs to be tokenized on the same lines as train set.
- TF-IDF vectorizer transform method must be applied on it.
- Both test and train set need to be padded to make all vectors uniform in size.

Steps to be followed while pre-trained embedding is used

Pre-trained embeddings like Glove and Word2Vec is available in Colab and Kaggle, if it must run in local system the embedding has to be downloaded.

- Use tokenizer tokenize both train and test set.
- Creating an embedding from the source data of the same vector length as it is in pre-trained word embedding used.
- Assign values corresponding to the word in the pre-trained vector to the new embedding created.
- Save the word embedding created for the dataset for both train and test set.

- Data split also we will explore different approaches like stratified, random and under sampling techniques.
- Transformers like Distil Bert and AL Bert will be explored.
- To build language models below algorithms will be implemented.
  - Classical machine learning algorithms: Logistic Regression and Naïve Bayes.
  - Boosting algorithms: LightGBM and CatBoost.
  - Deep learning algorithms: GRU.

- Ensemble implementation would be done by combining inputs of more than one model using both hard and soft voting approaches.
- Compare different approaches and evaluate them on performance and time efficiency.

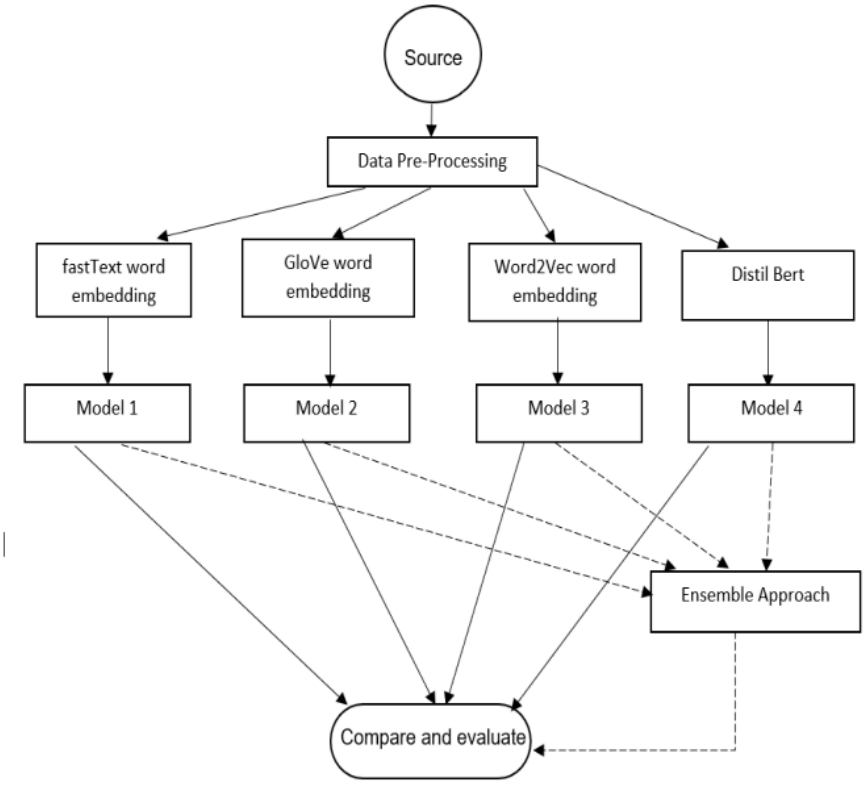


Figure 3-6 End to End Language Model

3.5.1 Machine Learning algorithms

Machine learning models like Logistic Regression, and Naïve Bayes will be explored as part of this study. We would use different word embeddings which involves both embeddings trained as part of language modelling using techniques like NER, TF-IDF and count vectorizer, also would be exploring pre-trained word embeddings like Glove and FastText.

We will also explore various word pre-processing techniques like lemming, stemming and stop word removal with these algorithms. In this study we will mainly explore Naïve Bayes and Logistic Regression as these models have proven over the years to train faster and also provide better performance.

3.5.1.1 Naïve Bayes

Naïve Bayes is based on Bayesian approach where probabilistic classifiers are applied on input features. It has been widely used in various language modelling tasks, it is both fast and requires less computational resources also simple to model. It has proven to perform well with smaller datasets.

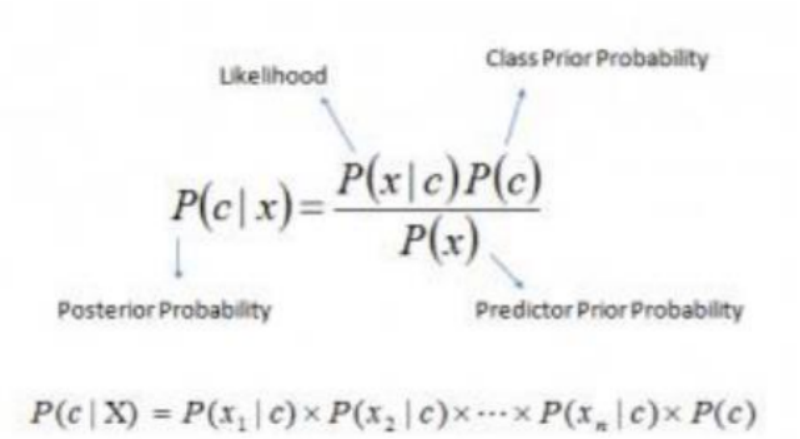


Figure 3-7 Naive Bayes Probability Equation

P(c|x) is the posterior probability of class (c, target) given predictor (x, attributes)

P(c) is the prior probability of class

P(x|c) is the likelihood which is the probability of predictor given class

P(x) is the prior probability of predictor

Naïve Bayes is considered a bad estimator, probability prediction is mostly ignored. It almost unlikely to get predictors which are independent in real world, Naïve Bayes assumes independence among predictors.

3.5.1.2 Logistic Regression

Logistic Regression is one more machine learning technique which has proven to perform on language modelling tasks. It is also simple to model, naïve bayes is a generative classifier whereas logistic regression is a discriminative classifier. Logistic Regression performs a tad better than naïve bayes, also it has various hyperparameters to tune the model. In case of overfitting Regularization techniques can be leveraged to build a stable model. Logistic Regression using a logit function predicts probability of binary event.

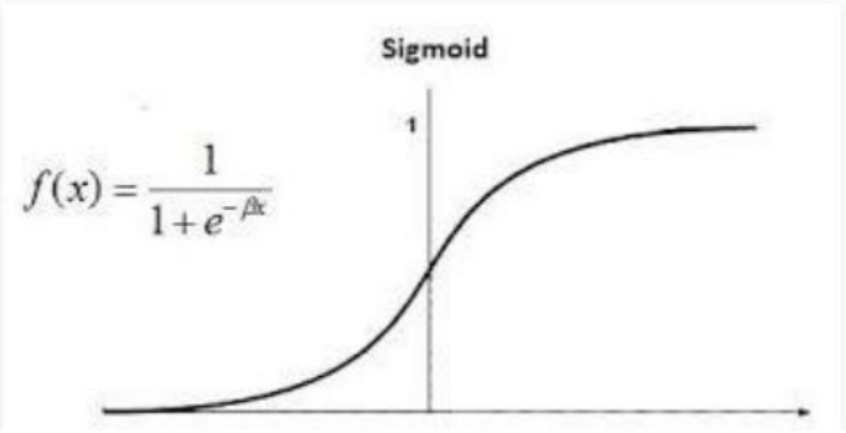


Figure 3-8 Logistic Regression Sigmoid Curve

Logistic Regression is evaluated using Maximum Likelihood Estimation (MLE). Using Logistic Regression, it is difficult to obtain complex relationships and non-linear problems cannot be solved.

3.5.2 Boosting Models

Boosting is an ensemble technique where multiple weak model learnings is combined to make it a strong model. AdaBoost, Gradient Boosting and XGBoost have been extensively in various modelling tasks and has proven track record of providing high performance. AdaBoost and Gradient Boosting are sequential models and are time consuming to train. XGBoost instead of being parallel requires massive computational resources. Introduction of LightGBM and CatBoost

which takes less computational resources and fast have made these techniques adaptable in various modelling tasks including language modelling.

3.5.2.1 LightGBM

LightGBM is a leaf-based algorithm which splits the tree leaf wise, leaf wise tree models generally reduces loss compared to level-wise algorithms. Leaf wise splits enhances accuracy but will result in increased complexity which will eventually lead to overfitting.

Figure 3-9 Level-Wise Growth

(Ke et al., 2017) came up with LightGBM performs equally or slightly better than XGBoost. LightGBM splits the tree leaf wise as opposed to other tree models which split the depth or level wise. Leaf based algorithms generally reduces loss compared to level-wise algorithms which results in higher accuracy. However, leaf wise splits will result in increased complexity which will eventually lead to overfitting. Parameter max-depth has to be fine tuned appropriately to avoid overfitting.

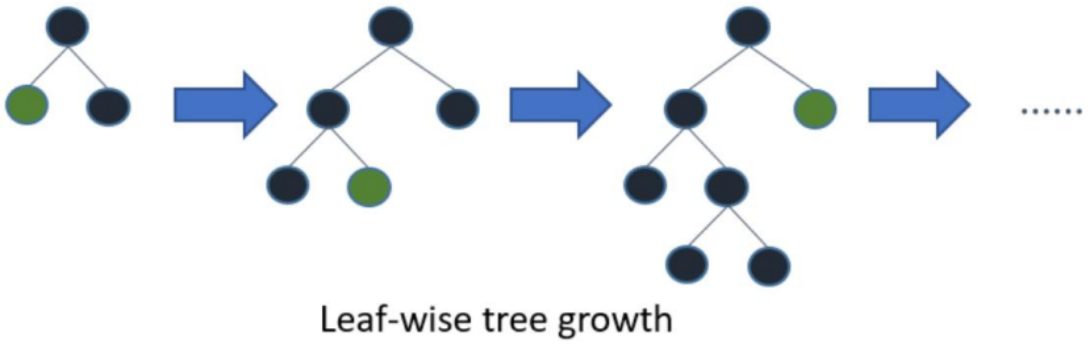


Figure 3-10 Leaf-Wise Tree Growth

LightGBM is 20 times computational faster compared but achieves same results. LightGBM is nowadays gaining popularity and has been used many NLP tasks. Parameter max-depth must be used appropriately to avoid overfitting.

3.5.2.2 CatBoost

CatBoost can return high accuracy models even with smaller datasets. Major advantage CatBoost has over other boosting models it can take categorical or test data directly and it internally converts into numerical form. CatBoost can detect overfitting in the training stage even with default parameters we can attain accurate model without overfitting. One major advantage seen is with little or no parameter tuning, using default parameters CatBoost is giving very accurate results.

However, we will evaluate the CatBoost using both word embeddings and also without transforming into word embeddings. CatBoost is gaining wider acceptance in various modelling tasks including language modelling.

(Chandra and Stefanus, 2020) used CatBoost (Prokhorenkova et al., 2018) for the finding duplicate questions in the Quora duplicate dataset. CatBoost has proven to perform better than XGBoost with better computational efficiency.

CatBoost name is derived from **Cat**egory **Boost**ing, it can work on data which has varied data types. Unlike LightGBM or deep learning models, CatBoost can return high accuracy models even with smaller datasets. Textual or categorical data can directly be fed to CatBoost which internally converts into numerical form. This is major advantage as effort to convert categorical or textual

data is no longer required. CatBoost can detect overfitting in the training stage even with default parameters we can attain accurate model without overfitting. Catboost is getting adapted in wide variety of domains, natural language tasks are also using CatBoost extensively. One major advantage seen is with little or no parameter tuning, using default parameters CatBoost is giving accurate results.

3.5.3 Deep Learning Approach

As seen in the earlier chapter LSTM has been widely used in language modelling as this study focuses more on models which are simple training has to be faster.

LSTMs are sequential networks which tend to hold information for inputs having long sequences. LSTMs like RNN is a chain like model wherein previous received input after processing is fed as input to the next sequence.

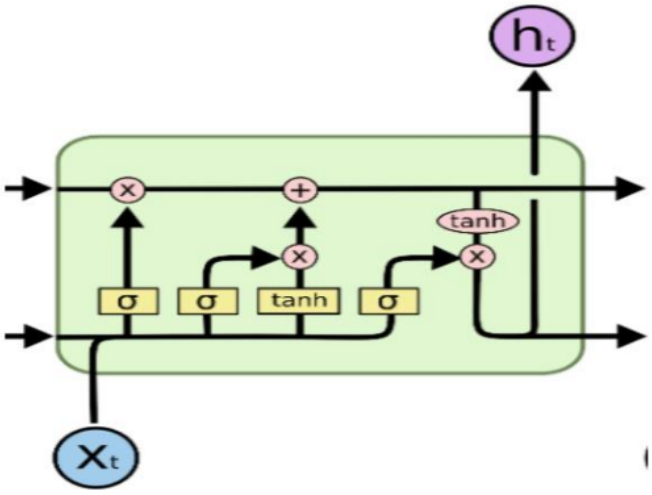


Figure 3-11 LSTM Structure **(OLAH, 2015)**

While processing lengthy sequences it is often required to forget learnt information and update with new information also required is to remember information for a long time. LSTM is primarily built to achieve this task. Principal components of LSTM are cell sates and gates, gates are of three types in LSTM forget gate, input gate and output gate. Cells and gates can be fine-tuned to either remember information for a long sequence or forget and update whenever required.

Cell state is a unique feature which carries information from beginning to end of the chain, this makes LSTM to carry across long ranges. Forget gate at every iteration tells whether the cell state must retain or forget information with help of sigmoid function.

LSTMs perform better than other deep learning models or any other traditional models but they are computationally intensive which will require more computational resources. (Chung et al., 2014) came up with GRU simpler version of LSTM which has proven to be as powerful as LSTM.

Gated Recurrent Units

Gated Recurrent Gates commonly known as GRUs, combine forget and input gate to make an update gate. Cell state and output gate make reset gate. GRUs performance is almost like that of LSTMs. Unless large sequences where still LSTMs perform better, in almost all other cases GRUs provide same performance as LSTM instead of being computationally light.

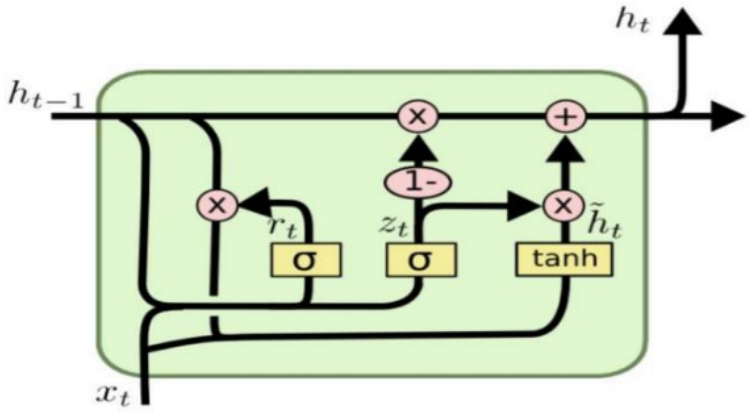


Figure 3-12 internal structure of GRUs **(OLAH, 2015)**

There were many other architectures which tried to simply LSTM architecture, (Gers et al., 2000) came up with an architecture wherein gates were allowed to see cell sate information. However, GRU and LSTM architecture was adapted more compared to any other architecture. Especially for NLP tasks GRU architecture performed exceedingly well. Both GRUs and LSTMs did not have the capability process parallelly they needed previous step output to be fed to the current step. This made them extremely slow and holding state for long time made them require more memory. Bi-directional architecture of both GRU and LSTM proved extremely costly instead of giving highly

accurate results it became less used owing to its computational needs. Modern cloud environment provides massive parallel capability GPU and TPU are available for parallel processing owing to its dependent architecture of current input waiting for the previous input's output LSTM and GRU was not able to make use of this.

GRU instead of being computationally intensive model we would like to explore a Bi-GRU with a smaller input embedding and a smaller number of hidden layers. As performance also is an important metric we are looking at. We would like to concentrate more on feature engineering techniques so that model can converge quickly thereby reducing the training time.

3.5.4 Transfer Learning

Transfer learning has revolutionized the natural language modelling. ELMO, BERT, GPT provide state-of-art results in all datasets they are applied on. These models have millions are extremely powerful and have millions of trainable parameters. They are large in size and also to fine tune to particular tasks requires more computational resources. DistilBERT and TinyBert transformer models are built using teacher-student mechanism from BERT. Instead of being smaller these models were as accurate as BERT was.

We will explore DistilBERT and TinyBERT in this study, various previous conducted research has shown how effectively these models are used to achieve state-of-art results.

3.5.5 Ensemble Technique

Finally, we would like to try ensemble technique applied on output of machine learning models used in the study. Ensemble techniques provide an interesting outlook as it captures learning of various models. Previously conducted research has not widely used ensemble techniques we would like to explore how this technique behaves in language modelling tasks.

Commonly used ensemble techniques are

Bagging in this technique similar leaners are used on sample populations, mean of all predictions are taken.

Boosting: It is an iterative technique which adjusts based on last prediction. Boosting decreases Bias error and in turn builds strong models.

Stacking: Here a learner model will combine output of different learners.

Each language model and each feature engineering technique capture information differently, combining knowledge of these different modelling techniques would be an interesting proposition we would like to try different combination of ensembles in this study.

3.6 Summary

In this section we briefly went through the methodology adapted in this study. We started initially with data pre-processing in which we discussed different techniques to cleanse the data and bringing to an inflectional form. Next, we saw "what is word embedding" and how crucial is it to represent textual data as word embedding. We explored different mechanism in place to build word embedding also we looked at different pre-trained word embeddings and saw how it can capture semantics of a language. After that we saw different machine learning techniques which are computationally efficient as main aim of or study to find language modelling techniques which are both accurate and computationally efficient, here we first saw algorithms like Logistic Regression and naïve bayes later boosting techniques like CatBoost and LightGBM. A GRU model which is known to be computationally intensive was chosen but over aim here is to keep model simple and focus more on feature engineering, by keeping model simple we aim to make it computationally efficient focusing on feature engineering would ensure us better accuracy. Finally, we saw how transfer learning is used effectively in modern day systems here also we choose smaller models like DistilBERT and TinyBERT which provide accurate results keeping in mind the computational complexity involved. Also, we discussed about ensemble approaches which is an area in which most previous researchers have not explored. As this a comparative study of various language modelling techniques we will be exploring all techniques discussed in various combinations.

3.7 Evaluation metrics

For performance evaluation of the language models various well-known evaluation parameters. The models in this study will be evaluated on below metrics.

- Accuracy: This can be defined as the proportion of the correct predictions out of total cases predicted. In cases of a dataset where the class imbalance is not present, this can be a good metric for evaluation.

- AUC: This is the Area Under the Curve of ROC and this illustrates to how much extent the model is good to segregate the predicted positive classes from the predicted negative classes. A good classifier model will usually have a higher AUC score, so this is one of the important metrics for evaluation.
- Precision: Precision can be measured as the proportion of the correctly predicted positive classes compared to the actual positive classes.
- Recall: This is also one of the useful metrics for classification and this measures the proportion of the positive classes identified correctly.
- F1 score: This is defined as a balance between the Precision and Recall and the value of this score is between 0 and 1
- Time taken is another metric which will be considered as an evaluation metric.

As objective of our research is find an optimal model which is accurate and time efficient, all proposed models will be evaluated on both these criteria. As we are using ensemble techniques also in this study time taken by each constituent model also would be criteria while evaluating.

4. Analysis and Implementation

4.1 Introduction

In this section we shall discuss about the different methodologies which we consider as part of the exploratory research. We will initially explore the data for better understanding of the underlying data, this step is very crucial influencing factor in building language models. We will also discuss on various data cleaning, pre-processing, machine learning techniques also transfer learning techniques.

4.2 Data Set Description

Data set used in publicly available Quora Question pairs dataset which can be downloaded from Kaggle.

