

THIS SHEET IS TO BE SIGNED AND DISPLAYED AT THE FRONT OF YOUR COURSEWORK

All sections except the "For official use only" section must be completed and the declaration below signed, for the submission to be accepted. All coursework should be submitted electronically directly to The Graduate School CampusMoodle in the relevant drop box provided.


Due Date
27/01/2020

Date submitted
27/01/2020

For official use only

SURNAME	UPADHYAY
FIRST NAME(S)	ASHISH
STUDENT ENROLMENT NUMBER	1716293
COURSE	POSTGRADUATE CERTIFICATE RESEARCHER DEVELOPMENT
MODULE NUMBER	GSM010 (Module 1)
MODULE TITLE	FUNDAMENTALS IN RESEARCHER DEVELOPMENT

- I confirm:
- (a) That the work undertaken for this assignment is entirely my own and that I have not made use of any unauthorised assistance.
 - (b) I confirm that the version submitted here is the identical version of text that has been submitted to TurnitinUK for plagiarism detection (with the exception of the reference list and appendices).
 - (c) That the sources of all reference material have been properly acknowledged. Refer Regulation A3-Section 2 at: www.rgu.ac.uk/academicregulations
 - (d) The University operates a Fit to Sit Policy which means that if you undertake an assessment then you are declaring yourself well enough to do so. Further details are available at: www.rgu.ac.uk/academicregulationsstudentforms

Signed: 

Date: 27/01/2020

For official use only

Provisional Grade:

Please note that all Grades are provisional until ratified by the Assessment Board.

Machine/Deep Learning based Performance Enhancement of Natural Language Processing in Business Processes

Ashish Upadhyay (a.upadhyay@rgu.ac.uk)

Principle Supervisor: Dr. Stewart Massie (s.massie@rgu.ac.uk)

Abstract

With the evolution of web and cloud computing the amount of data generated is higher than ever. Most of the data is in unstructured textual format. The generated data can help many industries automate their everyday business processes by analysing and processing them. There is a strong need for the development of effective approaches that can process the natural language automatically. Initial solutions to these problems were applying custom rule-based approaches that can be very specific to the problem domains. They require a lot of domain expertise for crafting the features and the performance with these methods was also poor. New approaches employed with machine and deep learning methods for solving the natural language related tasks has given a huge performance gain in the literature due to their ability to learn from the previous data. But the methods proposed in academic research often fail when implemented in real-world applications due to the difference in nature of problem in business processes. One of the drawbacks can be the requirement of a lot of labelled data for training the learning models as in the business processes the labelled data is generated as the part of the process.

In this research we will identify these kind of problems faced by business process during the implementation of current state-of-the-art methods and help them improve their natural language processing performance employed with machine/deep learning. We expect our research contributions to be in the area of business process automation with focus on machine and deep learning. Importantly this work will develop novel algorithms and possibly some dataset which will be made available in public doamin with the aim of helping industries with less resources to enhance their everyday business process.

Contents

1	Introduction	4
2	Research Questions and Objectives	5
3	Literature Review	6
3.1	Tasks	6
3.1.1	Extraction	7
3.1.2	Generation	8
3.1.3	Summarisation	9
3.2	Knowledge Engineering	9
3.2.1	Case Based Reasoning	10
3.2.2	Machine Learning	12
3.3	Representation Learning	14
3.3.1	Word Embeddings	14
3.3.2	Language Models	14
3.3.3	Recurrent Neural Networks	15
3.3.4	Long Short-Term Memory Networks	16
3.3.5	Convolutional Neural Networks	17
3.3.6	Sequence to Sequence Models	19
3.3.7	Contextual Word Embeddings	20
3.3.8	Transformers	21
4	Proposed Methodology	24
4.1	Objective 1	24
4.2	Objective 2	25
4.3	Objective 3	25
4.4	Objective 4	25
4.4.1	Reducing the Data Requirement	26
4.4.2	Generating New Training Data	26
5	Initial Work	26
5.1	Natural Language Generation (NLG)	27
5.1.1	Obituary Generation	27
5.2	Text Classification	27
5.2.1	Compliance Management	27
5.2.2	Weighted Ensemble of Text Classifiers	28
6	Identified Impact	28
7	Time Management Plan	30
	Appendices	35
A	Vitae Personal Development Planner	35

1 Introduction

The evolution of the Web combined with the emergence of cloud computing has led to an enormous growth in the amounts of data and information available for organisations to employ in their business processes. The statistics suggest 80% of business-relevant information originates in unstructured form, typically text; the volume of generated data is growing with predictions that by 2023, we'll be sending over 350 billion emails daily (Tschabitscher 2020). These statistics make a strong case for the development of effective approaches for natural language processing.

Sentiment analysis is an area of natural language processing that has delivered good performance, largely due to the general nature of the challenge (Sun et al. 2019, Yang et al. 2019). Effective solutions have been developed, initially employing lexicon-based approaches (Mukras et al. 2007), but more recently supplemented with Machine Learning and Deep Learning approaches (Devlin et al. 2018). Solutions have become more refined in looking at aspect-based sentiment analysis (Bandhakavi et al. 2018) and more fine grained in looking at emotion analysis (Bandhakavi et al. 2017). As a result of the improved performance from these developments, sentiment analysis is now a commonplace approach employed by industry.

The challenges is to develop approaches that go beyond sentiment analysis and to develop NLP solutions supporting business processes more generally (Budek 2019). One on-going requirement is to support the extraction of actionable knowledge from text for custom business processes (Bordignon et al. 2018). Custom rule-based solutions have been developed that typically fill out slots in predefined templates. But these methods lack in giving good performance as well as they are too domain specific. By just changing the problem domain, similar kind of problems can not be solved with the same approach. In this project we hope to develop machine and deep learning based approaches that can outperform the custom rule-based solutions and make similar performance gains that has been seen recently in sentiment analysis.

There are many business processes that require NLP solutions like document classification, text generation/summarisation, or knowledge extraction as part of the operation workflow e.g. risk assessment, compliance management, auditing, recruitment, and procurement (Sintoris & Vergidis 2017). In many industries, compliance management involve use of a large number of regulatory documents from different statutory bodies at various scenarios. The possible problems statements can be: sorting and prioritizing the documents based on their source organisation, e.g., HSE, BSI; each regulatory document may contain a large number of compliance requirement (CR) that needs to be identified and extracted; after extraction the CR needs to be classified by importance and assigned to different job roles.