Data Field	Description	Data Type
id	Unique column identifier	Object
qid1	Question ID 1	Object
qid2	Question ID 2	Object
question1	Text content of question1.	Object
question2	Text content of question2.	Object
is_duplicate	Flag which indicates the given question is duplicate or not.	Int

TABLE 4-1 DATASET DESCRIPTION

Dataset considered for the study has 6 columns with 404,290 rows of training data. Columns qid1 and qid2 are unique Id's which identify a question. Question 1 and Question 2 has actual text questions. Is_duplicate indicates whether the question pair is duplicate or not.

4.3 Exploratory data Analysis

Exploratory data analysis EDA in short helps one to understand the underlying data better. EDA provides insights on how to cleanse and model the underlying data. In total there are 537,933 questions in the data set.

There are 149,650 unique questions for the rows labelled 1 the ones which are duplicate. Out of 149,650 questions 46,032 questions are repeated more than once which is 30.75% of questions being asked. This is certainly a huge number nearly 31 percent of the questions are already answered.

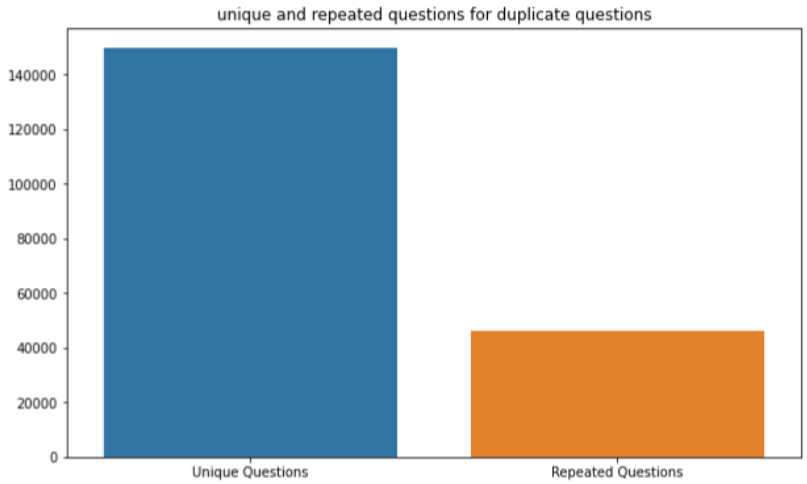


FIGURE 4-1 UNIQUE VS REPEATED QUESTIONS

For this study we will only be using 3 columns question1, question2 and is_duplicate.

Data Field	Description	Count	Data Type
question1	Text content of question1.	404290	Object
question2	Text content of question2.	404290	Object
is_duplicate	Flag which indicates the given question is duplicate or not.	404290	Int

TABLE 4-2 DATASET CONSIDERED FOR THE STUDY

There are no missing rows in any of the columns considered, there is no need to treat or impute missing data. is_duplicate column has two values 0 or 1, which indicates whether the question pair is duplicate or not. 1 indicates it is duplicate and 0 indicates it is not a duplicate. Language model to be built would be a binary classifier.

Is duplicate	Row Count	Percentage
0	255,027	63.08
1	149,263	36.92

TABLE 4-3 LABEL WISE COUNT

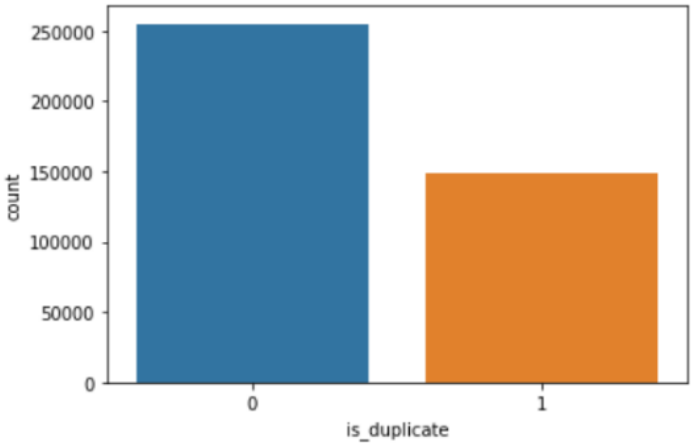


FIGURE 4-2 DUPLICATE VS NON-DUPLICATE COUNT

Binary classifier requires data to be balanced, if not model may either end up over to be an over fitted or under fitted model. Data set seems to be imbalanced where non duplicates have more rows of data as compared to duplicates.

Statistical Analysis

In this section we will statistically analyze inputs question1 and question2, this analysis will help us feature engineer the input texts which will eventually help us in build stable language model.

Number of words appearing in question is an important metric consideration as the embedding length to be used should accommodate all words used in the sentence.

Number of words appearing in Question 1

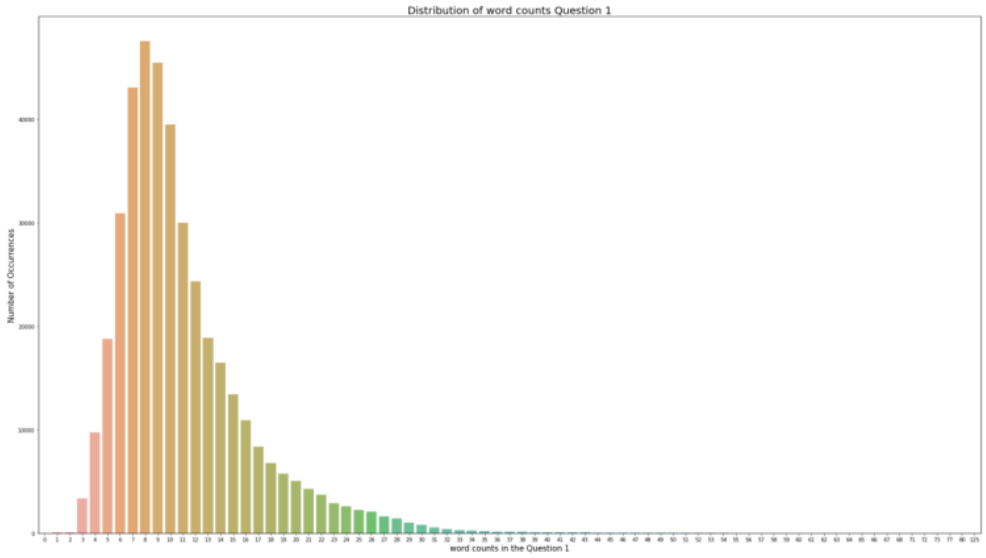


FIGURE 4-3 QUESTION 1 WORD COUNT DISTRIBUTION

Number of words appearing in Question 2

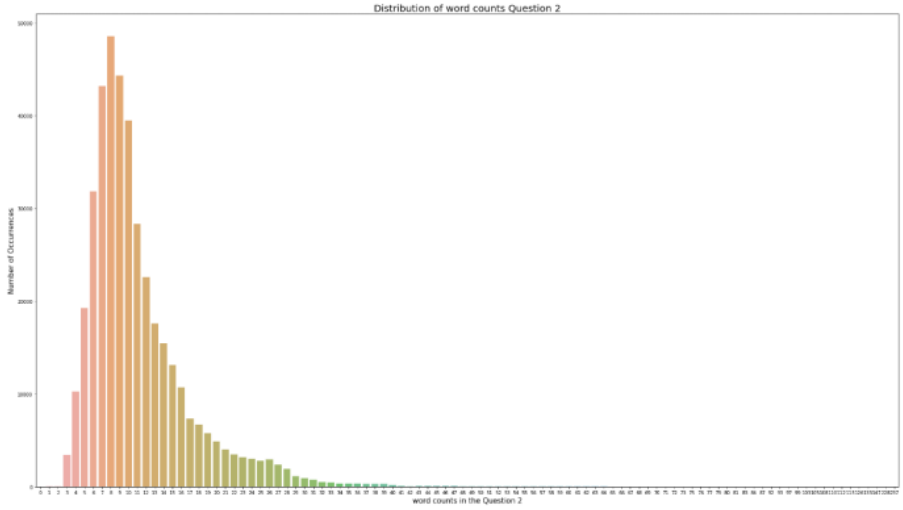


FIGURE 4-4 QUESTION 2 WORD COUNT DISTRIBUTION

Word Cloud

Word cloud visually depicts most common words used, higher the resolution more the words are used.

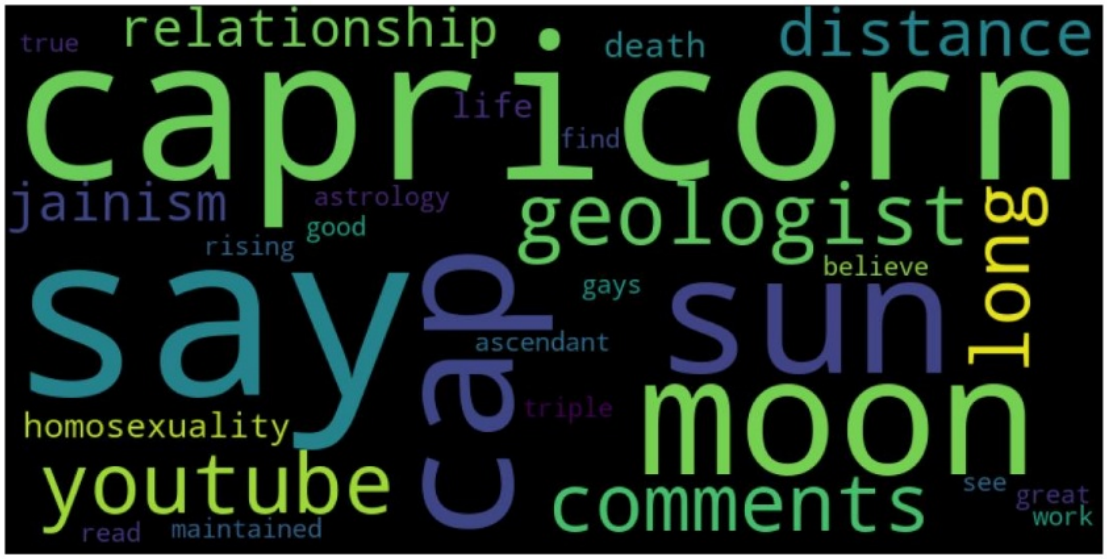


FIGURE 4-5 NON-DUPLICATE QUESTIONS WORD CLOUD

Word Cloud for duplicate questions

FIGURE 4-6 DUPLICATE QUESTIONS WORD CLOUD

Word cloud for non-duplicate questions

Word Cloud on a higher level gives us an overview of keywords used.

Duplicate questions: We see keywords like Capricorn, Sun, Moon, Astrology being used often. In case of non-duplicate questions, we see Step, Market, share, invest being used more.

	q1_length_dupllicate	q2_length_dupllicate	q1_length_nondupllicate	q2_length_nondupllicate
Count	149,263	149,263	255,027	255,027
Mean	9.85	9.86	11.58	11.95
Std	4.16	4.16	5.96	7.16
Min	1	1	1	1
25%	7	7	8	8
50%	9	9	10	10
75%	11	11	14	14
Max	80	60	125	237

TABLE 4-4 WORD COUNT DESCRIPTIVE STATISTICS

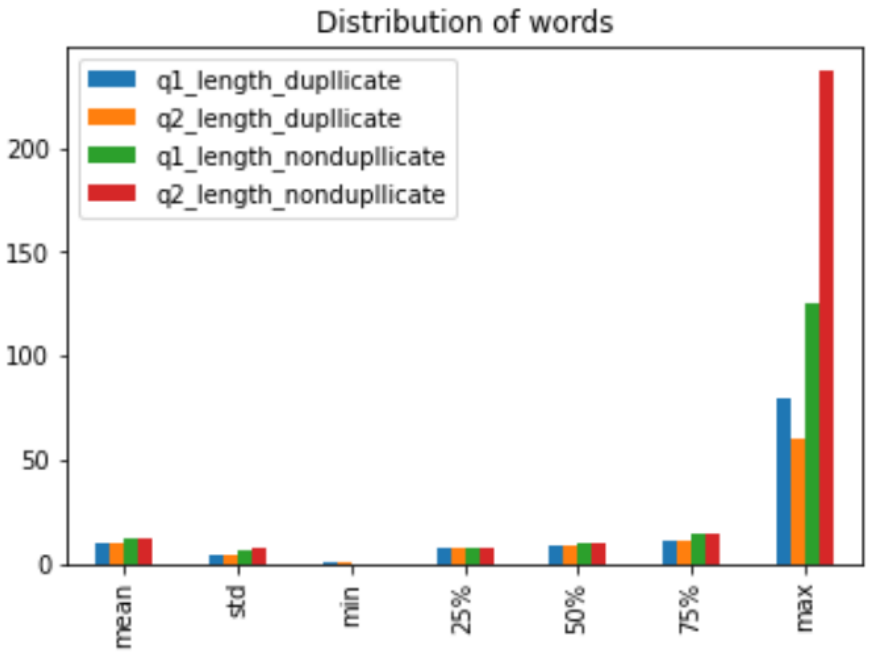


FIGURE 4-7 WORD COUNT DESCRIPTIVE STATISTICS

Non-Duplicates seems to have more words compared to duplicates. Overall length of both the questions in a pair seems to not differ more. Number of words in a question averages around 4-7 words, mostly we have smaller questions in the dataset and keywords play a significant role in determining similarity. Further will explore Unigram, Bi-Gram and Tri-gram word plots.

Unigram plot for duplicate and non-duplicate questions of top 20 words and number of times it is getting used can be seen in the plot below.

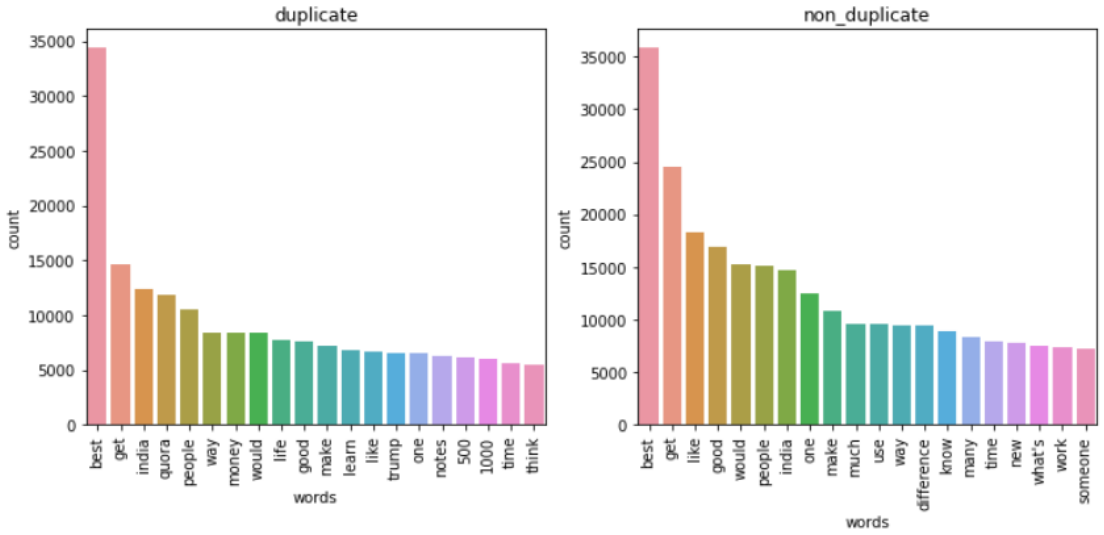


FIGURE 4-8 UNIGRAM WORD COUNT FOR DUPLICATE AND NON-DUPLICATE QUESTIONS

Unigrams we see words like best, people, India get used more. In non-duplicates also frequent word usage seems similar.

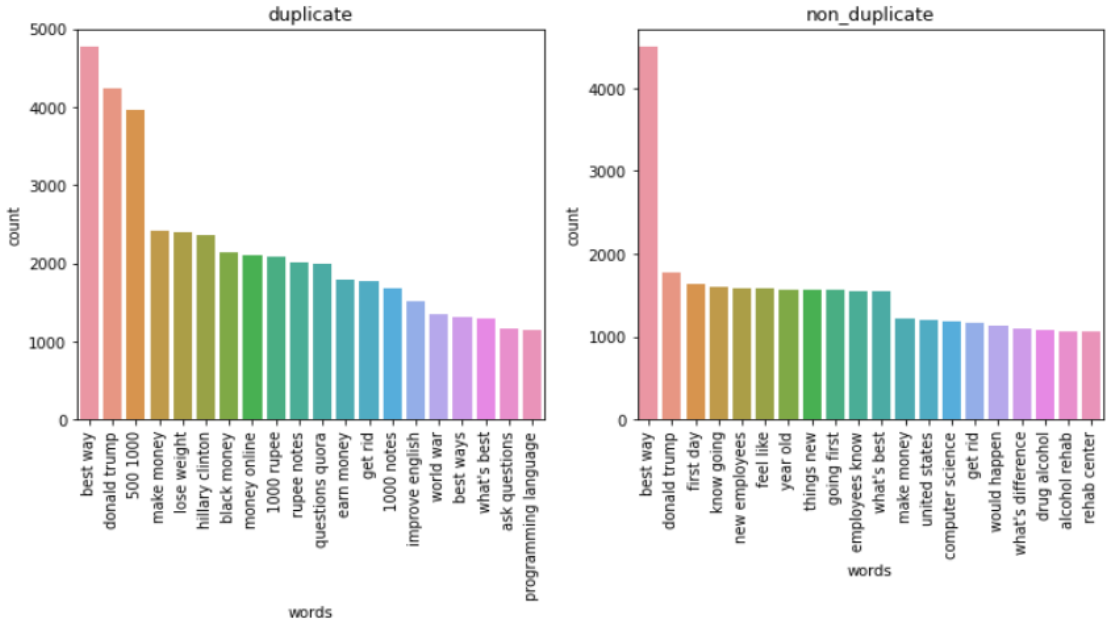


FIGURE 4-9 BI-GRAM WORD COUNT FOR DUPLICATE AND NON-DUPLICATE QUESTIONS

Bi-Grams 500-1000 rupees, Hillary Clinton, money, losing weight appears often in duplicate set, whereas employees, Donald trump, drug alcohol appearing more often. Compared to unigram here we see a difference between duplicate and non-duplicate more. This infers that N-Grams if used for feature engineering it is better to avoid One-Grams as that may not yield any good result.

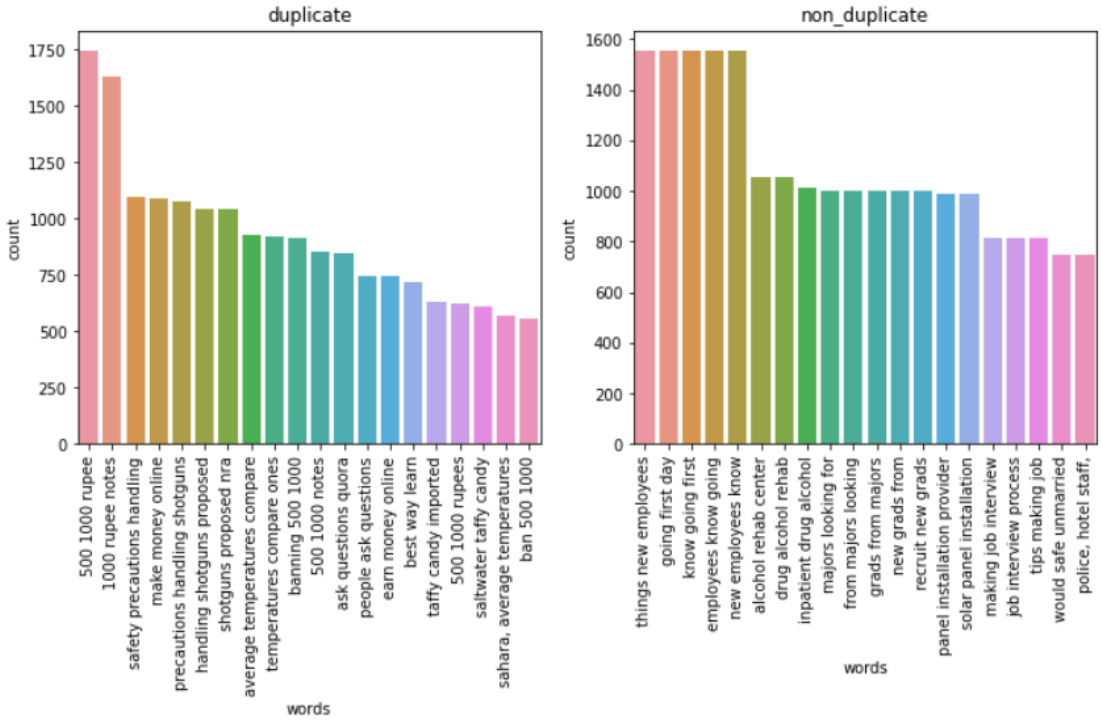


FIGURE 4-10 TRI-GRAM WORD COUNT FOR DUPLICATE AND NON-DUPLICATE QUESTIONS

Compared to unigrams and bigrams in trigrams we see more difference in duplicate and non-duplicate questions. It would be better to use trigrams as feature engineering technique.