There are multiple challenges in relation to these problem statements;

- NLP techniques can struggle when they encounter proprietary formats, such as pdf, because the structure of the document on which approaches may rely is lost.
- Automated business processes are expected to perform with extremely high accuracy in order to eliminate the manual intervention. Employing feedback loops is one approach to improving accuracy but are expensive to embed in rule-based techniques.
- Business processes may have unbalanced cost of error. For example, in case of CR extraction the focus is on minimising the false positives.

There are alternative use cases, e.g. assessing applicants' suitability for a job from analysis of CVs. For a company getting hundreds or thousands of applications for some job roles, it would be very difficult for them to hire a lot of experts in order to scrutinize the CVs. It would be very helpful for such company if an automated system is developed capable of filtering these CVs and selecting suitable candidates for further processes. The alternative use cases share similar kind of challenges.

Machine and Deep learning approaches can address the above challenges but rely on availability of labelled data, which is expensive to acquire. Business processes get labelled data as a part of their process. It is not feasible to wait for a long amount of time to get enough labelled data for supervised training. We need to build systems which are reliable with less training data as well. This scenario is referred as cold start problem where we have very minimal labelled to start with and as time goes on the labelled data builds up gradually as well.

The rest of the paper is organised as follows: in chapter 2, we talk about the research question and the objectives we hope to meet in order to answer the question; in chapter 3, we review the literature and talk about different tasks and methods in natural language processing; in the next chapter 4, we present our methodology that we hope to adopt in order to meet our objectives followed by some initial work in chapter 5 and identified impact in chapter 6. Finally in chapter 7, we conclude the paper with time management plan for next three years of the PhD.

2 Research Questions and Objectives

This research will explore the problems involved in text mining in a business process with the goal of improving performance of current state-of-the-art performance. The primary research question for this project will be:

Can Machine Learning help improve the performance of natural language processing related problems in business processes?

Business processes use different automated systems to solve the NLP related tasks. They mostly use the state-of-the-art methods proposed by academic research which is not always enough as the environment where academic research is conducted is different from the real-world scenario of a business process. We hope to meet following objectives in order to answer the research question.

Objectives

- O1** Identifying the NLP related problems in business processes that are not well addressed by academic research.
 - Academic research simplifies a real-world problem in order to gain good performance like working with a pre-processed labelled data.
 - But in a real world scenario, enough labelled data might not be available for training of learning algorithms. The data is generated as a part of the business process.
- O2** Identifying the common use-cases from business processes and gathering data for those use-cases.
 - There are different types of problems related to NLP in business process like natural language generation, text classification, or question answering.
 - We will identify some use-cases that are very common throughout the industry.
 - While selecting the use-case, focus will also be on the availability of datasets.
- O3** Setting up the benchmark performance by implementing state-of-the-art algorithms for the real-world scenario.

- By using public datasets, we will simulate the real-world scenario to apply the current state-of-the-art algorithms. For example, this can be achieved by applying time stamps on the public datasets.
- This will identify the performance of current state-of-the-art algorithms on real-world scenario.

O4 Developing novel methods to outperform the current state-of-the-art algorithms and evaluating them with real world data on different use-cases.

- After we identify the gaps between business processes and academic research we will propose our algorithms that can enhance the performance of NLP systems in business processes.
- We will also evaluate our methods on real-world scenario, if possible, with some industrial data.

3 Literature Review

Natural Language Processing (NLP) has come a long way from the era of batch processing and punch cards where a sentence analysis query used to take almost seven minutes to the era of search engines like Google and Bing where millions of web pages are crawled within seconds to produce the best results (Cambria & White 2014). NLP is a way for computers to perform natural language related tasks like information extraction (like POS tagging, Named Entity Extraction) or language generation and summarisation (like machine translation, dialogue systems) (Young et al. 2018).

In the next section 3.1, we briefly discuss the various tasks of NLP on different levels. To solve these tasks, there are various methods in the literature which we broadly divide into two categories; Knowledge Engineering (section 3.2) and Representation Learning (or in common words, deep learning section 3.3). Earlier, knowledge engineering approaches were used to solve NLP tasks. But problem with these methods is that they rely on domain expertise for engineering the knowledge from data. With the development and success of deep learning methods in vision tasks, deep learning has been employed for NLP tasks as well and has been successful as well. The benefit of these methods is that they do not need any special knowledge engineering as they automatically learn features modelling the data during their training phase. This learning is iteratively updated in various steps during training.

3.1 Tasks

NLP research is mostly driven by the basic question, how do humans understand the natural language and how can we make the machines do the same? In the research, understanding the natural language for machines differs from lower level problems of extracting information from the text like named entity or relation extraction to higher level problems like machine translation or question answering. In this section we will discuss about the different kind of tasks involved in NLP dividing them into three broad categories:

1. Extraction
2. Generation
3. Summarisation

3.1.1 Extraction

Knowledge Extraction is a sub field of NLP that deals with the automated extraction of structured knowledge from unstructured sources. These are the few examples of the tasks that can be treated as knowledge extraction.

Named Entity Recognition Named Entity Recognition is the a sub task of Knowledge Extraction that deals with identifying the various entities in a piece of text such as person, location, organization and date. It has various use cases, like automating hiring process or optimizing the search engine algorithms. One of the main approach is to use BIO notation (B for beginning, I for inside and O for non-entity tokens.) For example, in the sentence '*Boris Johnson won the UK general elections*' the entities can be marked as [*Boris*]_{B-PER} [*Johnson*]_{I-PER} [*won*]_O [*the*]_O [*UK*]_{LOC} [*general*]_O [*elections*]_O. PER represents person, LOC represents location and O is the non-entity tokens.

There are various open source libraries for NER such as spaCy which uses hand crafted features (not published) and StanfordNLP which uses a combination of hand crafted features and deep learning for the same task. One of the most popular benchmark for NER is CoNLL 2003 (Sang & De Meulder 2003) consisting text from news wires like Reuters dataset tagged in four different categories (PER, LOC, ORG, MISC). The state-of-the-arts in literature for NER right now are mostly based on the transformers architecture which we will discuss in section 3.3.8

Parts-of-Speech Tagging deals with the marking of parts of speech of a word in the text corpus. The category of words having similar grammatical properties can be termed as parts-of-speech. Some of the common parts-of-speech in English language are noun, pronoun, adjective, preposition, verb, adverb etc. NLTK is one of the most common open-sourced library used for POS tagging. A example of POS tagging in NLTK for the sentence '*And now for something completely different*' can be [('*And*', 'CC'), ('*now*', 'RB'), ('*for*', 'IN'), ('*something*', 'NN'), ('*completely*', 'RB'), ('*different*', 'JJ')]. Here CC is coordinating conjunction, RB is used for adverbs, IN is a preposition, NN is noun and JJ is adjective ¹.

The popular benchmarks for POS tagging are Penn Treebank (Marcus et al. 1993) and Universal Dependencies (Silveira et al. 2014). Penn Treebank portion of Wall Street Journal (WSJ) consists of 45 different POS tags whereas Universal Dependencies consists of more than 100 treebanks from 60 different languages. The current state-of-the-arts for POS tagging are mostly based on LSTM and BERT.

Natural Language Inference For a given piece of text or premise, finding that whether some 'hypothesis' about that text is true or not is called natural language inference. If a hypothesis is true then it is known as 'entailment', if it is false then 'contradiction' and for undetermined 'neutral'. For example, lets have a look at the table 1 ².

Table 1: NLI Example

Premise	Label	Hypothesis
A man inspects the uniform of a figure in some East Asian country.	contradiction	The man is sleeping.

¹Example taken from <https://www.nltk.org/>

²Example taken from <https://nlpprogress.com/>

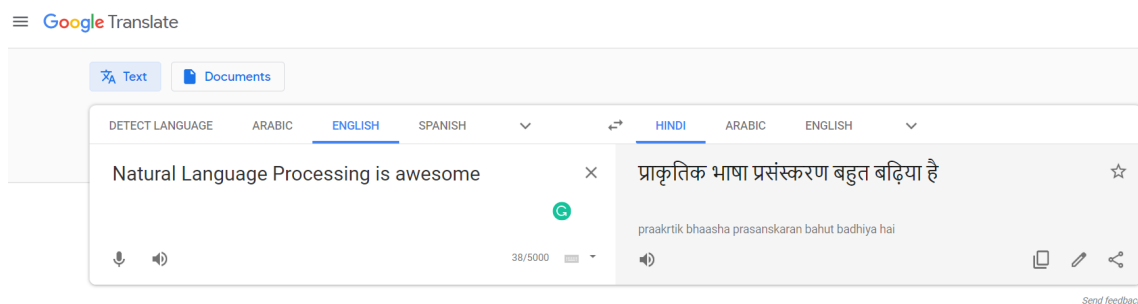


Figure 1: An Example of Machine Translation

The two most common corpus for NLI are Multi-Genre Natural Language Inference (Multi-NLI) (Williams et al. 2017) and Stanford Natural Language Inference (SNLI) (Bowman et al. 2015). Both of them contain around 500k labelled hypothesis/premise pairs. Current state of the arts are based on the slight modification of BERT architecture.

Relation Extraction The task of identifying semantic relationships between entities of text is known as relation extraction. Usually the relations are defined between two or more entities like person, location or organisation and fall under various pre-defined semantic categories.

There are lots of public datasets available for relation extraction and all of them define their own category of relations that need to be extracted from the text. Few examples are; Capturing discriminative attributes (SemEval 2018 Task 10) (Krebs et al. 2018) where users are asked to identify if a given attribute can help in discrimination between two concepts. Few-Shot Relation Classifier (FewRel) (Han et al. 2018) consisting of 70k sentences with 100 different kinds of relation annotated by crowd-sourcing on wikipedia corpus. New York Times corpus (Riedel et al. 2010) containing entities extracted from NYT corpus and linked to the freebase knowledge base.

3.1.2 Generation

Natural Language Generation (NLG) can be termed as the sub-field of NLP which deals with the generation of natural language given some structured or unstructured information. Machine Translation and Data Augmentation can be few examples of NLG.

Machine Translation Machine Translation has been one of the most popular problems in NLP. It deals with the translation of text from one natural language to another, for example, translating English text to Hindi or vice-versa. A snippet from Google translate performing MT is shown in fig. 1³.

The MT models are evaluated on the method called BLEU score which is used as the higher the better. Some of the common public datasets available are WMT 2014 EN-DE and WMT 2014 EN-FR (Bojar et al. 2014) which contains many english-german and english-french pair of sentences. The current state-of-the-arts are based on mainly transformers and some attention based LSTMs.

³<https://translate.google.com/>

Data Augmentation With the popularity of Deep Learning based NLP, there has been a huge requirement of training data for building the models. Collecting the labelled data is sometime very costly and take a lot of resource in terms of time and money.

The alternative way of getting more data is to generate a huge corpus of data by tweaking the using the small amount of training data available in some fashion. This technique is known as data augmentation. The way of tweaking differs from problem to problem. There are different ways of data augmentation like unsupervised back translation with consistency training (Xie et al. 2019) or contextualized word replacement (Kobayashi 2018).

3.1.3 Summarisation

The third category of NLP tasks is termed as the problem of summarisation. Given a piece of text, how can we summarize the information it posses in simpler and/or structured manner.

Text Classification Classifying a piece of text/document into some pre-defined categories can be termed as the task of text classification. Text classification is mainly used in applications like sentiment analysis and/or topic and document classification.

For example, in sentiment analysis we try to identify the sentiment or polarity of a sentence based on its content. Given a review about a product on Amazon and identifying if the customer who wrote the review is positive or negative about the product's quality. Some of the common examples for topic classification can be identifying the domain (political, religious or technological) of a news article based on its content.

Some popular datasets for sentiment analysis are; imdb movie reviews which contains 50,000 reviews of different movies from imdb website labelled into positive and negative classes. They also contain a numeric value from 1 to 10, 1 being the lowest rating and 10 being the highest rating. The current state-of-the-art is XLNet based on the transformer architecture for this (Yang et al. 2019) dataset.