We will further derive statistical metrics which helps with further deeper insights on the data set.

Feature Name	Description
freq_qid1	Frequency of question 1
freq_qid2	Frequency of question 2
q1len	length of question 1
q2len	length of question 2
q1_n_words	Number of words in question 1
q2_n_words	Number of words in question 2
freq_q1+q2	Sum of frequency of question 1 and 2
freq_q1-q2	Difference of frequency of question 1 and 2
word_Total	Sum of words in question 1 and 2
word_Share	number of words/ total words

TABLE 4-5 DERIVED STATISTICAL METRICS

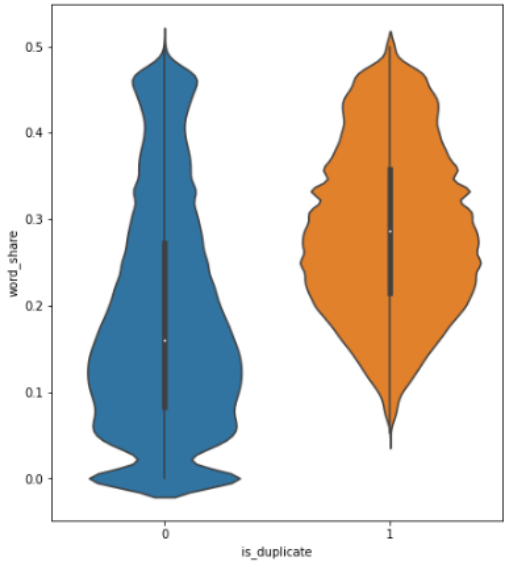


TABLE 4-6 VIOLIN PLOT DUPLICATE LABEL VS WORD COUNT

Violin Plot shows that the word-share and common number of words is more in question pairs which are duplicate as compared to the non-duplicate ones.

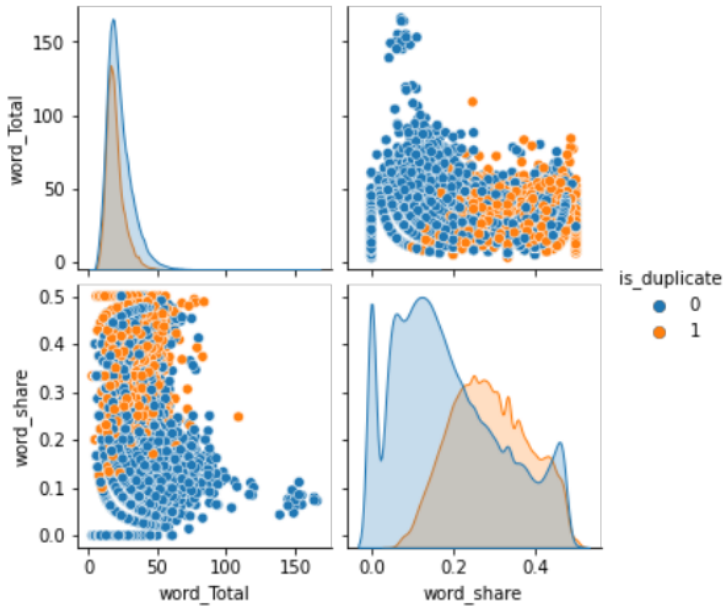


TABLE 4-7 PAIR PLOT FOR BI-VARIATE ANALYSIS

Pair plot above indicates that the duplicate questions are corelated whereas non-duplicates seem to be scattered.

4.4 Data Preparation

Data Pre-processing is one of the most important step in building machine learning models. Dataset has questions asked by users in Quora portal this is bound to have grammatical errors and spelling mistakes. Data pre-processing is required to clean and correct the data to be further processed.

Lower casing: Words used in sentence can be in lower case, upper case or camel case by lower casing it we will eventually eliminate duplicates.

New Line/ Extra Space removal: Extra spaces like new lines, tab spaces or any unwanted spaces will space to be considered. To avoid this happening while tokenization these spaces are removed.

Non-ASCII characters removal: Non-ASCII characters like “?”,” /” and others add no value to the language model, also becomes problematic when using pre-trained embeddings.

Stop word removal: Stop words like “the”, “and” and other common words add no value to the model. Removing this helps on focusing major keywords present in sentence.

De contractions: Words such as I'll, won't which used users while posting questions will not be present in the pre-trained embeddings. These contractions will be replaced with actual words for example I'll will be converted to "I will".

Spell check and corrections: As this is human entered questions there will be words with spelling mistakes which has to be corrected.

Data Split: Data set will be split 50:50 into training and testing sets, as this study is also about finding a time efficient approach for transformer models, we will go with lesser training set considering Number of parameters to be trained.

4.5 Model Building

In this study we explore various word embeddings along with various language modelling techniques. Steps taken in building this language model for this comparative model are:

Pre-processing:

Already pre-processing has been discussed in sections 4.3. Apart from the text pre-processing discussed already. Stemming, lemming and Pos tagging will be used as hyper parameters.

Tokenization and Embedding:

Words or sentences are converted into numerical vectors to be consumed in machine or deep learning language models. In this study we will be using various embeddings.

- Word2Vec
- GloVe
- TFIDF
- Bert (Distil-Bert, AL-Bert)

Pre-Trained Embeddings like Fast2Text or ELMo are not used in this study as they are resource intensive. Fast2Text is character level embedding which takes more time compared to GloVe and Word2Vec. Usage of TFIDF is due to the higher accuracy rate which it gives in non-pretrained embeddings. Count Vectorizer or bag of words were tried but the model performance was low

compared to TFIDF. Distil-Bert and AL-Bert was considered as transformer models rather others as they have lesser trainable parameters.

Class Imbalance:

As observed in exploratory data analysis we observed data being imbalanced. In this study we will address imbalanced data by stratifying data, under sampling and oversampling.

Stratifying data:

This is process wherein same proportion of data is maintained in all the subsets. As we use K-Cross validation while training we will maintain the same proportion in each sample.

Under Sampling:

In this sampling technique we will reduce the majority samples to match minority class.

Over Sampling:

In this sampling technique we will increase the minority samples to the majority class.

4.5.1 Machine Learning Models

In this section we will see various machine learning algorithms and hyper parameters considered in this study.

Model	Sampling	Embeddings	Hyper Parameters	Description
Naïve bayes	stratify, under sampling and over sampling	TFIDF, GloVe and Word2Vec	gaussian distribution	Gaussian is the preferred distribution for binary classification problems.
Logistic Regression	stratify, under sampling and over sampling	TFIDF, GloVe and Word2Vec	solver='sag'	sag solver used converge fast
Logistic Regression	stratify, under sampling and over sampling	TFIDF, GloVe and Word2Vec	C={1.e3,1.e4,1.e5}	Smaller the value stronger is the regularization strength
Light GBM	stratify, under sampling and over sampling	TFIDF, GloVe and Word2Vec	n_estimators={300,500,750,100}	It specifies number of trees in a Light GBM model
Light GBM	stratify, under sampling and over sampling	TFIDF, GloVe and Word2Vec	learning_rate={0.0001, 0.001,0.01, 0.1}	Controls how much co-efficient or model learns each time.
Light GBM	stratify, under sampling and over sampling	TFIDF, GloVe and Word2Vec	iterations={100,300,500} early_stopping_rounds= 10% of iterations	Number of iterations to run. Early stopping will stop if there is no improvement.
Cat Boost	stratify, under sampling and over sampling	TFIDF, GloVe and Word2Vec	learning_rate={0.0001, 0.001,0.01, 0.1}	Controls how much co-efficient or model learns each time.
Cat Boost	stratify, under sampling and over sampling	TFIDF, GloVe and Word2Vec	iterations={100,300,500}	Number of iterations to run.

TABLE 4-8 MACHINE LEARNING HYPER PARAMETERS

4.5.2 Deep Learning Model

In this study we are considering a simple GRU Model, below is the architecture of the model being considered.

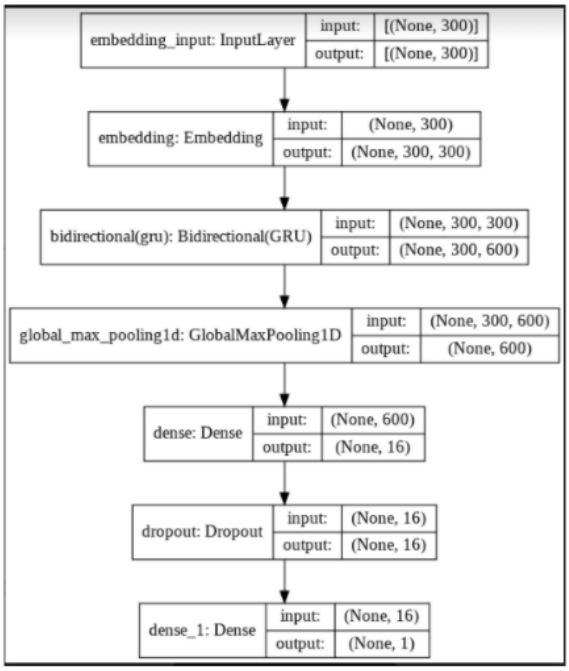


FIGURE 4-11 GRU MODEL ARCHITECTURE

As discussed in the literature survey lot of deep learning architecture with various pre-trained embeddings have been experimented on. In this study we will restrict to only one architecture, also no pre-trained embeddings will be used as already it has been used by many researchers. Hyper parameters for the GRU model considered are below.

Layer	Hyper Parameters	Value	Description
Word Embedding	Initialization Weights	TFIDF, GloVe and Word2Vec	Embedding layer or input layer where more than one embedding technique is evaluated.
GRU-Bidirectional	Activation Function	Relu	Activation function for fully connected layer, ReLU is defined as maximum of 0 or input value {y = max(0, x)}
GRU-Bidirectional	dropout	0.2	Used to regularize the model
Dense	Activation Function	tanh,sigmoid	Activation function for dense layer
GRU	Loss	SparseCategoricalCrossentropy,binary_crossentropy	Loss function for binary classification
GRU	optimizer	SGD, Adam, AdaGrad	Optimizer to find best learning rate for the model
GRU	Evaluation Metric	Accuracy, F1-Score, precision, recall	Model evaluation metrics after each epoch
GRU	batch size	512	Number of sentences to be fed into training model
GRU	Number of epochs	10	Number of times model will be run on entire data set.

TABLE 4-9 DEEP LEARNING GRU MODEL HYPER PARAMETERS

4.5.3 Ensemble Modelling

Ensemble modelling is a technique wherein more than one model is used. In this study we will consider stacking ensemble technique. In each ensemble we will have 3 machine learning models with different combinations. Stacked output of the three models will be given as input to LightGBM model output of which will be the output of the ensemble model. All hyperparameters are same as mentioned in machine learning section (4.5.1).

4.5.4 Transformer models

Transformer models like Distil-Bert and AL-Bert will be used, reason why only these models are chosen rather than any other Transformer model because the size of model. As this study considers computational cost and overall training time as a metric. As there are 2 input strings, we will add separators [SEP] to separate two strings and [POS] at the start of the first string. By adding [SEP] between two strings Bert tokenizer will be able to identify as two separate sentences while tokenizing.

Here the train data considered is 30 percent and the remaining 70 percent is the test data. The data split was stratified, except lower casing and removal of non-ASCII characters no other pre-processing technique has been applied.

Model	Hyper Parameters	Value	Description
Distil Bert	Learning Rate	0.00003,0.0004	Step Size for each iteration.
Distil Bert	Number of epochs	3,5	Number of times model will be run on entire data set.
Distil Bert	batch size	64,128	Number of sentences to be fed into training model
Distil Bert	Optimizer	SGD, Adam, AdaGrad	Optimizer to find best learning rate for the model
Distil Bert	Evaluation Metric	Accuracy, F1-Score, precision, recall	Model evaluation metrics after each epoch
Distil Bert	Loss	SparseCategoricalCrossentropy,binary_crossentropy	Loss function for binary classification
AL Bert	Learning Rate	0.00003,0.0004	Step Size for each iteration.
AL Bert	Number of epochs	3,5	Number of times model will be run on entire data set.
AL Bert	batch size	64,128	Number of sentences to be fed into training model
AL Bert	Optimizer	SGD, Adam, AdaGrad	Optimizer to find best learning rate for the model
AL Bert	Evaluation Metric	Accuracy, F1-Score, precision, recall	Model evaluation metrics after each epoch
AL Bert	Loss	SparseCategoricalCrossentropy,binary_crossentropy	Loss function for binary classification

TABLE 4-10 TRANSFORMER MODEL HYPER PARAMETERS

4.6 Resources Used

4.6.1 Hardware resources

For this study Google Colab, Nimble Box and Kaggle Kernels were used extensively. Google drive was also used as a storage for both code and data.

Google Colab Configuration:

- Intel(R) Xeon(R) CPU @ 2.20GHz
- 13 GB DDR RAM
- Nvidia Tesla K80 GPU

Kaggle Configuration:

- Intel(R) Xeon(R) CPU @ 2.20GHz
- 16 GB DDR RAM
- Nvidia Tesla K80

Nimble box Configuration:

- NVIDIA Tesla K80 GPU
- 15GB RAM
- core CPU

4.6.2 Software Resources:

Python was used as a programming language to build language models used in this study. NLTK and genism was primarily used for all NLP related processing. imblearn was used for under sampling and over sampling the imbalanced data. Scikit-Learn, LightGBM and CatBoost was used for machine learning modeling. Keras library was used for building GRU model also was used many other pre-processing tasks. Tensorflow and ktrain libraries were used in building transformer-based models.

Name of Library	Version
Python	3.7.3
Pandas	1.1.3
NumPy	1.16.1
Scikit learn	0.23.2
NLTK	3.5
TensorFlow	2.2.0
Keras	2.4.3
Matplotlib	3.0.3
Seaborn	0.9.0
Wordcloud	1.8.0
Transformers	3.3.1
CatBoost	0.24.4
imblearn	0
LightGBM	3.0.0
Ktrain	0.23.0
Scipy	1.4.1

TABLE 4-11 PACKAGES USED AND VERSION

4.7 Summary

This study we primarily focused on language modelling techniques which are both accurate and computationally efficient. We started off initially with exploratory data analysis which provided deeper insights of the dataset. The insights obtained helped us in addressing data related issues, we cleaned the data to remove stop words, NON-ASCII characters and lower casing. EDA also showed data being imbalanced this insight made us look at various balancing techniques. Choice of pre-trained embeddings was also influenced by computational factor which made us choose GloVe and Word2Vec over others. TFIDF was chosen over count vectorizer, bag of words or any other non-pretrained embedding as the accuracy obtained by other methods were way to lower less than 50 percent.

Machine learning modelling choice we choose Logistic Regression over other techniques like Naïve Bayes or KNN as logistic regression was able to provide better results compared to others. Also, model convergence was quicker compared to others.

In the Tree or boosting models reason why LightGBM and CatBoost was chosen over other popular methods like Random Forest or XGBoost was the speed execution also performance. Both

LightGBM and CatBoost have early stopping which stops once the model see no further improvement.

Ensemble technique we combine the output of 3 machine learning models to evaluate if they are better than the base models.

Deep Learning approach with a simple architecture was considered knowing that it requires higher computational power. GRU was chosen over LSTM as it is smaller architecture also provides results as good as LSTMs. We have kept the architecture of the model very simple with just 3 Bi-Directional layers.

Transformer approach where we utilize the trained model and fine tune it with our dataset has been adapted in many language models recently. As our aim is to build a language model which is computationally efficient, we chose smaller models like Distil-Bert and AL-Bert. In case of transfer learning the train-test data split adapted is 50:50 as the transformer model is built on huge corpus natural language understanding of this model is better compared to other techniques considered in this study. As we are stratifying the result wherein the ratio of labels would remain the same, we would still be able to achieve good results even with small training set.

Environment used in this study was mostly cloud based environments like Colab, Kaggle or Nimble-Box. Machine learning models were evaluated with CPU resources. Tree based models both LightGBM and CatBoost had the ability to run both in GPU and CPU, in this study we ran CPU as opposed to GPU. GRU and transformer models were run in GPU mode. This study was performed using python as the programming language and only open-source packages were used.

5. Result Evaluation

5.1 Introduction

In this section the result obtained from different language models will captured and discussed. As this is a comparative study every modelling technique has a set of study specific hyper parameters. We will capture the study specific hyper parameters, modelling tuning specific hyperparameters will not be captured here. Result obtained from the best hyper parameters will be displayed.

5.2 Machine Learning Model Evaluation

In this section we will discuss about the machine learning model performance both in terms of accuracy and computational efficiency.

Study Specific Hyper parameters used:

- pre-processing: Lemming, Stemming and N (indicating Nonem n).
- Embedding: TFIDF, GloVe 300 and Word2Vec.
- Sampling: Stratify, Under sampling, Over Sampling.

5.2.1 Naïve Bayes

Naïve Bayes model was trained with 4 study specific parameters. We experimented with 27 different combinations for naïve bayes, appendix C has the results captured. In this section we will look at top 3 results in terms of F1 score and accuracy.

Below table we have the train accuracy, F1 score, embedding time and Training Time captured. Embedding time includes the time taken to build the embedding matrix also the pre-processing time.

Model	Accuracy	F1 Score	Embedding Time	Training Time
N-word2vec-stratified	0.681	0.674	389.964	24.511
N-word2vec-underSample	0.686	0.674	392.438	24.615
N-word2vec-overSample	0.681	0.674	389.176	24.533

FIGURE 5-1 NAIVE BAYES TOP 3 MODEL TRAINING METRICS

Below table we have the test accuracy and F1 score.

Model	Accuracy	F1 Score
N-word2vec-stratified	0.682	0.658
N-word2vec-underSample	0.680	0.656
N-word2vec-overSample	0.679	0.656

FIGURE 5-2 NAIVE BAYES TOP 3 MODEL TESTING METRICS

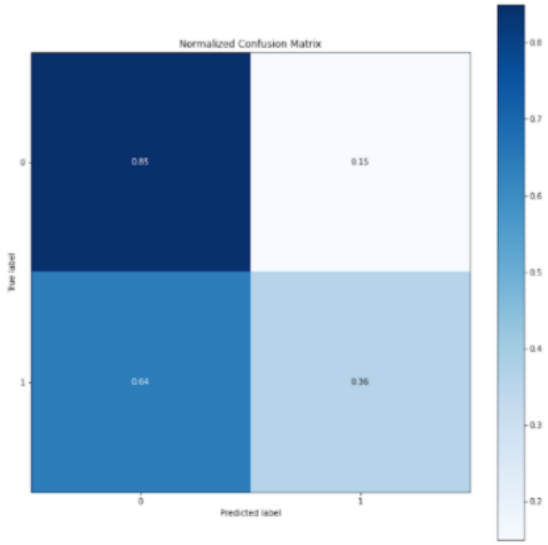


FIGURE 5-3 NAIVE BAYES CONFUSION METRICS WORD2VEC STRATIFIED

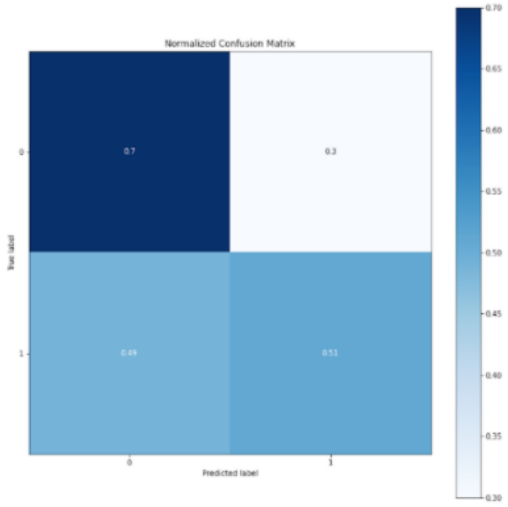


FIGURE 5-4 NAIVE BAYES CONFUSION METRICS WORD2VEC UNDER SAMPLE

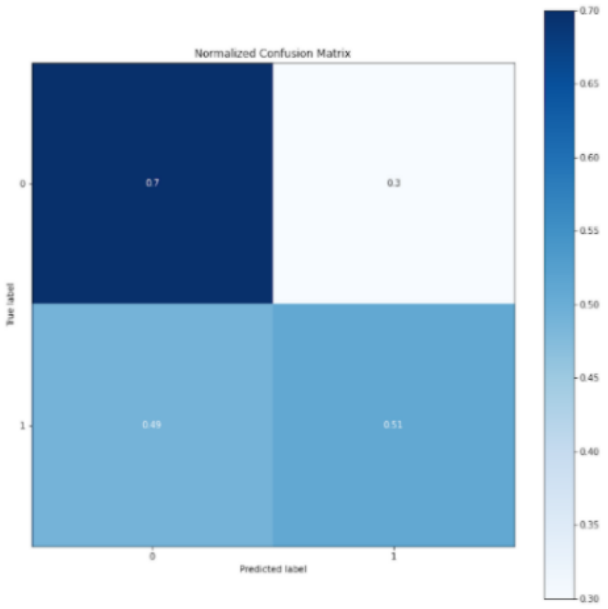


FIGURE 5-5 NAIVE BAYES CONFUSION METRICS WORD2VEC OVER SAMPLE

Observations:

- Both the test and train score look similar which indicates model built is a stable one.
- Confusion matrix also suggests the stability of the model even if the accuracy is not high model seems to be stable.
- Naïve Bayes is the fastest model considered in this study.
- Word2Vec embedding performs the little better than others.

5.2.2 Logistic Regression

Logistic regression model was trained on 4 study specific parameters. As the data volume was close to 400 thousand records solver SAG was used so that the model converges faster. Also, max_iter was set 4000 so that the model can converge quickly. We experimented on 27 different combinations for logistic regression, result of which can be found in appendix. In this section we will look at top 3 models in terms of F-1 Score. Metrics like precision, recall, auc_roc score has been captured which can be found in the appendix section.

Below table we have the train accuracy, F1 score, embedding time and Training Time captured. Embedding time includes the time taken to build the embedding matrix also the pre-processing time.

Model	Accuracy	F1 Score	Embedding Time	Training Time
lem-tfidf-underSample	0.685	0.660	50.647	89.220
tfidf-overSample	0.696	0.667	0.696	21.493
tfidf-stratified	0.695	0.676	0.696	21.470

TABLE 5-1 LOGISTIC REGRESSION TOP 3 MODELS TRAINING METRICS

Below table we have the test accuracy and F1 score.

Model	Accuracy	F1 Score
lem-tfidf-underSample	0.682	0.658
N-tfidf-overSample	0.680	0.656
N-tfidf-stratified	0.679	0.656

TABLE 5-2 LOGISTIC REGRESSION TOP 3 MODELS TESTING METRICS

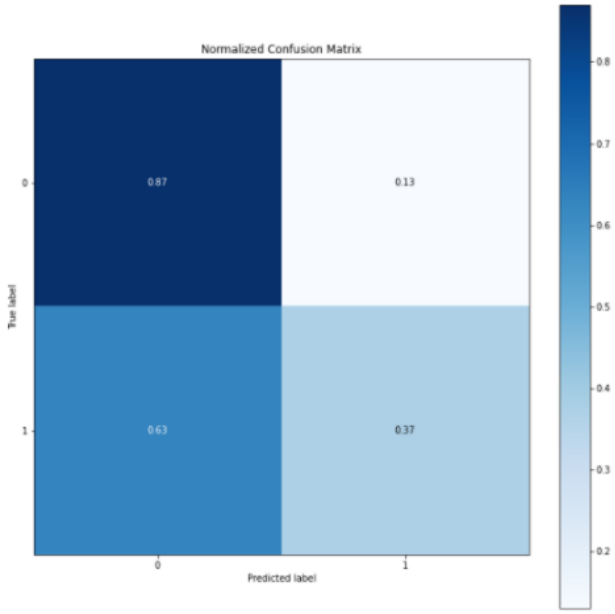


FIGURE 5-6 LOGISTIC REGRESSION CONFUSION MATRIX LEM TFIDF UNDER SAMPLE

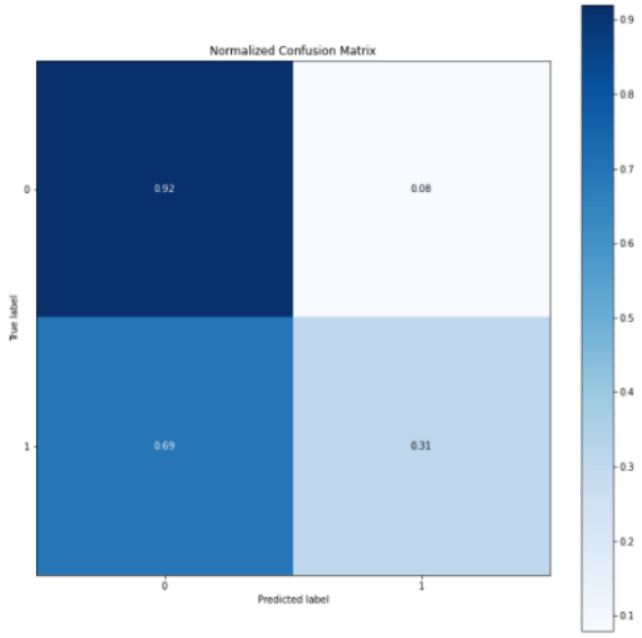


FIGURE 5-7 LOGISTIC REGRESSION CONFUSION MATRIX TFIDF OVER SAMPLE

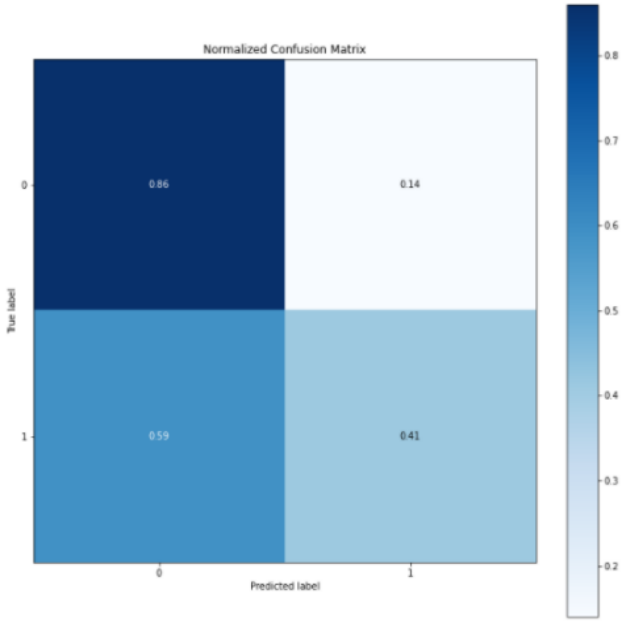


FIGURE 5-8 LOGISTIC REGRESSION CONFUSION MATRIX TFIDF STRATIFIED

Observations:

- Both the test and train score look similar which indicates model built is a stable one.

- Confusion matrix also suggests the stability of the model even if the accuracy is not high model seems to be stable.
- Lemming seems to perform better than stemming.

5.2.3 LightGBM

LightGBM is tree based boosting model which is trained on 4 study specific parameters. Hyper parameter tuning of the model was tried with various parameters mentioned in chapter 4. Considering this study is to find the optimal model we choose parameters number of estimators to be 500 and other tuning parameters we went with default parameters. Focus was ore on study specific parameters. In LightGBM also we built 27 language models details of which can be found in appendix. Here we shall discuss about top 3 models.

Below table has Accuracy, F1-Score, embedding time and Training time captured.

Model	Accuracy	F1 Score	Embedding Time	Training Time
N-glove300-stratified	0.802	0.794	0.755	386.920
N-glove300-underSample	0.803	0.794	0.754	388.499
N-glove300-overSample	0.803	0.794	0.755	387.241

TABLE 5-3 LIGHT GBM TOP 3 MODELS TRAINING METRICS

Below table has the test metrics captured.

Model	Accuracy	F1 Score
N-glove300-stratified	0.755	0.746
N-glove300-underSample	0.755	0.746
N-glove300-overSample	0.756	0.746

TABLE 5-4 LIGHT GM TOP 3 MODELS TESTING METRICS

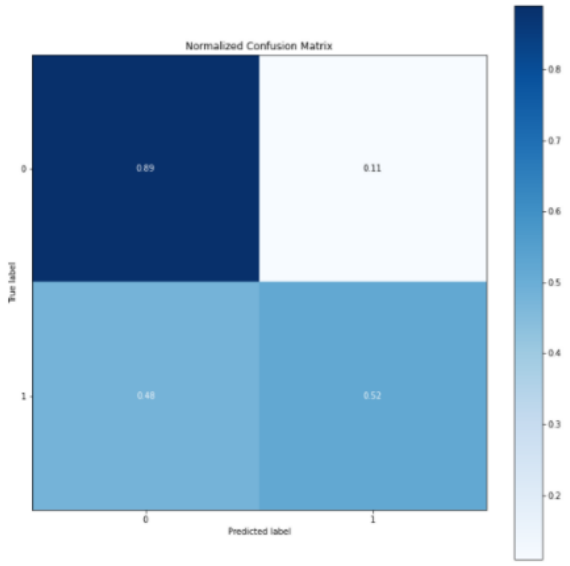


FIGURE 5-9 LIGHT GBM CONFUSION MATRIX GLOVE300 STRATIFIED

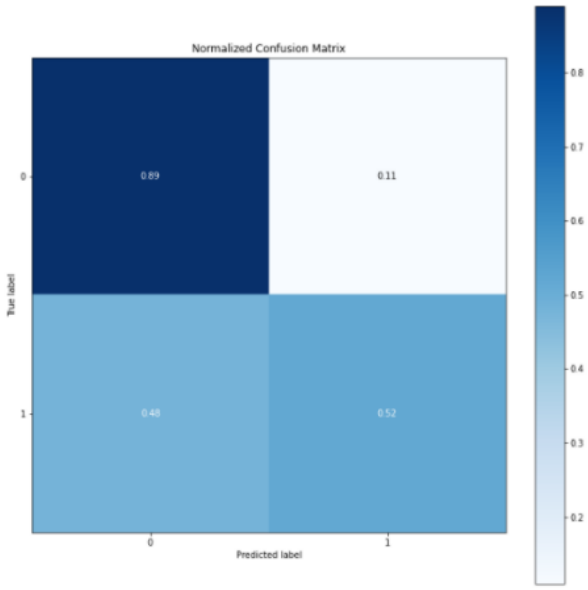


FIGURE 5-10 LIGHT GBM CONFUSION MATRIX GLOVE 300 UNDER SAMPLE

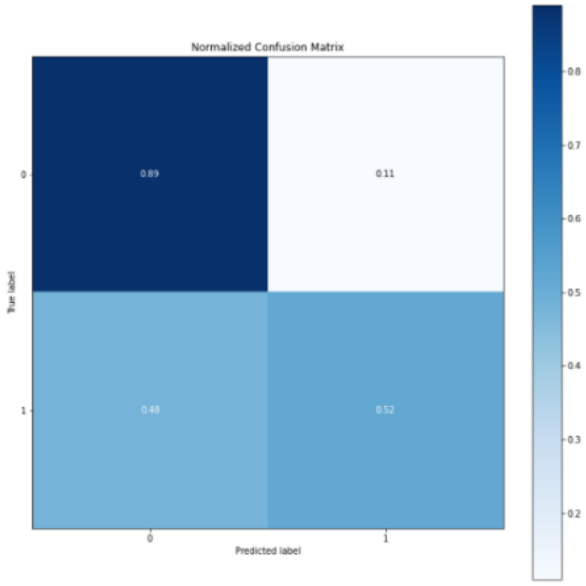


FIGURE 5-11 LIGHT GBM CONFUSION MATRIX GLOVE OVER SAMPLE

Observations:

- Train-Test variance is 4 percent for F1-Score nearly for all the combinations. Also, for the other metrics like accuracy, precision, recall and auc_roc score variance is uniform.
- GloVe 300 provided the best results compared to other embeddings.
- TFIDF was more stable variance between train and test is less than 1 percent, details of which can be found in appendix.

5.2.4 Cat Boost

Cat Boost model was trained on 4 different study specific parameters. Cat Boost is known to be model which does not require much parameter tuning expect number of iterations which we set to 50 rest we left the default parameters as is. As in Logistic Regression and LightGBM here also we built 27 models.

Below table we have Accuracy, F1-Score, embedding time and training time captured.

Model	Accuracy	F1 Score	Embedding Time	Training Time
N-tfidf-underSample	0.726	0.712	0.725	21.507
N-tfidf-stratified	0.727	0.714	0.725	21.866
N-tfidf-overSample	0.728	0.715	0.724	21.770

TABLE 5-5 CAT BOOST TOP 3 MODELS TRAINING METRICS

Below table has the Accuracy and F1 Score captured.

Model	Accuracy	F1 Score
N-tfidf-underSample	0.725	0.710
N-tfidf-stratified	0.724	0.711
N-tfidf-overSample	0.725	0.712

TABLE 5-6 CAT BOOST TOP 3 MODELS TESTING METRICS

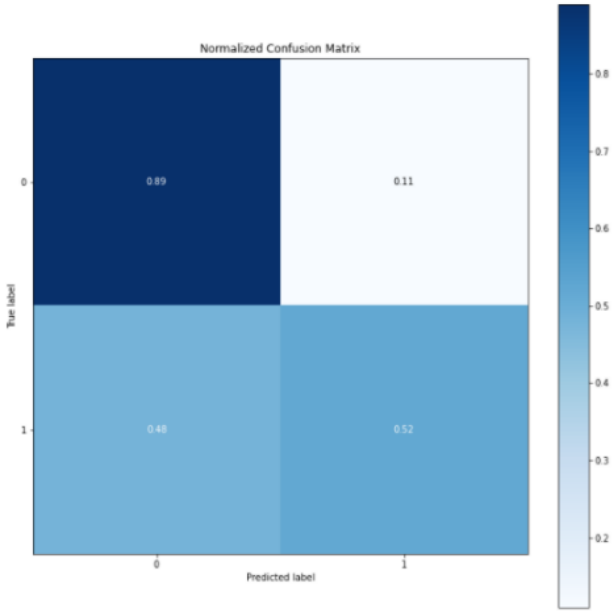


FIGURE 5-12 CAT BOOST CONFUSION MATRIX TFIDF UNDER SAMPLE

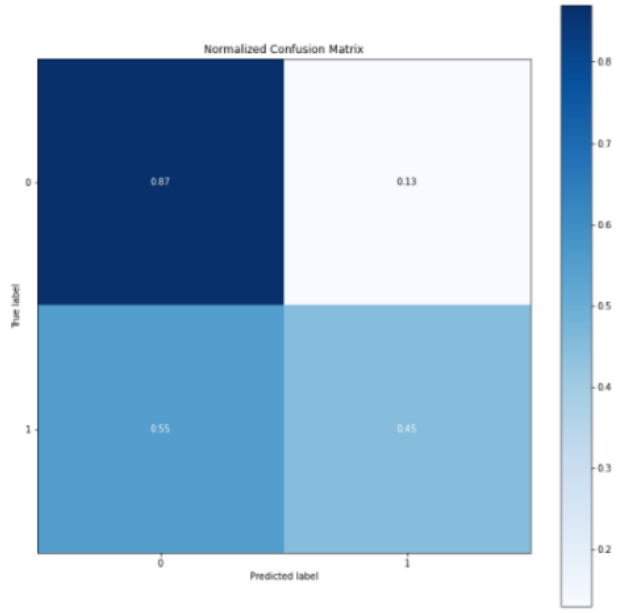


FIGURE 5-13 CAT BOOST CONFUSION MATRIX STRATIFIED STRATIFIED

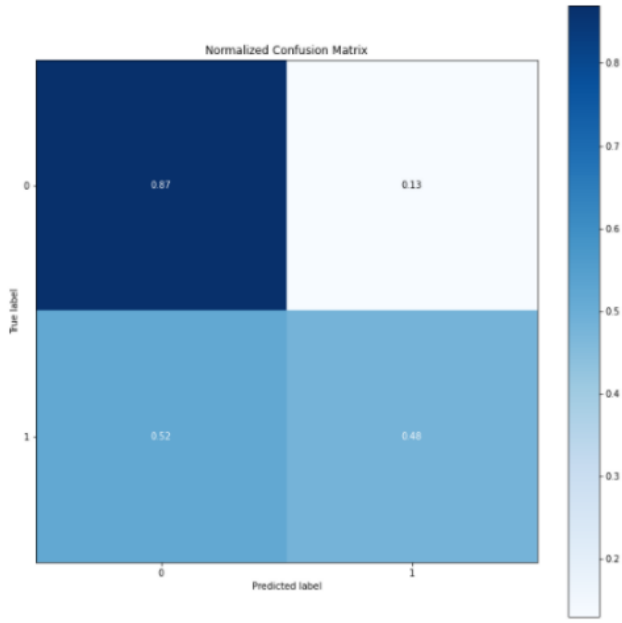


FIGURE 5-14 CAT BOOST CONFUSION MATRIX TFIDF OVER SAMPLE

Observations:

- Model seems to be stable difference between the test and train scores across all metrics is less than 0.1 percent, this even true for precision, recall and auc_roc score.
- For all the different combinations we have tried, all 5 metrics seems to have difference less than 0.1 percent.
- Cat Boost model seems to be the most stable language model across all the models considered in this study.

5.2.5 Deep Learning Model

Deep learning language model has been the most effective language modelling technique over the years. We will explore TFIDF, Word2vec and Glove. Here we are evaluating 3 language models all are having 30:70 test: train split.

Below table has loss, accuracy, F1-Score, time taken for each epoch and overall time. Number of epochs run was 3 for this study. We did not see any improvement in terms of performance by increasing the number of epochs or altering the batch size.

Model without dropout for both GloVe and Word2Vec in training gave 90+ percent accuracy whereas the test accuracy was less than 65 percent, after added dropout layer we were able to reduce the overfit. However, with SGD optimizer we saw less overfit overall ADAM optimizer provided a stable model. With optimizer ADGRAD we saw model overfitting, so went with ADAM as the optimizer.

Below table has the training metrics captured.