Another famous dataset for text classification is Dbpedia dataset which consist of 5.6M labelled texts from the 14 non-overlapping topics of wikipedia. BERT (Devlin et al. 2018) is the current best performing algorithm with only 0.64 error rate on this dataset.

Question Answering Question Answering is a token-level task of NLP where given a piece of information with some question related to it, the system needs to generate an acceptable answer for it. One of the most famous dataset for this task is Stanford's Question Answering Dataset (or SQuAD) which consists of different questions asked by users of wikipedia related to some wikipedia article. The answer to this question is a segment from the same article only. The current state-of-the-arts for this datasets are the ensemble models based on different versions of BERT. An updated leaderboard for this dataset can be found on their website⁴.

3.2 Knowledge Engineering

Initially, the solutions for NLP problems were all based on a common approach of identifying better features to represent the text data. These features are identified based on the domain expertise and are very specific to the problem they are applied in. This category of techniques can be classified as the techniques which requires a lot of domain knowledge in order to achieve good results. They don't need a lot of training data in order to gain good performance but the performance gets saturated after some amount of data, i.e., the performance doesn't increase with the increase in training data. We group these methods into two categories:

⁴<https://rajpurkar.github.io/SQuAD-explorer/>

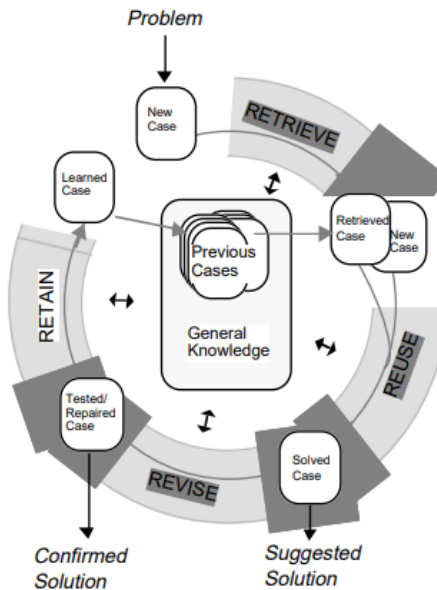


Figure 2: 4Rs of a CBR Cycle (Aamodt & Plaza 1994)

1. Case Based Reasoning
2. Machine Learning

3.2.1 Case Based Reasoning

Case Based Reasoning is a sub-field of Artificial Intelligence which solves the problems on the basis of modelling real word intuition that similar problem have similar solutions. When we see a new problem in our real life, we try to map it to our previous experience of similar problems and solve it taking similar approach. In CBR as well, we retrieve the problems similar to the new one and use the similar kind of approach to solve it using the solutions stored in knowledge base (KB). In the KB, we store these cases in a pair of problem and solution. A CBR cycle generally have 4 steps as defined in (Aamodt & Plaza 1994). Fig. 2 shows the 4Rs of a CBR cycle.

1. RETRIEVE the cases similar to the new problem.
2. REUSE the information or solution for similar cases to solve the new one.
3. REVISE the new proposed solution.
4. RETAIN the useful information of new case in the KB for future use.

k-Nearest Neighbours (k-NN) One of the most common CBR algorithm is k-Nearest Neighbour which computes the distance between the new case and all other previous cases. Then the k cases with the minimum distance are selected as the similar cases and their solution is used to make decision for the new case's solution. k-NN uses the euclidean distance to measure distance between two cases. The distance between two cases c_i and c_j is given in eq. (1)

$$Dist(c_i, c_j) = \sqrt{\sum_{v=1}^n (c_{iv} - c_{jv})^2} \quad (1)$$

where v is the dimension of problem in the case. This technique is also referred as lazy learning sometimes as it makes observation on the previous knowledge every time a new query is given rather than trying to learn a general model for knowledge base in training and using that model to answer a query when asked.

Textual Case-Based Reasoning (TCBR) TCBR is a bit different from general CBR in terms of representation of problem and solution of the cases, although the 4Rs of a CBR cycle holds true here as well. In TCBR, some or whole part of the previous cases are stored in textual form and we aim to solve the problem for new cases with the help of these textual knowledge sources in automated way (Weber et al. 2005). The cases are stored in two sets namely, problem set and solution set. In TCBR, the focus is on defining a good representation of these sets and then finding a way to evaluate these representations.

Text Representation Representing cases in text CBR is a trivial task as there is no standard representation defined that can work for every problem. But one of the most commonly used representation is using bag of words (Brüninghaus & Ashley 2001). In this representation, a sentence is broken into tokens which are essentially the words of the sentence and their appearance in different manner is used as the features. One of the drawback of using this technique is that we lose the sequence information as it is lost during the tokenisation process.

Another common way of representing texts is by grouping them into attribute-value pairs to generate feature vectors, where attributes are decided based on the domain and values are the pieces of text from the case base. This approach works for maximum cases but the attributes differ from problem to problem as they are domain specific. An example of attribute-value pair can be a case storing the information of a student with attributes such as *Student Name - Ashish Upadhyay* and *School - Computing Science and Digital Media*.

There are other examples of case representation as well, like used in popular CBR framework jCOLIBRI where cases are stored in tree structure where each case is treated as an individual capable of containing other individuals and thus forming the tree structure.

Evaluation After the representation of text cases is done, we need to evaluate it by measuring the similarity or alignment between cases problem and solution set. A good representation will provide much better alignment between cases than a bad representation.

Two most common ways of measuring the similarity can be local alignment and global alignment (Raghunandan et al. 2008). The local alignment measure uses the alignment of each case with its neighbourhood and then aggregates the values for all cases to find the alignment for whole case base whereas the global method measures the alignment value for whole case base as one single entity.

Table 2: TF vector representation.

be	black	coloured	covered	must	remain	the	valve	wire
0.166	0.166	0.166	0	0.166	0	0.166	0.166	0
0	0	0	0.166	0.166	0.166	0.166	0	0.166

3.2.2 Machine Learning

Machine Learning is another subset of Artificial Intelligence which learns a general distribution of the previous experience by using the training data and then uses the learned distribution to predict the answer a new problem. In Machine Learning, we mostly use sparse representation of the problems of a case and use statistical algorithms to learn the general distribution of training set.

The difference of machine learning approach from case-based approach is that machine learning, the distribution is learned during the training process and that learning is used to answer a query during test or deployment phase whereas in case-based reasoning, the case-base is searched to retrieve the similar cases every time a query is asked in order to answer it.

Some of the representations and learning algorithms used in machine learning are briefly discussed here. Term Frequency (TF), Term Frequency-Inverse Document Frequency (TFIDF) and Linguistic Features are some of the most common feature representations, whereas Support Vector Machines, Random Forest and Naive Bayes are some of the commonly used learning algorithms.

Term Frequency Vectors This feature vector is used to represent a document by assembling a vector of the frequency of the terms contained in the document. Here we first extract all the terms from the documents and create a corpus dictionary of all the terms. Then the frequency of the terms from each document is calculated to generate the vector representing that document. The vectors can be presented as a document term matrix, where each row is a document vector and each column represent a term from the corpus dictionary.

To calculate the term frequency we apply the following formula:

$$tf(t) = m/k \quad (2)$$

Where, the number of times term t has appeared in a document is represented by m and the total number of terms in the document is denoted by k .

Let's understand this with an example. Suppose, in our training dataset we have N documents. After extracting all the terms from the documents and creating the corpus, we get total M unique terms. So our count vector will be a $N \times M$ matrix where every row is a document and every column is a term from the corpus. So each document is represented as a $1 \times M$ vector.

For example, let's say we have two documents:

D1: The valve must be coloured black.

D2: The wire must remain covered.

Thus the corpus dictionary created would look like this:

['be', 'black', 'coloured', 'covered', 'must', 'remain', 'the', 'valve', 'wire']

So, the count vector for this example would be a 2×10 matrix and can be represented as follows:

Term Frequency - Inverse Document Frequency Vector This feature vector is different from the TF vector in a way where it takes into account the occurrence of a term in whole corpus not just the occurrence of that term in a document. The intuition behind this is to reflect how important a term is to document by giving popular terms a low weight and rare terms a higher weight. The TF-IDF can be divided into two parts: term frequency and inverse document frequency.

For calculating term frequency, we can use the same method discussed above in Equation 1.

Then we calculate the inverse document frequency by using the following formula.

$$idf(t) = \log(N/n) \quad (3)$$

Where n is the number of documents term t has appeared and N is the number of documents.

So, the formula to calculate $tfidf$ is given as:

$$tfidf(t) = tf * idf = (m/k) * \log(N/n) \quad (4)$$

That's why in this way, terms like *the, is, a* are heavily penalized. Rarely occurring terms in a corpus get high idf score. To compare with the TF representation, let's take the same example discussed in previous section. So, the $tfidf$ for term *the* from document $D1$ can be calculated as:

$$\begin{aligned} tf &= 2/8 = 0.25 \\ idf &= \log(2/2) = 0 \\ tfidf &= 0.25 * 0 = 0 \end{aligned}$$

Linguistic Features A sentence can be processed to extract parts of speech, noun chunks, and entities. Some of the features that can be generated afterwards are as follows:

1. word length: number of tokens in sentence.
2. word length 2: number of words in sentences with stopwords removed.
3. has modal: indicates if the sentence contains a modal.
4. num of modals: number of modals in sentence.
5. modal position: average position of modal verbs in a sentence (-1 if none).
6. num of entities: number of named entities in sentence.

Random Forest Random forest (Breiman 2001) is an ensemble algorithm of both multiple decision trees used for classification or regression. It initialises a forest of many random decision trees by applying sampling on the train data. Every tree initialised in the forest uses subset or the random samples of number of observations available as well as only taking the random sample of features from the dataset.

Support Vector Machine A Support Vector Machine (SVM) formally defined by a separating hyperplane is a discriminative classifier that can be used to classify both linear and non-linear data (Vapnik 2013). A Support Vector Machine gives an optimal hyperplane dividing the training examples into different classes. For a multi-class classification where number of classes is more than two, an SVM iteratively finds the optimal plane by dividing the problem into one-vs-all. In every iteration, it takes one class as positive and rest of the classes as negative and finds the hyperplane dividing that data combination. SVM can be very slow for large datasets as the time complexity is proportional to the square of features used.

Naive Bayes Naive Bayes is an algorithm based on Bayes theorem which models the distribution of a class using independent conditional probability (Han et al. 2011). It is very fast and scalable and can be applied to various problems with bigger datasets in text mining.

3.3 Representation Learning

Representation Learning or in common words Deep learning can be considered as a subset of machine learning but has gained attention as a different field due to its success in various vision, text and speech tasks. The difference about deep learning that makes it perform better on various complex problems is that it doesn't require domain expertise to identify the features to represent a problem or case from previous experience. Instead it automatically learns the features during training process using a large set of training data on various iterations.

Initially deep learning outperformed a lot of state-of-the-art algorithms in vision tasks and established itself as the benchmark solutions (Simonyan & Zisserman 2014, He et al. 2016), but during the past few years with the advancement in computing powers and the availability of large datasets the **“The Deep Learning Tsunami”** (Manning 2015) has taken over NLP as well. In this section we will sequentially talk about the development of deep learning applied in NLP to solve the various tasks.

3.3.1 Word Embeddings

When using deep learning and to solve an NLP task, each word in the vocabulary is represented in a vector form so that it can be fed into the neural networks. Word embedding is the way of converting a word w_i in vocabulary V in the form of a vector of n dimensions. These word vectors are generated by unsupervised training on a large corpus of words in order to gain the semantic similarities between the words. Algorithms like word2vec (Mikolov et al. 2013) and GloVe (Pennington et al. 2014) are used to train these word embeddings. These word embeddings are generally pre-trained and made available to be downloaded from the internet and used directly in the deep learning models.