Embedding	Loss	Accuracy	F1 Score	Time for each epoch	Overall time
TFIDF	0.299	0.675	0.58	572 seconds	5849.89
Word2Vec	0.4525	0.7876	0.6553	219 seconds	2431.88
GloVe	0.3512	0.798	0.7038	224 seconds	2589.63

TABLE 5-7 DEEP LEARNING TRAINING METRICS

Below table has the testing metrics captured.

pre-process	accuracy	F1 Score
TFIDF	0.662	0.592
Word2Vec	0.7413	0.6717
GloVe	0.7589	0.6807

TABLE 5-8 DEEP LEARNING TEST METRICS

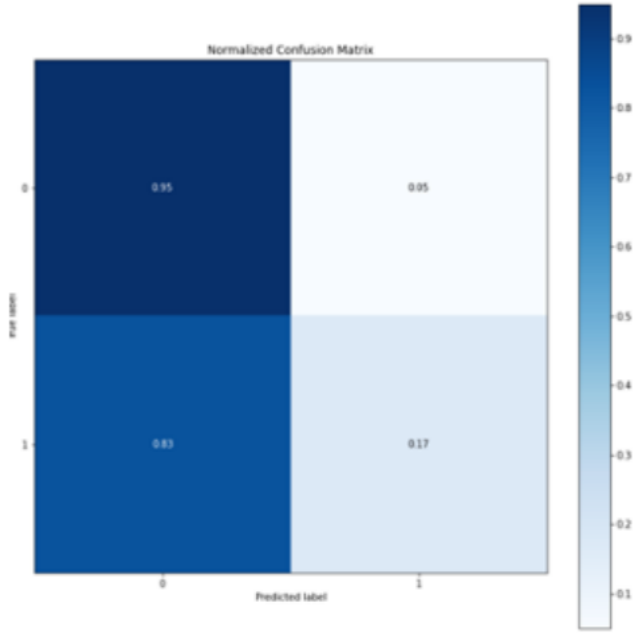


FIGURE 5-15 CONFUSION MATRIX TFIDF GRU

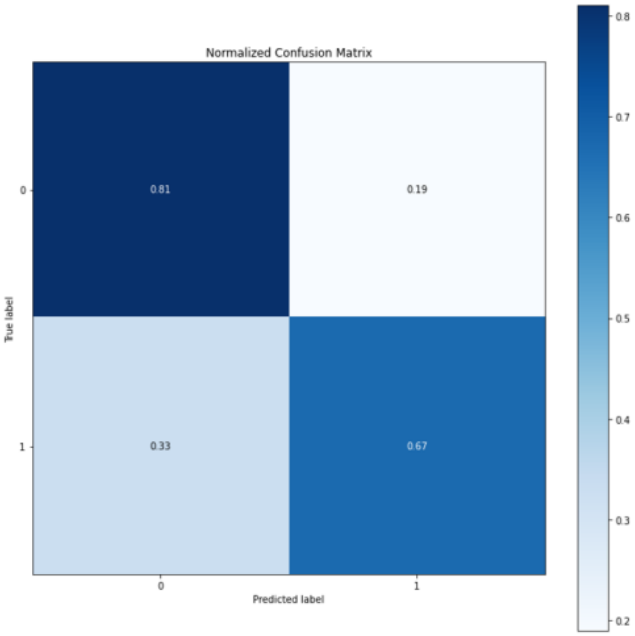


FIGURE 5-16 CONFUSION MATRIX WORD2VEC GRU

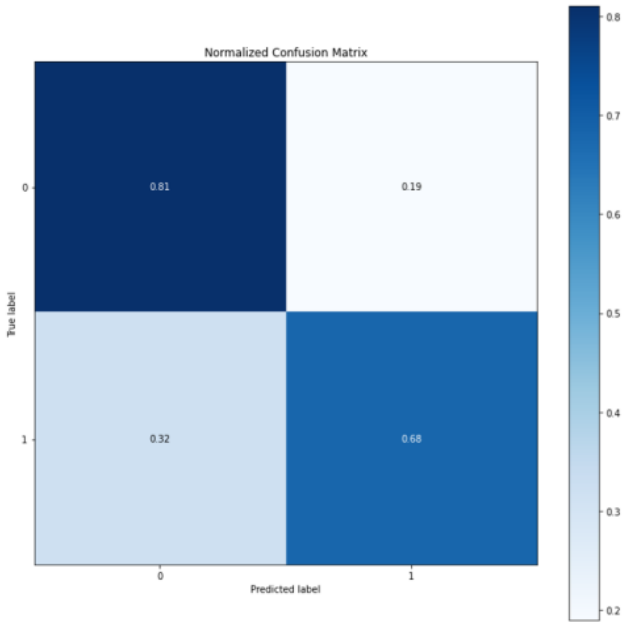


FIGURE 5-17 CONFUSION MATRIX GLOVE GRU

Outcome:

- Regularization bought the train performance down but result in model being stable. Without drop out tuning test accuracy was less than 55 percent.
- No balancing mechanism was applied still the model was able to overcome class imbalance.

5.2.6 Ensemble techniques

We have used a stacking model and at a time 3 machine learning models were part of an ensemble. Stacked output of 3 models were given to all LightGBM classifier in this study.

Below table has metrics captured for 2 ensemble approaches we took up.

- Best of 3 models: In this we chose the best 3 machine learning models used in this study. Model considered: (glove300- over Sample -Light) +(tfidf-over sample-Cat) +(tfidf-stratified-Lr)
- Overall best 3 models: In this we considered the best three models overall.
  Model considered: (glove300-overSample-Light) +(glove300-underSample-Light) +(glove300-stratified-Light)

Model	Accuracy	F1 Score	embedding time	Training Time
Best 3 models of each type	0.764	0.747	298.971	498.318
Overall best 3 models	0.823	0.812	779.558	989.764

TABLE 5-9 ENSEMBLE BEST 3 MODELS OF EACH TYPE TRAIN METRICS

Model	Accuracy	F1 Score
Best 3 models of each type	0.741	0.723
Overall best 3 models	0.786	0.771

TABLE 5-10 ENSEMBLE BEST 3 MODELS OF EACH TYPE TEST METRICS

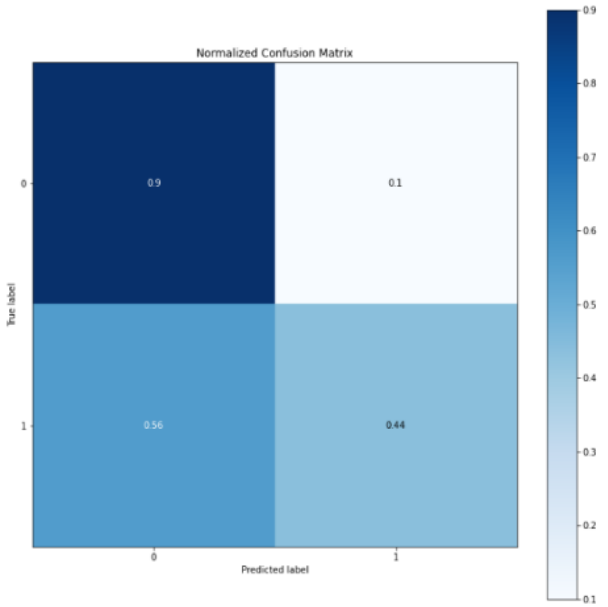


FIGURE 5-18 CONFUSION MATRIX TOP 3 MODELS OF EACH TYPE

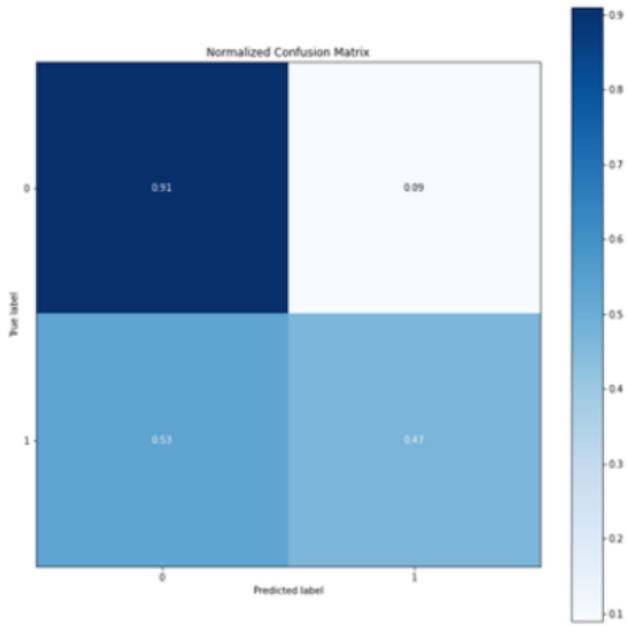


FIGURE 5-19 CONFUSION MATRIX OVERALL TOP 3 MODELS

Observations:

- Best 3 models of each type: This model underperformed 7 LightGBM models, whereas combining 3 best models all of whom were light GBM over performed the existing machine learning models.
- Performance benefit is not huge, but the time and resource consumed is significant.
- There is a slight improvement in the train-test difference which indicates model is more stable.
- Parallel processing would benefit immensely in reducing the overall time.

5.2.7 Transformer Model Evaluation

Transformer models DistilBert and ALBert are considered part of this study. Transformers models for NLP tasks has been the new state of art. To conduct this study, we have used Keras Tensor-flow package.

This study uses learning rate of 0.0003, number of epochs used was 3, batch size was 64 and used Adam as the optimizer. Evaluation is done using Accuracy, F1-Score, Precision and recall.

Below table has the training matrix.

Model	Loss	Accuracy	F1 Score	Training Time
Distil Bert	0.208	0.891	0.752	36900.876
Al Bert	0.481	0.911	0.750	52212.878

Below table has the test matrix.

Model	Accuracy	F1 Score
Distil Bert	0.916	0.761
Al Bert	0.903	0.738

Observations:

- Transformer models have outperformed all the models in this study in terms of accuracy.

- Distil Bert seems bit more stable than AL Bert instead of having lesser trainable parameters.
- Precision and F1 score was lower less than 75 percent, It has been mentioned by Quora that there may be mistake in some of the labels this may be a probable or the fine tuning required to be trained more.

5.3 Model Comparison

Best performing model in terms of overall performance is considered here. As the main aim of this study is to find an optimal performing model, we would like to consider a trade-off between model performance and training performance. Ensemble technique, GRU and machine learning algorithms, we have considered 30:70 test-train split. Transformers models the split considered is 50:50 test-train split.

Machine learning models are run in CPU mode or on laptop. GRU or Transformer model has been run on GPU.

	Train		Test		
Model	**Accuracy**	**F1 Score**	**Accuracy**	**F1 Score**	**Training Time**
Naïve Bayes	0.681	0.674	0.682	0.658	413.709
Logistic Regression	0.695	0.676	0.679	0.656	**139.87**
LightGBM	0.803	0.794	0.756	0.746	387.996
CatBoost	0.728	0.715	0.708	0.687	22.494
Ensemble	0.823	0.812	0.786	0.771	989.322
GRU	0.798	0.7038	0.7589	0.6807	2318
DistilBert	0.891	0.752	0.916	0.761	36900.876
ALBert	0.911	0.750	0.903	0.738	52212.878

As mentioned, both model accuracy and model training time is important considerations. We will discuss every approach mentioned accordingly.

Naïve Bayes:

The model converges faster than any other model considered in this study.

Model is less stable compared to other machine learning models considered.

Logistic Regression:

This is one of the fastest model, converges faster and also stable.

Model seems to be stable across all metrics considered, but accuracy of the model is on lower side.

LightGBM:

In this study this modelling technique with glove embedding and under sampling is the most optimal model.

There is a 4 percent variance in train and test but overall is a stable model, metric data in the appendix section validates the same.

CatBoost:

The most stable model in all the models considered, all 5-evaluation metrics have less than 0.2 variance between test and train.

With less parameter tuning and training time model was able to converge and provide a stable model.

Ensemble Technique:

Base model performance has been enhanced by using this technique.

GRU:

Simple architecture was considered without class balancing, model underfitted.

Transformer:

Both the methods were trained with lesser proportion of data, instead of that in terms of model accuracy outperformed all other models.

5.4 Research Question Answers

This study was to find an optimal text similarity model which could be used in real time applications. In this section we will look at the research questions we intend to address from this comparative study.

1) Which word embedding gives better performance for the task of text similarity?

a. GLoVe, Word2Vec and TFIDF was considered in this study to build word embedding for machine and deep learning models.
b. GloVe 300 with under sampling performed gave better performance.
c. Bert embedding was used for transformer models, this approach outperformed other embedding techniques.
d. **GloVe** was the best performing pre-trained embedding in machine learning approach overall **Bert** was the better word embedding.

2) Will lighter models like Distil Bert be able to provide higher accuracy and be time efficient?
   a. Both Distil Bert and AL Bert were able to provide high accuracy, outperformed all other language models.
   b. Transformer models are computationally expensive and require more hardware resources.
   c. With very less training data higher performance can be achieved in case of less data transformer models are best suited.
   d. Intent of this study being optimal models which can work with larger datasets these models may **not** serve the purpose.
3) Will usage of ensemble approach result in better performance?
   a. **Yes**, Ensemble approach has proven to perform better than individual model.
   b. Training time might be higher than an individual model but yields better performance.
4) Can we have text similarity models with higher accuracy which can be used in real time applications?
   a. **Yes**, Models like LightGBM, CatBoost in conjunction with ensemble techniques run in parallel can be used.
   b. This study has shown how proper feature engineering can yield better results.
5) Are traditional machine learning methods with proper usage of feature engineering methodologies be able to match the performance of neural networks or transformer models?

a. **Not entirely**, As seen in this study transformer models outperformed any other method with lesser data.
b. As part of this study we saw the model performance of machine learning models definitely improved by applying feature engineering techniques, still may not be able to match the model performance

5.5 Summary

In this section we analyzed results of various language modelling techniques built as part of this study. Hyper parameters mentioned in the chapter 4 was used to tune the models, best hyper parameters were then used to build the language models. We also had study specific parameters, which were applied to these base language models already built on tuned hyper parameters. Each language model was measured primarily on accuracy and F1-score, also precision, recall, auc_roc score was captured.

We evaluated the results not only on model performance but also on time taken to build the model. Considering both model performance and training time of the model an optimal model which suffices both requirements was required. Implementation not only captured the model training time it captured entire time for the model to be built from data import-feature engineering-final model. We considered Naïve Bayes, Logistic regression, LightGBM, catBoost, simple GRU architecture, ensemble method (Logistic regression, LightGBM and catBoost) and Transformer method (Distil BERT and Al Bert) to build ur language models, There are various feature engineering techniques considered, from text pre-processing to building embedding layer, only techniques which take less time was considered and every step time was captured.

Technique adopted for modelling each architecture differed for machine learning models we used larger training set but for transformer we used lesser training data. Performing this extensive comparative study, we were able to arrive at an optimal model LightGBM with Glove 300 which has train data under sampled as the optimal model, which is both fast and accurate. We were able to answer research questions formulated to be addressed in this study.

6. Conclusion and Future Recommendation

6.1 Introduction

This chapter let us discuss on how this study addressed the objectives and aims defined in chapter 1. Will also discuss on limitations of this study and the contribution this study has made to language modelling particularly text similarity. Finally, this section discusses on future research direction.

6.2 Discussion and Conclusion

Text similarity is a language modelling task in which lot of research has been undertaken. Language model architecture or pre-trained models which are highly accurate are already available. An extensive literature survey on language modelling techniques indicated the lack of existing research on optimized model. Model which are both accurate and fast, so that it can be used in real time applications which necessitates to be computationally less expensive along with being fast and accurate. We also did not find a comprehensive study on pre-processing of data in this study we would like to explore how the text pre-processing and balancing techniques will impact the overall model performance.

In this explorative study we will focus on various feature engineering techniques also on language models which are computationally efficient and accurate. As mentioned in aims and objectives we explored various feature engineering techniques like data pre-processing, balancing of data and different embedding techniques. All these methods were considered so that they are efficient, we captured time taken by the dataset to be transformed into an embedding. Various machine learning approaches were explored again the time efficiency along with model efficiency was an important metric consideration. Language model built were evaluated using evaluation metrics like accuracy, F1-Score, Precision, Recall, roc_auc score and confusion metric plots. Also, ensemble technique was explored and to see whether it adds any performance benefit. Transformer models like Distil Bert and Al Bert were also evaluated and an extensive comparison study was done comparing all the language models.

EDA performed on the data set indicated that 31 percent of the duplicate questions were repeated. Duplicate questions have more words in common and are co-related whereas non duplicate questions common words are less and are distributed. Length of non-duplicate questions are more

than the duplicate questions. EDA also suggested the dataset is imbalanced, under sampling proved to be a better approach in the approaches considered for this study.

The study took different approach for machine learning models, deep learning models and transformers. Machine and deep learning models, we experimented both lemming and stemming to see how effective it is. Word embedding was built using TFIDF, GloVe and Word2Vec all these word embedding techniques were applied with lemming and stemming and were evaluated. Class imbalance was dealt with under sampling, over sampling and stratifying the data, again these sampling techniques were experimented with different combination of word embeddings and pre-processing techniques. We used 4 machine learning algorithms Naïve Bayes, Logistic regression, LightGBM and CatBoost and applied different combinations of all the language modelling methods mentioned. As the main aim of the study was to find an optimal model the GRU model considered was of a simple architecture.

GRU architecture considered in this study we experimented with all three embeddings considered for this study without lemming or stemming. Models initially over fitted it was regularized by introducing drop out layers thus resulting in a stable model. Although a smaller model was considered it required GPU environment to be executed. Considering a simple model performance obtained from some of the machine learning models and the GRU model was same in this study.

Transformer models which are smaller in size was considered, although smaller it again requires GPU or TPU to be fine-tuned. Even with getting trained with smaller data set as compared other language models, transformer models outperformed the other models.

In the comparative study we built various language models, for the machine learning models more weight was given on text pre-processing, addressing imbalance and embeddings. Whereas the deep learning and transformer we explored models with smaller architecture as the aim of the study was to find an optimal model. Focusing on feature engineering methods helped us achieving stable models. CatBoost and LightGBM models were very stable the difference between the test and train metrics were negligible. Ensemble approach also proved that the combining more than one model would result in a more stable model which even more accurate. Deep learning approaches like GRU or the transformers have provided state of art for many language modellings tasks. These models take more time to converge whereas the intent of this study was to do a comparative study

of optimal models. We experimented on these models by keeping number of trainable parameters as less as possible.

One of the major objectives of the study was to find whether properly feature engineered data set when applied with simple machine learning models will be able to match the results obtained from deep learning models like GRU and transformers. This study displayed that proper feature engineering would help the model converging faster also result in a stable model. Model performance may not match but certainly can be an optimal model.

6.3 Contribution to Knowledge:

As there has been no significant research done on identifying an optimal model, which is a need of the hour considering the explosion of the data over the years. This study would act as a starting point in this direction.

Most of the previously conducted research have not concentrated more on feature engineering approaches in language modelling tasks. The extensive exploratory study done on various data processing and data transformation methods which then is applied with various models provides deeper understanding of each method used. These metrics captured can serve as a bench for any researcher how intends research in this field.

Machine learning models like CatBoost and LightGBM are not extensively present in existing literature for text similarity tasks. Distil Bert and AL Bert are not found in any previous conducted research measuring of text similarity.

Time taken by each step was captured this serves as an important metric, as any further research this time serves as a benchmark and any further research can also work on speeding up each of these steps not only model training time.

6.4 Future Recommendation

Extensive study was done to find an optimal language model, however there are lot many areas where the proposed methodology can be improved:

- Given the large amount of data getting generated, Distributed computing techniques can be used in both text processing and running parallel modelling tasks.

- Usage of Spark data frame rather than Pandas data frame can improve overall time efficiency also can work on extremely large datasets.
- Sent2Vec embedding which vectorizes an entire sentence at a time can be explored. Also Sentence Bert can be explored.
- Hyper parameters for LightGBM and CatBoost can further be tuned to improve both model accuracy and time efficiency.
- More ensemble techniques can be explored as the stacking model performed better than individual models, the ensemble approach must be run in parallel to improve the time efficiency.
- Language model built in this study can be evaluated with other text similarity data sets like stack overflow duplicate questions dataset.
- AI explain ability approaches LIME and SHAP can be explored which provide deeper insights on the model built.
- Attention mechanism can be used as it has the capability to work in parallel rather than GRU's which requires to be executed sequentially.

References

Addair, T., (2017) Duplicate Question Pair Detection with Deep Learning.

Aggarwal, N., Asooja, K. and Buitelaar, P., (2012) DERI&UPM: Pushing corpus based relatedness to similarity: Shared task system description. In: **SEM 2012 - 1st Joint Conference on Lexical and Computational Semantics*.

Anon (2014) *Proceedings of the Annual Conference of the International Speech Communication Association, INTERSPEECH*. *Proceedings of the Annual Conference of the International Speech Communication Association, INTERSPEECH*.

Benard Magara, M., Ojo, S.O. and Zuva, T., (2018) A comparative analysis of text similarity measures and algorithms in research paper recommender systems. In: *2018 Conference on Information Communications Technology and Society, ICTAS 2018 - Proceedings*.

Bengio, Y., Ducharme, R., Vincent, P. and Jauvin, C., (2003) A Neural Probabilistic Language Model. In: *Journal of Machine Learning Research*.

Chandra, A. and Stefanus, R., (2020) *Experiments on Paraphrase Identification Using Quora Question Pairs Dataset*. *arXiv*.

Chorowski, J., Bahdanau, D., Serdyuk, D., Cho, K. and Bengio, Y., (2015) Attention-based models for speech recognition. In: *Advances in Neural Information Processing Systems*.

Chung, J., Gulcehre, C., Cho, K. and Bengio, Y., (2014) Gated Recurrent Neural Networks on Sequence Modeling. *arXiv*.

Collobert, R. and Weston, J., (2008) A unified architecture for natural language processing: Deep neural networks with multitask learning. In: *Proceedings of the 25th International Conference on Machine Learning*.

Deng, J., Dong, W., Socher, R., Li, L.-J., Kai Li and Li Fei-Fei, (2010) ImageNet: A large-scale hierarchical image database.

Devlin, J., Chang, M.W., Lee, K. and Toutanova, K., (2019) BERT: Pre-training of deep bidirectional transformers for language understanding. In: *NAACL HLT 2019 - 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human*

Language Technologies - Proceedings of the Conference.

Dhakal, A., Poudel, A., Pandey, S., Gaire, S. and Baral, H.P., (2018) Exploring Deep Learning in Semantic Question Matching. In: *Proceedings on 2018 IEEE 3rd International Conference on Computing, Communication and Security, ICCCS 2018*.

Farouk, M., (2019) Measuring Sentences Similarity: A Survey. *Indian Journal of Science and Technology*.

George, A., Ganesh H. B., B., Kumar M, A. and K P, S., (2018) TeamCEN at SemEval-2018 Task 1: Global Vectors Representation in Emotion Detection.

Gers, F.A., Schmidhuber, J. and Cummins, F., (2000) Learning to forget: Continual prediction with LSTM. *Neural Computation*.

He, H., Gimpel, K. and Lin, J., (2015) Multi-perspective sentence similarity modeling with convolutional neural networks. In: *Conference Proceedings - EMNLP 2015: Conference on Empirical Methods in Natural Language Processing*.

Hinton, G., Vinyals, O. and Dean, J., (2015) Distilling the Knowledge in a Neural Network. pp.1–9.

Hochreiter, S. and Schmidhuber, J., (1997) Long Short-Term Memory. *Neural Computation*.

Hopfield, J.J., (1982) Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Sciences of the United States of America*.

Howard, J. and Ruder, S., (2018) *Introducing state of the art text classification with universal language models*. 15-05.

Imtiaz, Z., Umer, M., Ahmad, M., Ullah, S., Choi, G.S. and Mehmood, A., (2020) Duplicate Questions Pair Detection Using Siamese MaLSTM. *IEEE Access*.

Jiao, X., Yin, Y., Shang, L., Jiang, X., Chen, X., Li, L., Wang, F. and Liu, Q., (2019) TinyBERT: Distilling BERT for Natural Language Understanding. pp.1–14.

Joulin, A., Grave, E., Bojanowski, P. and Mikolov, T., (2017) Bag of tricks for efficient text classification. In: *15th Conference of the European Chapter of the Association for Computational Linguistics, EACL 2017 - Proceedings of Conference*.

Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., Ye, Q. and Liu, T.Y., (2017) LightGBM: A highly efficient gradient boosting decision tree. In: *Advances in Neural Information Processing Systems*.

Lan, Z., Chen, M., Goodman, S., Gimpel, K., Sharma, P. and Soricut, R., (2019) *Albert: A lite bert for self-supervised learning of language representations*. *arXiv*.

Liu, B., Zhang, T., Niu, D., Lin, J., Lai, K. and Xu, Y., (2019) Matching Long Text Documents via Graph Convolutional Networks. *ACL*.

Liu, W., Zhang, M., Luo, Z. and Cai, Y., (2017) An Ensemble Deep Learning Method for Vehicle Type Classification on Visual Traffic Surveillance Sensors. *IEEE Access*.

Maharjan, S., Montes-Y-Gómez, M., González, F.A. and Solorio, T., (2020) A genre-aware attention model to improve the likability prediction of books. In: *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, EMNLP 2018*.

Mueller, J. and Thyagarajan, A., (2016) Siamese recurrent architectures for learning sentence similarity. In: *30th AAAI Conference on Artificial Intelligence, AAAI 2016*.

Nickel, M. and Kiela, D., (2017) Poincaré embeddings for learning hierarchical representations. In: *Advances in Neural Information Processing Systems*.

Olah, C., (2015) Understanding LSTM Networks [Blog]. *Web Page*.

Padurariu, C. and Breaban, M.E., (2019) Dealing with data imbalance in text classification. In: *Procedia Computer Science*.

Patrizio, A., (2019) *Expect 175 zettabytes of data worldwide by 2025*. NetworkWorld.

Patro, B.N., Kurmi, V.K., Kumar, S. and Namboodiri, V.P., (2018) *Learning semantic sentence embeddings using pair-wise discriminator*. *arXiv*.

Peng, Y., Yan, S. and Lu, Z., (2019) *Transfer learning in biomedical natural language processing: An evaluation of BERT and ELMo on ten benchmarking datasets*. *arXiv*.

Pennington, J., Socher, R. and Manning, C.D., (2014) GloVe: Global vectors for word representation. In: *EMNLP 2014 - 2014 Conference on Empirical Methods in Natural Language Processing, Proceedings of the Conference*.

Peters, M., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K. and Zettlemoyer, L., (2018) [ELMo] [seminal LSTM contextualised embedding model] 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)*.

Prokhorenkova, L., Gusev, G., Vorobev, A., Dorogush, A.V. and Gulin, A., (2018) Catboost: Unbiased boosting with categorical features. In: *Advances in Neural Information Processing Systems*.

Radford, A. and Salimans, T., (2018) Improving Language Understanding by Generative Pre-Training (transformer in real world). *OpenAI*.

Reimers, N. and Gurevych, I., (2020) Sentence-BERT: Sentence embeddings using siamese BERT-networks. In: *EMNLP-IJCNLP 2019 - 2019 Conference on Empirical Methods in Natural Language Processing and 9th International Joint Conference on Natural Language Processing, Proceedings of the Conference*.

Sanh, V., Debut, L., Chaumond, J. and Wolf, T., (2019) DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. pp.2–6.

Shahmirzadi, O., Lugowski, A. and Younge, K., (2019) Text similarity in vector space models: A comparative study. In: *Proceedings - 18th IEEE International Conference on Machine Learning and Applications, ICMLA 2019*.

Sharma, L., Graesser, L., Nangia, N. and Evci, U., (2019) Natural Language Understanding with the Quora Question Pairs Dataset. *arXiv preprint arXiv:1907.01041*.

Shaw, P., Uszkoreit, J. and Vaswani, A., (2018) Self-attention with relative position representations. In: *NAACL HLT 2018 - 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies - Proceedings of the Conference*.

Tian, J., Zhou, Z., Lan, M. and Wu, Y., (2018) ECNU at SemEval-2017 Task 1: Leverage Kernel-based Traditional NLP features and Neural Networks to Build a Universal Model for Multilingual and Cross-lingual Semantic Textual Similarity.

Wan, J., Wang, D., Hoi, S.C.H., Wu, P., van Merrienboer, B., Bahdanau, D., Dumoulin, V.,

Serdyuk, D., Warde-farley, D., Chorowski, J., Bengio, Y., Szegedy, C., Reed, S., Sermanet, P., Vanhoucke, V., Rabinovich, A., Pan, J., Ng, C.J., Teoh, A.J., Lipton, Z.C., Konda, K., Bouthillier, X., Vincent, P., Memisevic, R., Juergen Schmidhuber, Hinton, G.E., Srivastava, N., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.R., Hang Su, H.C., Gong, Y., Ke, Q., Isard, M., Lazebnik, S., Denton, E., Chintala, S., Szlam, A., Fergus, R., Dauphin, Y.N., Bengio, Y. and Daniel Povey, X.Z.& S.K., (2015) A Critical Review of Recurrent Neural Networks for Sequence Learning arXiv : 1506 . 00019v2 [cs . LG] 29 Jun 2015. *International Journal of Computer Vision*.

Wang, L., Zhang, L. and Jiang, J., (2020) Duplicate Question Detection with Deep Learning in Stack Overflow. *IEEE Access*.

Wang, Z., Hamza, W. and Florian, R., (2017) Bilateral multi-perspective matching for natural language sentences. In: *IJCAI International Joint Conference on Artificial Intelligence*.

Wang, Z., Mi, H. and Ittycheriah, A., (2016) Sentence similarity learning by lexical decomposition and composition. In: *COLING 2016 - 26th International Conference on Computational Linguistics, Proceedings of COLING 2016: Technical Papers*.

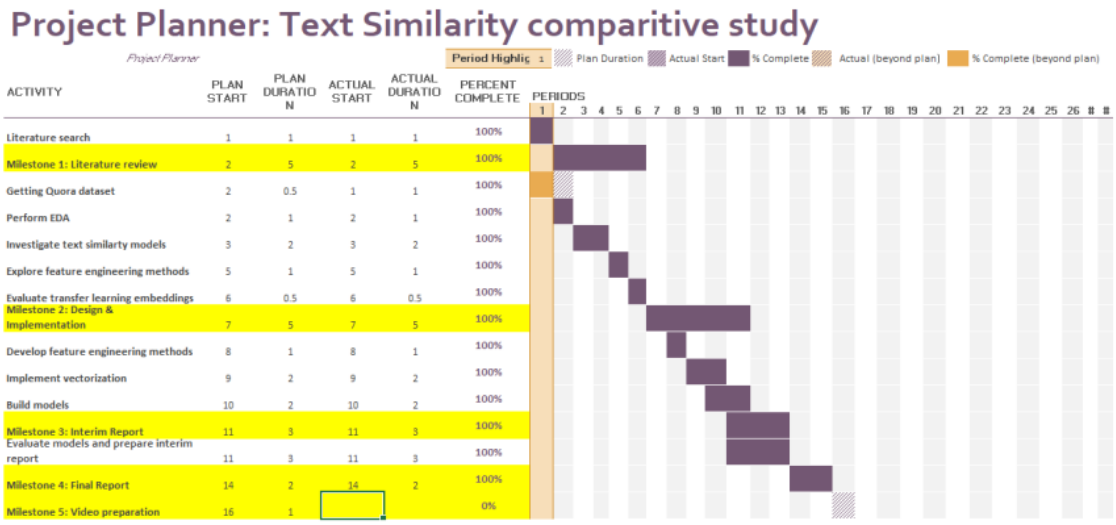
Werbos, P.J., (1990) Backpropagation Through Time: What It Does and How to Do It. *Proceedings of the IEEE*.

Yang, Z., Dai, Z., Yang, Y., Carbonell, J., Salakhutdinov, R. and Le, Q. V., (2019) *XLNet: Generalized autoregressive pretraining for language understanding*. *arXiv*.

Yao, L., Pan, Z. and Ning, H., (2019) Unlabeled Short Text Similarity with LSTM Encoder. *IEEE Access*.

Zhou, H., Young, T., Huang, M., Zhao, H., Xu, J. and Zhu, X., (2018) Commonsense knowledge aware conversation generation with graph attention. In: *IJCAI International Joint Conference on Artificial Intelligence*.

Appendix A


Project Planner: Text Similarity comparitive study
Project Planner
Period Highlig 1
Plan Duration
Actual Start
% Complete
Actual (beyond plan)
% Complete (beyond plan)
ACTIVITY
PLAN START
PLAN DURATION
ACTUAL START
ACTUAL DURATION
PERCENT COMPLETE
PERIODS
Literature search 1 1 1 1 100%
Milestone 1: Literature review 2 5 2 5 100%
Getting Quora dataset 2 0.5 1 1 100%
Perform EDA 2 1 2 1 100%
Investigate text similarty models 3 2 3 2 100%
Explore feature engineering methods 5 1 5 1 100%
Evaluate transfer learning embeddings 6 0.5 6 0.5 100%
Milestone 2: Design & Implementation 7 5 7 5 100%
Develop feature engineering methods 8 1 8 1 100%
Implement vectorization 9 2 9 2 100%
Build models 10 2 10 2 100%
Milestone 3: Interim Report 11 3 11 3 100%
Evaluate models and prepare interim report 11 3 11 3 100%
Milestone 4: Final Report 14 2 14 2 100%
Milestone 5: Video preparation 16 1 0%

Appendix B

COMPARATIVE STUDY OF DIFFERENT PREDICTIVE MODELS FOR THE TASK OF TEXT SIMILARITY

Vasist G N

Student ID: **931122**

Under the supervision of

SUVAJIT MUKHOPADHYAY

Research Proposal

Liverpool John Moores University – Master's in Data Science

Abstract

Text similarity is very important consideration in designing systems like design of contextual chatbots, duplicate question detection and machine translation. One major challenge in real world applications is tradeoff between computational cost and accuracy while designing a text similarity model. For instance, over 100 million visit Quora month and more than 14 million questions have been asked so far. There is a need to have a system in place which is both accurate and time efficient. This will help systems like Quora and Stack-overflow to identify duplicate questions and end users will be benefitted as they would find answers to their questions rather than waiting for someone to answer.

Complex neural network models have proven their advantage over classical models however the training time required, and computational cost associated with them is very high. Explosion of data in recent times have necessitated need of an optimal time efficient text similarity model. Transfer Learning has bought about a revolutionary change in natural language processing paradigm, but these models again incur huge computational cost and are less time efficient (Reimers and Gurevych, 2020).

We will evaluate various neural networks models with transfer learning which are simple with lesser trainable parameters also classical shallow learning models with different combination of text pre-processing, text processing and feature engineering along with ensemble approaches.

Keywords: Quora, Stack-overflow, Neural Networks, transfer learning, shallow learning and feature engineering

Table of Contents

List of Figures

List of Tables

List of Abbreviations

Abbreviation	Expansion
LSTM	Long short-term memory
NLP	Natural language processing
GRU	Gated Recurrent Units
ULMFit	Universal Language Model Fine-tuning for Text Classification
SGD	Stochastic gradient descent
NLTK	Natural Language Toolkit
POS	Parts of Speech
NER	Named Entity Recognition
RNN	Recurrent neural network
BERT	Bidirectional Encoder Representations from Transformers
TF-IDF	Term Frequency – Inverse Document Frequency
AUC	Area under curve
CNN	Convolutional Neural Network
AWD-LSTMs	Average SGD Weight Dropped LSTMs

Background

Natural language processing (NLP) is evolving at an unprecedented pace. NLP is a field which focusses on how a digital computer can interpret and understand natural language spoken by humans. To understand or to interpret human language doesn't only deal with understanding words also the context of its usage. In this section we will discuss on how text similarity approaches evolved and the available approaches. Broadly we can categorize text similarity approaches into String Based, Corpus Based and Knowledge based.

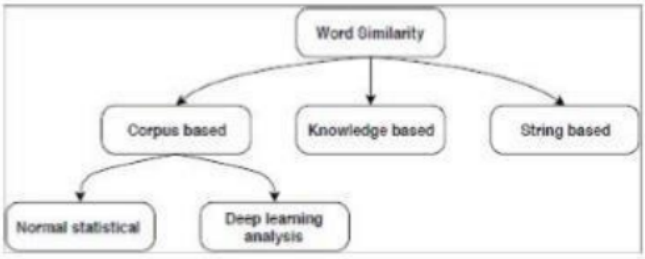


FIGURE 1 DIFFERENT APPROACHES FOR CALCULATING WORD-TO-WORD SIMILARITY.
(Farouk, 2019)

String Based approaches which can further be categorized character based and term based was prevalent before once the machine learning algorithms usage came into use, we saw algorithms like logistic regression, Naïve Bayes and tree models being used for determining text similarity. Lately Corpus based approaches are more prevalent however we also see Knowledge based approaches gaining prominence. Various corpus-based text pre-processing like count vectorizer, TFIDF, bag of words and other techniques which used to vectorize the text and later these vectors were fed as input to machine learning models and various neural network models.

Deep learning models like RNN started outperforming [Natural Language Processing (Almost) from Scratch] most existing state of art approaches once these word vectors were introduced.

Further we saw improved architectures like Bi-Directional RNN's, Seq-Seq and Encoder-decoder mechanisms which enhanced the performance of text similarity models. RNNs traditionally have an issue while modeling long-range dependencies, the problem of vanishing gradients. This issue of long range dependencies and vanishing gradients was addressed with invent of LSTM's (Hochreiter and Schmidhuber, 1997). LSTM's aided with mechanism of gates and cell states was able to retain long-range dependencies. GRU's (Gated Recurrent Units) (Chung et al., 2014) a lighter version of LSTM's which performs on par with LSTM's also started gaining popularity.

Encoding long sequences into vectors still was a problem with long range sequences. Attention mechanism (Chorowski et al., 2015) wherein special attention to input tokens overcame these issues and started to give better performance later (Shaw et al., 2018) self-attention Transformer Models which works work upon Self Attention improved performance further. Self-Attention is a new type of mechanism where a given token looks around its neighboring tokens to form embedding or its meaning. The Attention mechanism with Transformers have been a revolution in the field of Natural Language modeling. modelling (Howard and Ruder, 2018) Transfer learning and language began the new era of transfer learning in natural language processing tasks, ULMFit built using state-of-art AWD-LSTMs gained prominence in various language models.

(Peters et al., 2018) introduced bi-directionality into natural language modeling tasks which bought ELMo to fore in natural language tasks. ELMO is a feature-based approach where to form word embeddings the context of the sentence is involved. It is done by making the model look from either direction (forward, backward) and concatenating the results. Bi-directional LSTMs is used to capture context from either side. This has allowed one to either use the word embeddings for language modeling or combine with other available embeddings.

(Devlin et al., 2019) BERT has proven to be an efficient state of art model for natural language processing tasks it has outperformed various state of art. For text similarity models like sentence or document similarity, it requires that both sentences or documents to be fed into the network, which causes a massive computational overhead. There is a necessity to chose language models which can process a document in a simple yet in a powerful manner.

FIGURE 2 PARAMETER COUNTS OF SEVERAL RECENTLY RELEASED PRETRAINED LANGUAGE MODELS. (Sanh et al., 2019)

We are also seeing various lighter word embedding which are equally provided state-of-art results. (Reimers and Gurevych, 2020) came up with Sentence-BERT (SBERT) which was able to produce same accuracy as BERT but was able to find the text similarity in about 5 seconds. (Sanh et al., 2019) came up with DistilBERT which is 40% of the size of BERT yet has proven to be as powerful as BERT we could also see improved time efficiency as it lighter than BERT. (Jiao et al., 2019) is as effective as BERT while being 7.5x smaller and 9.4x faster.