These distributed representation of words give a certain amount of semantic understanding of the words in a high dimensional vector space. For example, the distance between words *king* and *queen* will be similar to the distance between *boy* and *girl*. In this contrast the eq. (5) fits perfectly.

$$boy - girl + king = queen \quad (5)$$

These representations have a drawback that they fail to take context of a word into account. For example, the word 'Scotland' will have a different meaning in the sentence 'Scotland is one of the most best places to live on earth' than in the sentence 'Royal Bank of Scotland is one of the top banking firms in the UK'. The glove or word2vec word embedding will fail to differentiate the two meanings of Scotland and will assign same vector in both the cases. These drawbacks are tackled by a new concept of contextual word embeddings discussed in section 3.3.7

3.3.2 Language Models

A language model is a type of system that predicts the probability of possible next words given a sequence of words as the input. In general terms, for a given sequence of input (x_1, x_2, \dots, x_t) , the probability distribution of next term x_{t+1} is computed from a vocabulary V of k words ($V = (w_1, w_2, \dots, w_k)$).

$$P(x_{t+1}|x_1, \dots, x_t) \quad (6)$$

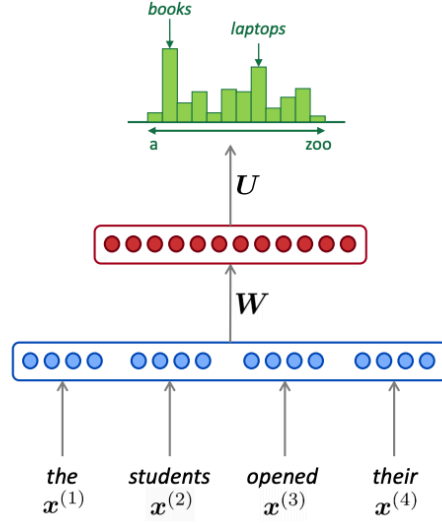


Figure 3: Prediction of next word using a Language Model

Earlier the language models were based on statistical approaches where they used to take a window of n words as a context from the sentence to predict the next word. This approach is also known as *ngram* Language Model. It takes a simple approach of calculating the conditional probability of next word in the sentence given the window of n words as a context. These *ngram* probabilities are calculated from counting them in some large corpus of text.

$$P(x_{t+1}|x_1, \dots, x_t) = \frac{P(x_{t+1}, x_t, \dots, x_{t-n+2})}{P(x_t, \dots, x_{t-n+2})} \quad (7)$$

or in simpler terms:

$$P(w_3|w_1, w_2) = \frac{\text{count}(w_1, w_2, w_3)}{\text{count}(w_1, w_2)} \quad (8)$$

This statistical approach has mainly two problems. First sparsity, consider the eq. (8), what if w_1 , w_2 and w_3 never occurred together. The probability of w_3 will be 0. Also, if w_1 and w_2 then there's no way to calculate the probability of w_3 . Second storage, as we increase the value of n , the count of all *ngrams* we see in the corpus increases as well and so does the need of memory to store them.

The neural language model using Recurrent Neural Networks (section 3.3.3) were able to model all the words in a sentence without having a need of window to predict the next word at each timestep using hidden state and output from the previous time step combining with the new input at each time step.

3.3.3 Recurrent Neural Networks

Text is a sequential form of data. To process text and extract information we need a model capable of processing the sequential data. *Recurrent Neural Networks* (Elman 1990) are the most elementary deep learning architectures that are able to learn from sequential data. RNN, are wide in nature as they unroll through time. These networks will have a 'memory' component which can store the information about

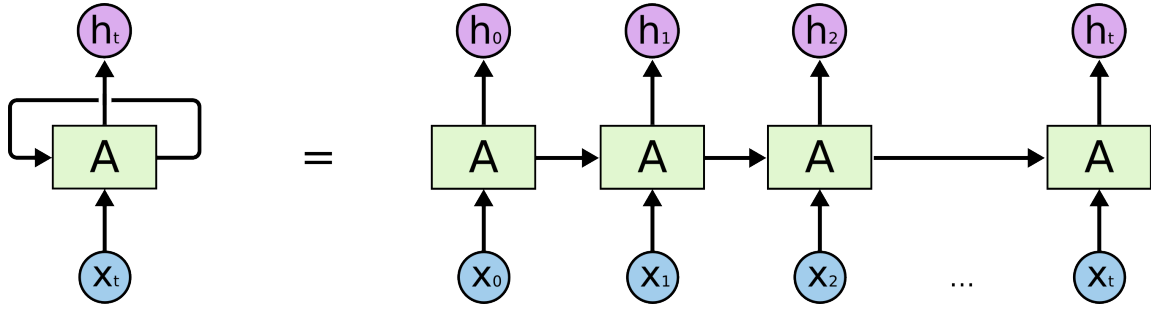


Figure 4: RNN Unrolled

previous state. They share same set of weights throughout the layers, however, will receive a new input at every layer or time-step. The output to every time-step is dependent on the input taken at the current time-step t_i as well as the information gained from previous time-step t_{i-1} . Specifically, an RN Net will maintain a hidden state h_t at every step which is referred as the memory of network. An illustrated diagram of unrolled RNN is shown in fig fig. 4 ⁵.

The operations performed in RNN at every time step is given in the eq. (9).

$$\begin{aligned} h_t &= \sigma_h(W_e x_t + W_h h_{t-1} + b_h) \\ y_t &= \sigma_y(W_y h_t + b_y) \end{aligned} \quad (9)$$

Here σ_y and σ_h are the activation functions. W_h is the weight matrix to apply transformation on previous hidden state h_{t-1} , W_e is the weight matrix to apply transformation on the input x_t received over time t . Combining these with the bias b_h yields hidden state h_t for time t . Applying activation on the h_t with W_y gives the output y_t for every time-step t .