Problem Statement

Although we have seen a lot research being done in determining text similarity there hasn't been a comprehensive study of these approaches in terms of both accuracy and computational complexity. Text similarity can measure using fuzzy approaches, string-based approaches, traditional machine learning approaches, neural networks with different models like ANN, RNN, CNN, LSTM, etc. also we have various transfer learning models with attention and transformer mechanisms.

Some of the previously conducted research on detecting text similarity are:

- (Dhakal et al., 2018) have carried out a survey using traditional machine learning approaches and an ANN architecture using on the Quora dataset. Random forest

outperformed other machine learning models however ANN outperformed Random forest with an accuracy of 86.09%.

- (Benard Magara et al., 2018) evaluated various machine learning text similarity approaches both for accuracy and time efficiency. RPart algorithm was able to provide accuracy and time efficiency of 80.73 and 2.354628 seconds respectively on a small dataset however random forest and XGBoost took close to 40 seconds.
- (Sharma et al., 2019) conducts extensive exploration of Quora dataset and used various machine learning models, including linear, tree-based model and neural network models. CBOW Neural network model outperformed compared to other models.
- (Reimers and Gurevych, 2020) Sentence BERT is a novel approach which fine-tunes BERT. SBERT is computationally more efficient than BERT and more accurate than BERT. SBERT architecture makes it an apt for usage in sentence similarity systems which has less computational power and still will be able to give better performance.
- (Wang et al., 2020) this paper shows how pre-trained word embedding word2vec is used with different neural architectures to detect duplicate questions. word2vec with LSTM outperformed compared to other models.
- (Imtiaz et al., 2020) three different word embeddings were used with Siamese LSTM. Ensemble method of combining the outputs of three embeddings was done 91% accuracy was achieved by this ensemble approach. Ensemble approach have proven in various machine and deep learning systems to be delivering better performance compared to individual models.
- (Liu et al., 2017) used ensemble technique wherein a set of learning algorithms will be used rather than constituent learning algorithm alone. Individual language model tends to cause bias in terms of fixed set of parameters, Ensemble approach helps reducing such a bias.

Most of these studies mentioned above try to achieve better accuracy. But real-world applications need to be both time efficient and accurate. In this exploratory research we will evaluate both time efficiency and accuracy. We will explore various text pre-processing steps like stemming, lemming, removal of stop words with and without text processing steps like POS tagging, NER etc. Then we will explore feature engineering approaches like TF-IDF, Count

Vectorizer and continuous Bag of words with various machine and simple neural network models.

We shall also explore transfer learning with lighter word embeddings like Distil BERT, Tiny Bert with various machine and simple neural network models. There hasn't been research conducted on these lighter word embeddings for the task text similarity, evaluation of both accuracy and time efficiency will determined on the word embeddings. Finally, we will try out ensemble methods by combining output of one or models.

Research Question

The central research questions part of this study are listed below:

- Which word embedding gives better performance for the task of text similarity?
- Will lighter models like Distil Bert be able to provide higher accuracy and be time efficient?
- Will usage of ensemble approach result in better performance?
- Can we have text similarity models with higher accuracy which can be used in real time applications?
- Are traditional machine learning methods with proper usage of feature engineering methodologies be able to match the performance of neural networks or transformer models?
- Usage of NER or POS how effective are they in terms of performance against not using it?

Aim and Objectives

The main aim of this research to evaluate various text similarity models which can be used in real world use cases. The goal here is to find a language model which can detect similarity between texts accurately and in a time efficient manner.

The research objectives are formulated based for this study is as follows:

- To explore various feature engineering techniques and available pre-trained embeddings.

- To investigate various language models which can determine text similarity.
- To analyze the different language models considered in this study in terms of accuracy and time efficiency.
- To investigate whether ensemble approaches will be able to improve overall performance as compared to individual language models significantly.

Significance of the Study

- Explosion of data in recent times have necessitated need of an optimal time efficient text similarity model, (Patrizio, 2019) predicts 175 Zettabytes of data by 2025. This huge data necessitates the need of more computational resources to accomplish text/document similarity tasks. Training time required to build these models also will be significantly high. Simple models with less trainable parameters which can give good performance can make us well prepared to handle data explosion better.
- Language models in the era of big data should be simple yet powerful, it requires to be both accurate and take less to train and predict. Real time applications need the result to be returned in couples of seconds/minutes. Organizations which offer text similarity solutions require more computational resources to provide seamless service to users as more computational resources are required this in turn results more cost to avail these services. If there is a model which can perform the task with less resources and still be accurate, we can see more real time applications in this domain and overall experience of using web search/ plagiarism check applications would be better.
- This comparative study becomes very significant to address the need of the optimal language models.

Scope of the Study

The scope of this comparative study will be limited to the below items:

- Exploring data pre-processing and feature engineering techniques for the task of text similarity.
- Building language model with simple architecture using both transfer learning and traditional corpus-based methods.

- Models will be tuned for better performance by evaluating with different tunable parameters.
- Evaluating ensemble approach with various combination of models. Each ensemble model will have same feature engineering methodology but will have different combination of learning models.

The following will not be included as a part of the study:

- More than on embedding technique in an ensemble.
- Only Quora duplicate questions data set will be considered for this study no other dataset like this will be evaluated.
- No new method or model will be proposed from this study.

Research Methodology

Introduction

This exploratory comparative study of different predictive models using natural language processing paradigm will evaluate various language models right from pre-processing to the end model which can predict the text similarity for a given input. Every language model will be evaluated for its accuracy and the time efficiency.

The key process involved to determine text similarity is depicted in the workflow below:

- Understanding the dataset.
- Selecting part of data as a subset if the data volume is high.
- Pre-processing the data.
- Balancing data if required.
- Embedding using pre-trained word vectors.
- Implementing supervised learning techniques.
- Implementing deep learning techniques.
- Implementing ensemble techniques.
- Evaluating performance both in terms of accuracy and time taken.

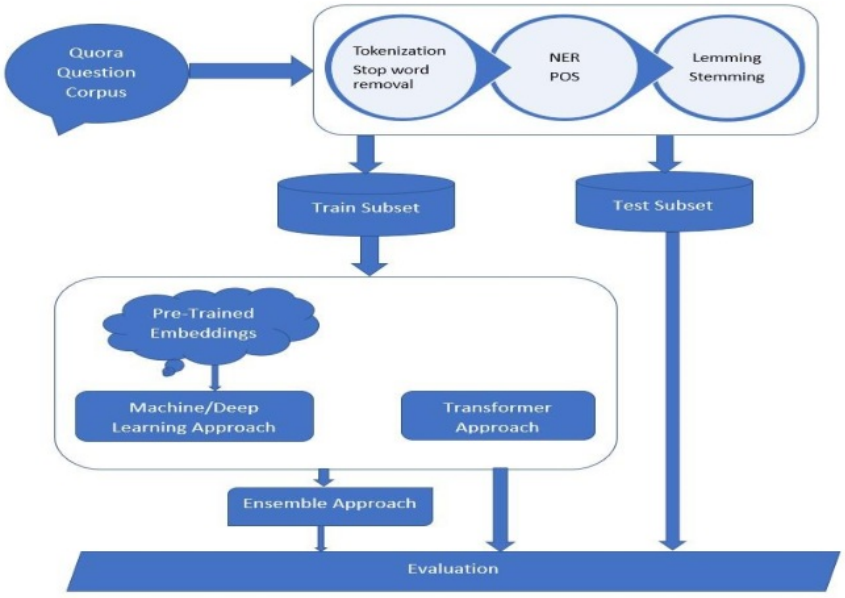


FIGURE 3 END TO END WORKFLOW

Dataset Description

As Quora is a general-purpose knowledge sharing platform we could get questions in all domains. Each record has a pair of questions (question1 & question2) and a target class(is_duplicate) which indicates whether the question are duplicate or not. Dataset contains 2,425,740 question pairs in training set, we will be considering training set for our analysis as the test set doesn't have labels provided with. There are 1,530,162(63%) question pairs with value 0(indicating no match) and 895,578(37%) with value 1(indicating match). Below table has details of data fields in the dataset with its associated description.

Data Field	Description
Id	ID of the question pair.
qid1	ID of question1.
qid2	ID of question2.
question1	Text content of question1.
question2	Text content of question2.
is_duplicate	Flag which indicates the given question is duplicate or not.

TABLE 1 ATTRIBUTES DESCRIPTION OF DATASET

We only be considering 3 columns for this study question1, question2 and Is_duplicate.

Data pre-processing

Data Profiling/Cleaning

Check for any NULL/Blanks in fields question1, question2 and is_duplicate. Check count of these columns so there is no row with missing data.

Lower Casing and remove unwanted characters

Will convert the textual data to lowercase it will be helpful in resolving duplicates words. All characters except numbers and letters can be removed as they add no value.

Stop word Elimination

A stop word is a commonly used word (such as “the”, “a”, “an”, “in”) which can be taken out as they add no value to the model.

Stemming

Stemming is a way of reducing inflection in a given word to its root form. Stemming converts the words or a collection of words to its root form stemmed word may be an invalid word in the English dictionary.

Lemmatization

Both lemming and stemming results in reducing the feature space, the key difference between them is that, lemming converts words to the root form which will be a valid word in the dictionary.

Text Augmentation

This is a process of supplementing textual data with information by providing semantic insights than what is present in original data. Example: POS tagging, NER.

Transformation

As we are going to compare various modelling approaches which is both time efficient and accurate. We will explore various pre- trained word embeddings.

Word Embeddings

Word embedding is an natural language feature engineering technique in which words or phrases from the vocabulary will be mapped to vectors of real numbers. Word embeddings have significant advantage over regular sparse word representations like one hot encoding, Word embeddings have more semantic information than them.

Embedding Layer

An embedding layer in a neural network model learns from the neural network weight bias updation when applied on natural language processing task. Preprocessing of corpus should have been completed using one hot encoding. The embedding layer is first layer of a neural network and is adjusted iteratively during the Backpropagation.

Pre-Trained Word Embeddings

Word2Vec:Word2Vec is a pre-trained word embedding which will convert word to vector preserving its semantic meaning.

GloVe: Global Vectors for Word Representation is an extension to the word2vec method for efficiently learning word vectors, developed at Stanford.

fastText: fastText is a word embedding method which is an extension of the word2vec model. Rather than learning from words directly, fastText represents words as an n-gram of characters.

Modelling

Below flow diagram depicts the architecture of the end to end modelling to be followed in this study.

As we are performing a comparative study of different text similarity approaches, we will try different combinations right from text pre-processing to modelling. All the steps mentioned above are broken into individual functions which can be used in different combinations. Dotted lines indicate individual models output which are feed to the ensemble model approach, here it can be a combination of two or more approaches.

- Different feature engineering options will be implemented.
- Fast Text Word embedding will be implemented with both machine and deep learning approaches.
- Glove Word embedding will be implemented with both machine and deep learning approaches.
- Word2Vec Word embedding will be implemented with both machine and deep learning approaches.
- Transformers like Distil Bert and Tiny Bert will be implemented.
- To build language models below algorithms will be implemented:
    - Classical machine learning algorithms: Logistic Regression, Naïve Bayes and SVM
    - Boosting algorithms: LightGBM, CatBoost and XGBoost
    - Deep learning algorithms: Bidirectional GRU and GRU + attention mechanism
- Ensemble implementation would be done by combining inputs of more than one model using both hard and soft voting approaches.
- Compare different approaches and evaluate them on performance and time efficiency.

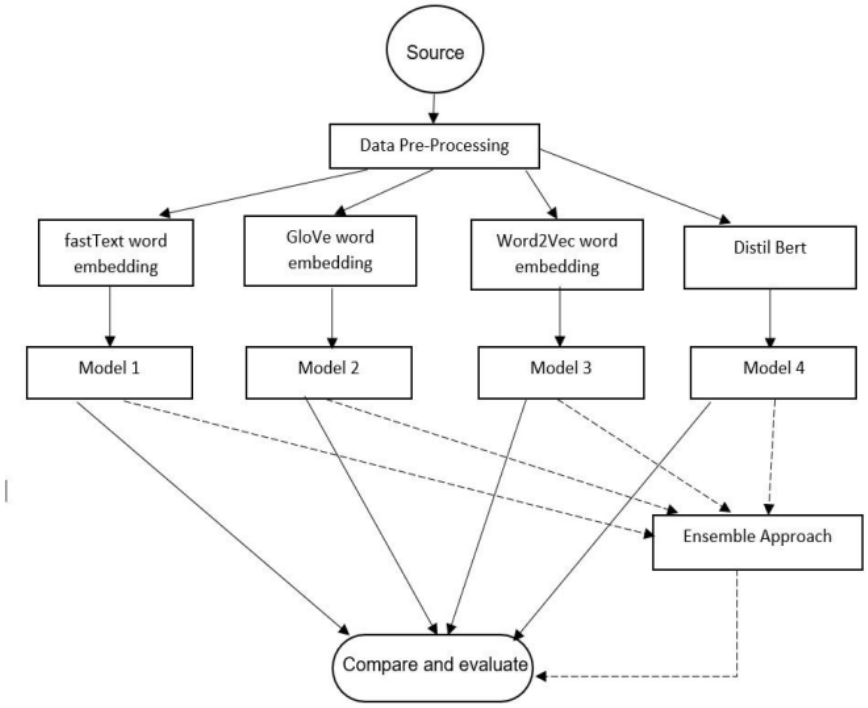


FIGURE 4 HIGH LEVEL RESEARCH APPROACH

Evaluation metrics

For performance evaluation of the language models various well-known evaluation parameters. The models in this study will be evaluated on below metrics:

- Accuracy: This can be defined as the proportion of the correct predictions out of total cases predicted. In cases of a dataset where the class imbalance is not present, this can be a good metric for evaluation.
- AUC: This is the Area Under the Curve of ROC and this illustrates to how much extent the model is good to segregate the predicted positive classes from the predicted negative classes. A good classifier model will usually have a higher AUC score, so this is one of the important metrics for evaluation.
- Precision: Precision can be measured as the proportion of the correctly predicted positive classes compared to the actual positive classes.

- Recall: This is also one of the useful metrics for classification and this measures the proportion of the positive classes identified correctly.
- F1 score: This is defined as a balance between the Precision and Recall and the value of this score is between 0 and 1
- Time taken is another metric which will be considered as an evaluation metric.

As objective of our research is find an optimal model which is accurate and time efficient, all proposed models will be evaluated on both these criteria's. As we are using ensemble techniques also in this study time taken by each constituent model also would be criteria while evaluating.

Requirements / resources

1

Hardware Requirements

In this study, experiments will be performed on PC and configurations are as below:

- Intel Core i5-850U CPU, 1.6 GHz, 4 cores
- 8 GB RAM
- Windows 10 64-bit OS
- Also, would be leveraging NimbleBox for GPU usage.

Software Requirements

Pre-Trained word embeddings:

- GloVe
- Word2Vec
- fastText

Transformer models used for transfer learning:

- Distil Bert
- Tiny Bert

Python will be used as programming language version of python would be 3.7.

Open Source libraries required:

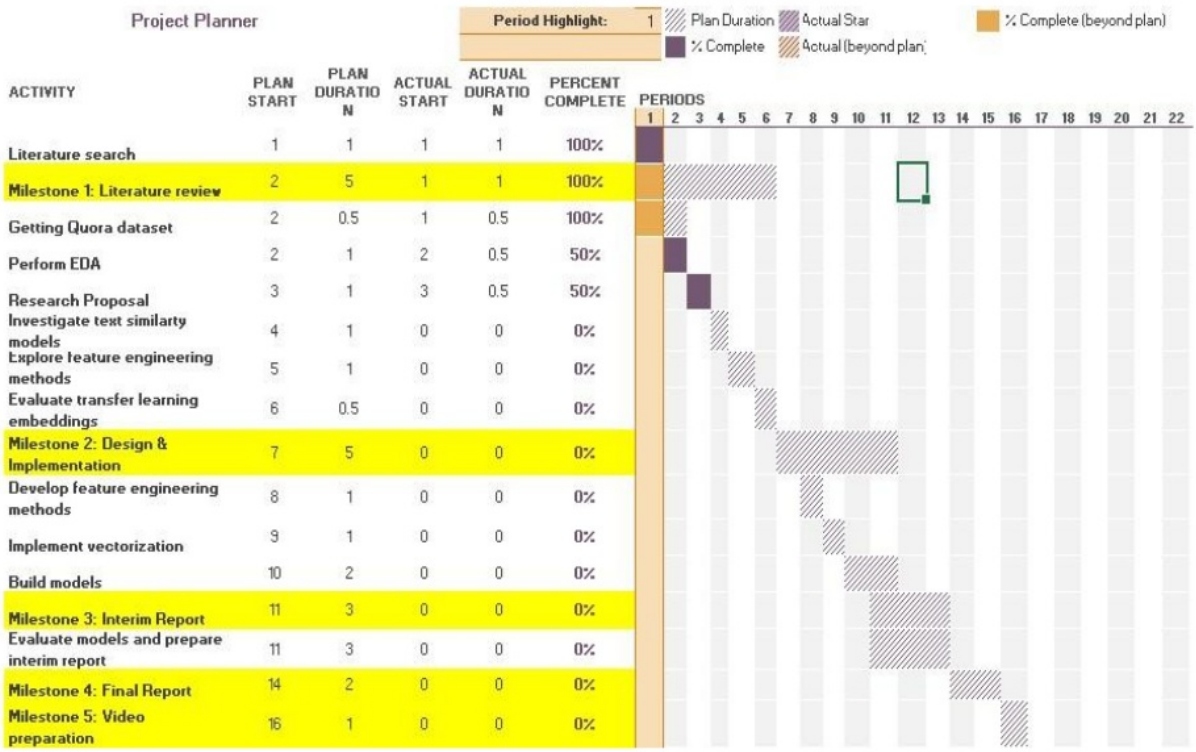
- *Numpy* for vector operations.
- *Pandas* and *Dask* for processing data.
- *Keras, sklearn, scipy* and *pytorch* for model building.
- *NLTK* and *Genism* for language processing.

Risk and Contingency Plan

Risk	Contingency Plan
NimbleBox Platform is down and unable to run the models. The laptop or hardware resource malfunctions and causes data loss.	We will be using other GPU enabled platforms like Google Colab and Kaggle as a backup measure. The code and the documents will also be stored on a Cloud drive as a backup to mitigate this kind of risks.
The proposed method/architecture is not taking the right direction	By dividing the approach in modular format, we will aim for smaller goals to achieve each task, we will also be having calls/discussions with the supervisor on regular basis making sure the approach is taking the correct path.
Unable to meet deadlines as planned due to unanticipated emergencies.	All tasks are planned with 10-15% contingency, we would be accomplishing the tasks mentioned few days before the deadline so that we can avoid unanticipated emergencies affecting the overall goal.

TABLE 2 RISK AND CONTINGENCY

Research Plan



References

Benard Magara, M., Ojo, S.O. and Zuva, T., (2018) A comparative analysis of text similarity measures and algorithms in research paper recommender systems. In: *2018 Conference on Information Communications Technology and Society, ICTAS 2018 - Proceedings.*

Chorowski, J., Bahdanau, D., Serdyuk, D., Cho, K. and Bengio, Y., (2015) Attention-based models for speech recognition. In: *Advances in Neural Information Processing Systems*.

Chung, J., Gulcehre, C., Cho, K. and Bengio, Y., (2014) Gated Recurrent Neural Networks on Sequence Modeling. *arXiv*.