3.3.4 Long Short-Term Memory Networks

Although, in theory the RNNs are designed to handle the sequence input but in practice they lack in storing the long term dependencies because of the problem of exploding and vanishing gradients (Bengio et al. 1994). Long Short-Term Memory Networks (Hochreiter & Schmidhuber 1997) are the advanced version of RNNs with a slight modification of being capable of deciding what to ‘remember’ and what to ‘forget’ from the input sequence with the help of a series of gates. LSTM has a number of gates, an output gate, an input gate i_t , forget gate f_t , o_t , all of which are the functions of previous hidden state h_t and current input x_t . These gates interact with the previous cell state c_{t-1} , the current input x_t , and the current cell state c_t and enable the model to selectively retain or information from the sequence. The full version of LSTM is given in the

⁵taken from <https://colah.github.io/posts/2015-08-Understanding-LSTMs/>

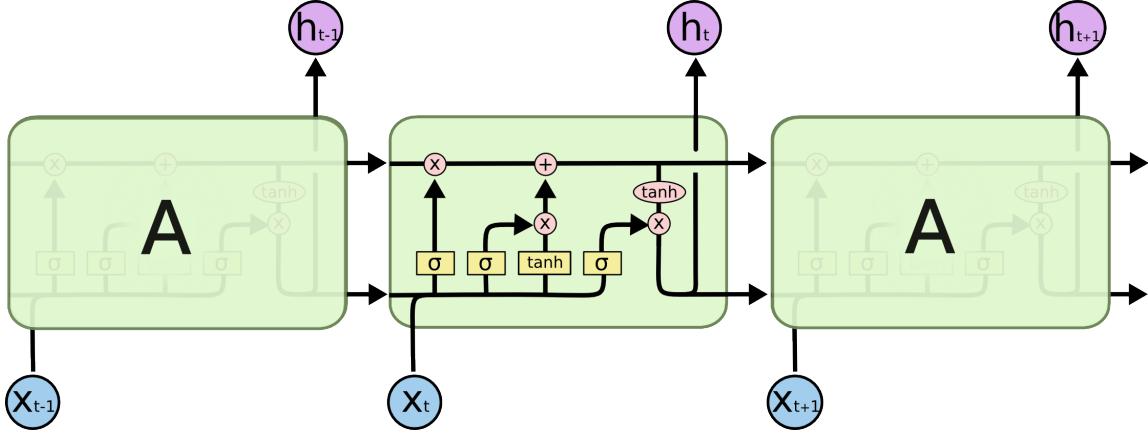


Figure 5: LSTM

eq. (10).

$$\begin{aligned}
f_t &= \sigma_g(W_f x_t + U_f h_{t-1} + b_f) \\
i_t &= \sigma_g(W_i x_t + U_i h_{t-1} + b_i) \\
o_t &= \sigma_g(W_o x_t + U_o h_{t-1} + b_o) \\
\tilde{c}_t &= \sigma_c(W_c x_t + U_c h_{t-1} + b_c) \\
c_t &= f_t \circ c_{t-1} + i_t \circ \tilde{c}_t \\
h_t &= o_t \circ \sigma_h(c_t)
\end{aligned} \tag{10}$$

where σ_g is the *sigmoid* activation function, σ_c and σ_h are the *tanh* activation function, and \circ is element-wise multiplication, also known as ‘*Hadamard product*’. An illustrated diagram of unrolled RNN is shown in fig fig. 5⁶.

LSTM layers can be stacked on each other to form multiple layer LSTM architecture. One of the most popular LSTM architecture is Bidirectional LSTM (BiLSTM) (Graves et al. 2013), where two separate LSTMs are ran forward and backward to gain the sequential information in both directions.

3.3.5 Convolutional Neural Networks

Convolutional Neural Networks (LeCun et al. 1998) are the version of deep neural networks established as state-of-the-art in various computer vision tasks (Barbu et al. 2019, Ali-Gombe & Elyan 2019). After the release of AlexNet (Krizhevsky et al. 2012) in ImageNet competition 2012, CNNs have been the benchmark for almost every vision task. Inspired from the popularity of CNN in vision, researchers proposed an CNN architecture for sentence classification which outperformed many benchmarks on various text classification dataset ranging from sentiment analysis to topic classification (Kim 2014).

CNNs are capable of computing vectors for all possible sub-phrases in a sentence, not just grammatically correct one as done by RNNs. A CNN takes an input sentence of word length n where each word is represented distributively in d dimension. So the input $X_{1:n}$ is a 2D matrix of shape $n \times d$ where $x_i \in \mathbb{R}^d$. Input

⁶taken from <https://colah.github.io/posts/2015-08-Understanding-LSTMs/>

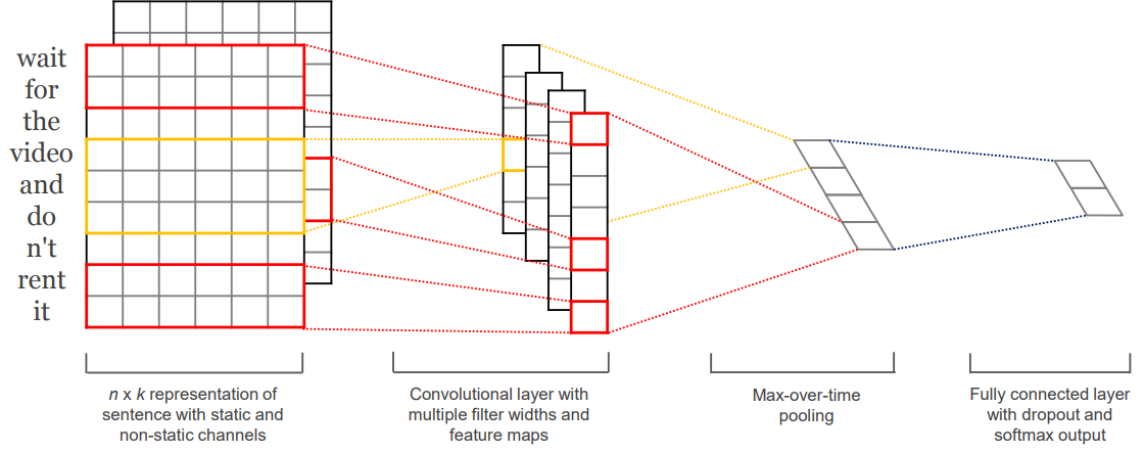


Figure 6: CNN for sentence classification

$X_{1:n}$ can be represented as eq. (11).