Devlin, J., Chang, M.W., Lee, K. and Toutanova, K., (2019) BERT: Pre-training of deep bidirectional transformers for language understanding. In: *NAACL HLT 2019 - 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies - Proceedings of the Conference*.

Dhakal, A., Poudel, A., Pandey, S., Gaire, S. and Baral, H.P., (2018) Exploring Deep Learning in Semantic Question Matching. In: *Proceedings on 2018 IEEE 3rd International Conference on Computing, Communication and Security, ICCCS 2018*.

Farouk, M., (2019) Measuring Sentences Similarity: A Survey. *Indian Journal of Science and Technology*.

Hochreiter, S. and Schmidhuber, J., (1997) Long Short-Term Memory. *Neural Computation*.

Howard, J. and Ruder, S., (2018) *Introducing state of the art text classification with universal language models*. 15-05.

Imtiaz, Z., Umer, M., Ahmad, M., Ullah, S., Choi, G.S. and Mehmood, A., (2020) Duplicate Questions Pair Detection Using Siamese MaLSTM. *IEEE Access*.

Jiao, X., Yin, Y., Shang, L., Jiang, X., Chen, X., Li, L., Wang, F. and Liu, Q., (2019) TinyBERT: Distilling BERT for Natural Language Understanding. pp.1–14.

Liu, W., Zhang, M., Luo, Z. and Cai, Y., (2017) An Ensemble Deep Learning Method for Vehicle Type Classification on Visual Traffic Surveillance Sensors. *IEEE Access*.

Patrizio, A., (2019) *Expect 175 zettabytes of data worldwide by 2025*. NetworkWorld.

Peters, M., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K. and Zettlemoyer, L., (2018) [ELMo] [seminal LSTM contextualised embedding model] 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)*.

Reimers, N. and Gurevych, I., (2020) Sentence-BERT: Sentence embeddings using siamese BERT-networks. In: *EMNLP-IJCNLP 2019 - 2019 Conference on Empirical Methods in Natural Language Processing and 9th International Joint Conference on Natural Language Processing, Proceedings of the Conference*.

Sanh, V., Debut, L., Chaumond, J. and Wolf, T., (2019) DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. pp.2–6.

Sharma, L., Graesser, L., Nangia, N. and Evci, U., (2019) Natural Language Understanding with the Quora Question Pairs Dataset. *arXiv preprint arXiv:1907.01041*.

Shaw, P., Uszkoreit, J. and Vaswani, A., (2018) Self-attention with relative position representations. In: *NAACL HLT 2018 - 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies - Proceedings of the Conference*.

Wang, L., Zhang, L. and Jiang, J., (2020) Duplicate Question Detection with Deep Learning in Stack Overflow. *IEEE Access*.

Appendix C

Metrics captured as part of the study.

Training result metrics of machine learning language models considered for the study.

Model	train accuracy	f1 train	roc_auc train	precision train	recall train	embedding Time	training Time
lem-glove300-overSample-lr	0.669	0.617	0.578	0.662	0.669	260.209	35.979
lem-glove300-stratified-lr	0.669	0.618	0.579	0.662	0.669	259.422	35.098
lem-glove300-underSample-lr	0.671	0.626	0.585	0.661	0.671	266.388	37.112
lem-glove300-overSample-cat	0.716	0.697	0.656	0.711	0.716	265.401	368.754
lem-glove300-underSample-cat	0.717	0.698	0.657	0.712	0.717	260.872	371.815
lem-glove300-stratified-cat	0.717	0.698	0.657	0.713	0.717	261.378	370.995
lem-glove300-overSample-light	0.798	0.787	0.749	0.804	0.798	263.315	2529.797
lem-glove300-underSample-light	0.798	0.787	0.749	0.805	0.798	269.589	2534.271
lem-glove300-stratified-light	0.799	0.788	0.751	0.805	0.799	257.280	2570.206
lem-tfidf-stratified-lr	0.683	0.633	0.593	0.686	0.683	43.213	87.694
lem-tfidf-overSample-lr	0.684	0.660	0.617	0.672	0.684	42.975	85.759
lem-tfidf-underSample-lr	0.685	0.660	0.618	0.673	0.685	50.647	89.220
lem-tfidf-overSample-cat	0.705	0.674	0.631	0.703	0.705	44.325	466.967
lem-tfidf-underSample-cat	0.706	0.677	0.634	0.705	0.706	43.918	454.212
lem-tfidf-stratified-cat	0.706	0.677	0.634	0.704	0.706	44.016	450.881
lem-tfidf-stratified-light	0.741	0.723	0.682	0.741	0.741	46.238	590.312
lem-tfidf-overSample-light	0.741	0.723	0.682	0.741	0.741	43.525	583.403
lem-tfidf-underSample-light	0.741	0.724	0.683	0.741	0.741	45.993	579.842
lem-word2vec-overSample-lr	0.628	0.579	0.543	0.622	0.628	268.744	37.159
lem-word2vec-stratified-lr	0.629	0.581	0.544	0.622	0.629	267.931	36.249
lem-word2vec-underSample-lr	0.630	0.587	0.549	0.620	0.630	275.125	38.329
lem-word2vec-overSample-cat	0.673	0.654	0.616	0.668	0.673	274.106	380.849
lem-word2vec-underSample-cat	0.673	0.655	0.616	0.669	0.673	269.428	384.011
lem-word2vec-stratified-cat	0.674	0.655	0.616	0.669	0.674	269.951	383.163
lem-word2vec-overSample-light	0.749	0.739	0.703	0.755	0.749	271.951	2612.774
lem-word2vec-underSample-light	0.750	0.739	0.704	0.756	0.750	278.431	2617.395
lem-word2vec-stratified-light	0.750	0.740	0.705	0.756	0.750	265.719	2654.509
N-glove300-underSample-lr	0.676	0.632	0.591	0.669	0.676	0.676	396.878
N-glove300-underSample-light	0.803	0.794	0.760	0.805	0.803	0.754	388.499
N-glove300-underSample-cat	0.725	0.712	0.673	0.719	0.725	0.712	386.682

N-glove300-overSample-lr	0.670	0.611	0.574	0.671	0.670	0.670	384.892
N-glove300-overSample-light	0.803	0.794	0.760	0.805	0.803	0.755	387.241
N-glove300-overSample-cat	0.724	0.710	0.672	0.718	0.724	0.713	386.266
N-glove300-stratified-lr	0.671	0.614	0.577	0.670	0.671	0.671	385.257
N-glove300-stratified-light	0.802	0.794	0.759	0.805	0.802	0.755	386.920
N-glove300-stratified-cat	0.723	0.709	0.670	0.717	0.723	0.714	385.942
N-tfidf-underSample-lr	0.694	0.658	0.616	0.693	0.694	0.693	22.162
N-tfidf-underSample-light	0.777	0.769	0.735	0.775	0.777	0.756	21.721
N-tfidf-underSample-cat	0.726	0.712	0.673	0.721	0.726	0.725	21.507
N-tfidf-overSample-lr	0.696	0.667	0.624	0.689	0.696	0.696	21.493
N-tfidf-overSample-light	0.776	0.768	0.734	0.773	0.776	0.755	21.627
N-tfidf-overSample-cat	0.728	0.715	0.678	0.722	0.728	0.724	21.770
N-tfidf-stratified-lr	0.695	0.676	0.635	0.684	0.695	0.696	21.470
N-tfidf-stratified-light	0.777	0.769	0.736	0.775	0.777	0.755	21.590
N-tfidf-stratified-cat	0.727	0.714	0.676	0.721	0.727	0.725	21.866
N-word2vec-underSample-lr	0.654	0.612	0.572	0.647	0.654	0.684	401.641
N-word2vec-underSample-light	0.777	0.769	0.735	0.779	0.777	0.763	393.161
N-word2vec-underSample-cat	0.702	0.689	0.652	0.696	0.702	0.721	391.322
N-word2vec-overSample-lr	0.649	0.591	0.556	0.649	0.649	0.678	389.511
N-word2vec-overSample-light	0.777	0.769	0.736	0.779	0.777	0.764	391.888
N-word2vec-overSample-cat	0.701	0.688	0.650	0.695	0.701	0.722	390.902
N-word2vec-stratified-lr	0.649	0.595	0.558	0.648	0.649	0.679	389.880
N-word2vec-stratified-light	0.777	0.769	0.735	0.779	0.777	0.764	391.563
N-word2vec-stratified-cat	0.700	0.686	0.649	0.694	0.700	0.722	390.573
stem-glove300-underSample-lr	0.652	0.553	0.536	0.674	0.652	307.232	34.450
stem-glove300-overSample-lr	0.661	0.590	0.559	0.660	0.661	304.486	35.979
stem-glove300-stratified-lr	0.665	0.651	0.613	0.651	0.665	306.399	34.811
stem-glove300-overSample-cat	0.714	0.690	0.648	0.711	0.714	304.661	362.558
stem-glove300-stratified-cat	0.714	0.691	0.648	0.711	0.714	306.424	364.887
stem-glove300-underSample-cat	0.715	0.692	0.649	0.711	0.715	302.237	361.907
stem-glove300-underSample-light	0.790	0.777	0.738	0.797	0.790	303.935	2438.455
stem-glove300-overSample-light	0.791	0.779	0.740	0.798	0.791	304.114	2444.972
stem-glove300-stratified-light	0.791	0.779	0.740	0.798	0.791	305.073	2465.918
stem-tfidf-stratified-lr	0.681	0.625	0.587	0.690	0.681	103.040	83.725
stem-tfidf-overSample-lr	0.682	0.635	0.594	0.683	0.682	101.333	82.896
stem-tfidf-underSample-lr	0.683	0.637	0.596	0.684	0.683	122.981	80.467
stem-tfidf-overSample-cat	0.707	0.676	0.632	0.708	0.707	102.522	432.647
stem-tfidf-underSample-cat	0.707	0.678	0.635	0.707	0.707	101.564	436.776
stem-tfidf-stratified-cat	0.708	0.679	0.636	0.706	0.708	101.497	434.406
stem-tfidf-stratified-light	0.743	0.726	0.686	0.743	0.743	100.943	579.195
stem-tfidf-underSample-light	0.744	0.727	0.686	0.745	0.744	102.971	578.318

stem-tfidf-overSample-light	0.744	0.727	0.686	0.745	0.744	101.440	576.248
stem-word2vec-underSample-lr	0.612	0.519	0.503	0.633	0.673	317.309	35.580
stem-word2vec-overSample-lr	0.620	0.554	0.525	0.620	0.682	314.473	37.159
stem-word2vec-stratified-lr	0.624	0.612	0.575	0.611	0.687	316.449	35.953
stem-word2vec-overSample-cat	0.670	0.648	0.608	0.667	0.737	314.654	374.450
stem-word2vec-stratified-cat	0.670	0.649	0.609	0.667	0.737	316.475	376.855
stem-word2vec-underSample-cat	0.671	0.649	0.610	0.668	0.738	312.150	373.778
stem-word2vec-underSample-light	0.742	0.730	0.693	0.748	0.816	313.904	2518.436
stem-word2vec-overSample-light	0.743	0.731	0.694	0.750	0.817	314.089	2525.167
stem-word2vec-stratified-light	0.743	0.731	0.695	0.749	0.817	315.079	2546.800

Testing result metrics of the machine learning language models considered for the study.

Model	test accuracy	f1_test	roc_auc_test	precision_test	recall_test	embeddingTime	predTestTime
lem-glove300-stratified-lr	0.670	0.619	0.580	0.663	0.670	260.209	0.063
lem-glove300-underSample-lr	0.672	0.621	0.582	0.666	0.672	259.422	0.057
lem-glove300-overSample-cat	0.670	0.625	0.584	0.660	0.670	266.388	0.065
lem-glove300-stratified-cat	0.706	0.686	0.645	0.699	0.706	265.401	1.066
lem-glove300-underSample-cat	0.708	0.687	0.646	0.701	0.708	261.378	0.969
lem-glove300-stratified-light	0.709	0.688	0.647	0.702	0.709	260.872	1.096
lem-glove300-underSample-light	0.753	0.738	0.699	0.754	0.753	257.280	2.216

lem-glove300-overSample-light	0.753	0.739	0.699	0.754	0.753	269.589	2.231
lem-glove300-stratified-lr	0.754	0.740	0.700	0.755	0.754	263.315	2.183
lem-tfidf-overSample-lr	0.681	0.631	0.591	0.685	0.681	43.213	0.096
lem-tfidf-underSample-lr	0.684	0.660	0.618	0.672	0.684	42.975	0.061
lem-tfidf-overSample-cat	0.684	0.660	0.618	0.672	0.684	50.647	0.065
lem-tfidf-underSample-cat	0.703	0.672	0.629	0.702	0.703	44.325	6.882
lem-tfidf-stratified-cat	0.702	0.672	0.629	0.700	0.702	43.918	7.454
lem-tfidf-underSample-light	0.704	0.675	0.632	0.702	0.704	44.016	6.712
lem-tfidf-stratified-light	0.727	0.708	0.666	0.724	0.727	45.993	2.167
lem-tfidf-overSample-light	0.727	0.708	0.666	0.725	0.727	46.238	2.331
lem-tfidf-overSample-lr	0.730	0.711	0.669	0.728	0.730	43.525	2.313
lem-word2vec-stratified-lr	0.630	0.582	0.545	0.623	0.630	267.964	0.064
lem-word2vec-underSample-lr	0.632	0.584	0.547	0.627	0.632	267.153	0.059
lem-word2vec-overSample-cat	0.630	0.587	0.549	0.620	0.630	274.326	0.067
lem-word2vec-stratified-cat	0.664	0.645	0.606	0.657	0.664	273.310	1.098

lem-word2vec-underSample-cat	0.665	0.646	0.607	0.659	0.665	269.167	0.998
lem-word2vec-stratified-light	0.666	0.647	0.608	0.660	0.666	268.646	1.129
lem-word2vec-underSample-light	0.708	0.694	0.657	0.709	0.708	264.947	2.282
lem-word2vec-overSample-light	0.708	0.694	0.657	0.709	0.708	277.623	2.297
lem-word2vec-overSample-lr	0.709	0.695	0.658	0.710	0.709	271.161	2.248
N-glove300-stratified-lr	0.670	0.611	0.574	0.670	0.670	384.892	0.049
N-glove300-underSample-lr	0.671	0.614	0.577	0.671	0.671	385.257	0.051
N-glove300-underSample-cat	0.676	0.632	0.591	0.669	0.676	396.878	0.060
N-glove300-overSample-cat	0.712	0.697	0.658	0.705	0.712	386.682	0.946
N-glove300-stratified-cat	0.713	0.698	0.659	0.705	0.713	386.266	0.914
N-glove300-underSample-light	0.714	0.699	0.661	0.706	0.714	385.942	0.908
N-glove300-stratified-light	0.754	0.743	0.705	0.752	0.754	388.499	2.167
N-glove300-	0.755	0.743	0.706	0.752	0.755	386.920	2.205

overSample-light							
N-glove300-underSample-lr	0.755	0.744	0.706	0.753	0.755	387.241	2.281
N-tfidf-overSample-lr	0.693	0.655	0.613	0.691	0.693	22.162	0.100
N-tfidf-stratified-lr	0.696	0.667	0.624	0.690	0.696	21.493	0.106
N-tfidf-underSample-cat	0.696	0.677	0.636	0.686	0.696	21.470	0.091
N-tfidf-overSample-cat	0.725	0.710	0.671	0.719	0.725	21.507	6.759
N-tfidf-stratified-cat	0.724	0.711	0.673	0.718	0.724	21.770	6.617
N-tfidf-stratified-light	0.725	0.712	0.674	0.718	0.725	21.866	6.903
N-tfidf-overSample-light	0.755	0.746	0.711	0.750	0.755	21.590	2.054
N-tfidf-underSample-light	0.755	0.746	0.711	0.751	0.755	21.627	1.966
N-tfidf-overSample-lr	0.756	0.746	0.712	0.752	0.756	21.721	1.967
N-word2vec-stratified-lr	0.652	0.594	0.559	0.652	0.652	388.279	0.049
N-word2vec-underSample-lr	0.653	0.597	0.561	0.652	0.653	388.647	0.052
N-word2vec-underSample-cat	0.657	0.614	0.574	0.650	0.657	400.371	0.061
N-word2vec-overSample-cat	0.693	0.678	0.640	0.685	0.693	390.085	0.954
N-word2vec-	0.693	0.679	0.641	0.686	0.693	389.665	0.922

stratified-cat							
N-word2vec-underSample-light	0.694	0.680	0.642	0.686	0.694	389.338	0.916
N-word2vec-stratified-light	0.733	0.722	0.686	0.731	0.733	391.918	2.186
N-word2vec-overSample-light	0.734	0.722	0.686	0.732	0.734	390.325	2.224
N-word2vec-underSample-lr	0.734	0.723	0.687	0.732	0.734	390.648	2.301
stem-glove300-overSample-lr	0.652	0.553	0.536	0.672	0.652	307.232	0.050
stem-glove300-stratified-lr	0.662	0.592	0.560	0.662	0.662	304.486	0.049
stem-glove300-overSample-cat	0.663	0.649	0.610	0.649	0.663	306.399	0.049
stem-glove300-underSample-cat	0.706	0.682	0.639	0.702	0.706	304.661	0.912
stem-glove300-stratified-cat	0.706	0.682	0.640	0.701	0.706	302.237	1.063
stem-glove300-overSample-light	0.707	0.683	0.641	0.701	0.707	306.424	0.936
stem-glove300-stratified-light	0.748	0.731	0.690	0.749	0.748	304.114	2.188
stem-glove300-underSample-light	0.750	0.734	0.693	0.752	0.750	305.073	2.247

stem-glove300-stratified-lr	0.750	0.734	0.693	0.752	0.750	303.935	2.314
stem-tfidf-overSample-lr	0.680	0.624	0.586	0.688	0.680	103.040	0.082
stem-tfidf-underSample-lr	0.683	0.635	0.595	0.685	0.683	101.333	0.080
stem-tfidf-overSample-cat	0.683	0.637	0.596	0.684	0.683	122.981	0.071
stem-tfidf-underSample-cat	0.705	0.674	0.630	0.705	0.705	102.522	6.195
stem-tfidf-stratified-cat	0.706	0.676	0.633	0.705	0.706	101.564	6.494
stem-tfidf-overSample-light	0.706	0.677	0.634	0.704	0.706	101.497	6.613
stem-tfidf-underSample-light	0.729	0.710	0.668	0.727	0.729	101.440	2.233
stem-tfidf-stratified-light	0.731	0.713	0.671	0.729	0.731	102.971	2.205
stem-tfidf-underSample-lr	0.732	0.714	0.673	0.730	0.732	100.943	2.274
stem-word2vec-overSample-lr	0.613	0.520	0.504	0.632	0.613	316.387	0.051
stem-word2vec-stratified-lr	0.622	0.556	0.526	0.623	0.622	313.560	0.051
stem-word2vec-overSample-cat	0.623	0.610	0.574	0.610	0.623	315.530	0.050
stem-word2vec-underSample-cat	0.664	0.641	0.601	0.660	0.664	313.740	0.939
stem-word2vec-stratified-cat	0.664	0.642	0.602	0.659	0.664	311.244	1.095

stem-word2vec-overSample-light	0.664	0.642	0.602	0.659	0.664	315.556	0.964
stem-word2vec-stratified-light	0.703	0.687	0.649	0.705	0.703	313.176	2.253
stem-word2vec-underSample-light	0.705	0.690	0.651	0.707	0.705	314.164	2.314
stem-word2vec--	0.705	0.690	0.652	0.707	0.705	312.992	2.383

Appendix D

GitHub Link: https://github.com/vasist1987/MS_LJMU_Vasist/tree/master

final thesis.pdf

ORIGINALITY REPORT

2%	0%	0%	2%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Submitted to Liverpool John Moores University Student Paper	2%

Exclude quotes	Off	Exclude matches	< 1%
Exclude bibliography	On		