$$X_{1:n} = x_1 \oplus x_2 \oplus \cdots \oplus x_n \quad (11)$$

where \oplus is the concatenation operator. On the input layer, convolution filter $W \in \mathbb{R}^{hd}$ is applied over window of h words to generate a new feature. So, a feature c_i is generated from the word window $x_{i:i+h-1}$ with the following operation:

$$c_i = \sigma(Wx_{i:i+h-1} + b) \quad (12)$$

where W is the weight matrix for the connections, σ is the activation function and $b \in \mathbb{R}$ is the bias. Now, this filter is applied to each possible window of words giving an feature map $C \in \mathbb{R}^{n-h+1}$.

$$C = [c_1, c_2, \cdots, c_{n-h+1}] \quad (13)$$

The entries in feature map C are sharing the parameter W , where each $c_i \in C$ is a result of calculation on small segment of the input. Then a *max-pooling* operation is applied on these feature maps to capture the most important part.

$$\tilde{C} = \max(C) \quad (14)$$

This parameter sharing helps the model to incorporate an inductive bias into the model, helping to become learn the location invariant local features. There are k number of filters applied to the input with different window sizes which are then concatenated to form a vector $\mathbf{K} \in \mathbb{R}^k$. Which is then fed to next hidden layer or output layer.

An illustrated diagram of an CNN architecture for text classification is shown in fig. 6⁷.

⁷taken from (Kim 2014)

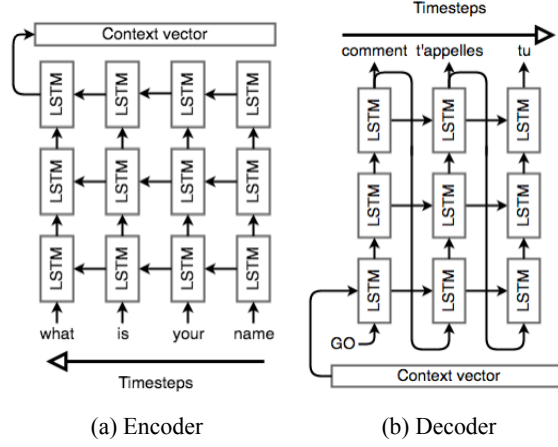


Figure 7: Encoder-Decoder architecture using LSTM for seq2seq model

3.3.6 Sequence to Sequence Models

Most of the NLP tasks require sequential output instead of a single output label unlike classification or regression (Sutskever et al. 2014). These tasks can be Machine Translation of natural language, Question-Answering or Summary generation systems. These systems take a sequence of input and process it to produce yet another sequence for output. The goal is to take a sequence (x_1, x_2, \dots, x_n) as input and map it to another sequence (y_1, y_2, \dots, y_n) as output.

The architecture used to deal with these kind of problems is known as Sequence to Sequence model or in common terms, seq2seq model. It is a combination of auto-encoders and decoders which works in a sequential manner where encoder is an neural architecture to generate a context vector from input sequence and decoder, another neural architecture taking context vector as input and generating the output sequence.

The encoder takes an input X and maps it to fixed size context vector Z using the formula eq. (15)

$$Z = \sigma(WX + b) \quad (15)$$

where σ is activation function. A decoder then maps the context vector Z to a new form of input X' as shown in equation eq. (16)

$$X' = \sigma'(W'Z + b') \quad (16)$$

where σ' is another activation function. The loss is calculated as the squared error between original and reconstructed input as shown in equation eq. (17). An illustrated diagram of seq2seq model is shown in fig. 7⁸.

$$L = ||X - X'||^2 \quad (17)$$

⁸Taken from <http://web.stanford.edu/class/cs224n/>

Attention A problem with general encoder-decoder model is that they give equal importance to all the parts of input sequence. Also, the input sequence is compressed into a single context vector which creates the bottleneck problem, where a long information is tried to be kept into one small representation.

A solution to this problem was proposed in the work (Bahdanau et al. 2014) introducing a new mechanism called Attention. Attention aligns the output at each decoding step to the whole input sequence in order to learn the most important part of the input aligning with the current step output by providing an attentive output.

Let's say we have calculated the encoding hidden states (h_1, h_2, \dots, h_n) for the input sequence (x_1, x_2, \dots, x_n) during the calculation of context vector Z . For a decoder hidden state s_t on timestep t , we get attention score e^t as follows:

$$e^t = [s_t^T h_1, \dots, s_t^T h_n] \quad (18)$$

We take the softmax of these scores to get the attention distribution at timestep t .

$$\alpha^t = \text{softmax}(e^t) \quad (19)$$

The attention output a_t is then calculated as the weighted sum of encoder hidden state using α^t :

$$a_t = \sum_{i=1}^n \alpha_i^t h_i \quad (20)$$

Finally, we concatenate the attention output a_t with decoder hidden state s_t and proceed to calculate the negative log loss same as the non-attention decoder model.

3.3.7 Contextual Word Embeddings

As discussed in section 3.3.1, the word embeddings generated by algorithms like word2vec (Mikolov et al. 2013) and GloVe (Pennington et al. 2014) lacks the contextual awareness and fail to differentiate a word with different sense. For example, word *get* has thirty different senses (meaning) in wordnet⁹ based on the different contexts. But if we use pre-trained word embeddings to generate the vector representation of *get*, it will be same for all the thirty times and we will lose the semantic information.

A new trend of transfer learning came in Deep Learning based NLP with the introduction works like ELMo (Peters et al. 2018) and ULMFiT (Howard & Ruder 2018) where language models are used to learn the nuances of the language grammar and then the pre-trained language model is then fine-tuned on a task specific dataset to achieve better results.

ELMo stands for Embeddings Learned from Language Models (Peters et al. 2018) uses a bidirectional language model to capture the context of a word in a sentence from both sides (left to right and vice-versa). ELMo uses a character-level CNN to convert raw text into a word vector which is then fed into a bidirectional language model. The output of this BiLM is then sent to the next layer of BiLM to form a set of intermediate word vectors. The final output of ELMo is the weighted sum of raw vectors and the intermediate vectors formed from two layers of the BiLMs. The two language models used here are based on LSTM architectures. An illustration of ELMo is shown in fig. 8¹⁰.

⁹<https://wordnet.princeton.edu/>

¹⁰Taken from <https://www.analyticsvidhya.com/blog/2019/03/learn-to-use-elmo-to-extract-features-from-text/>

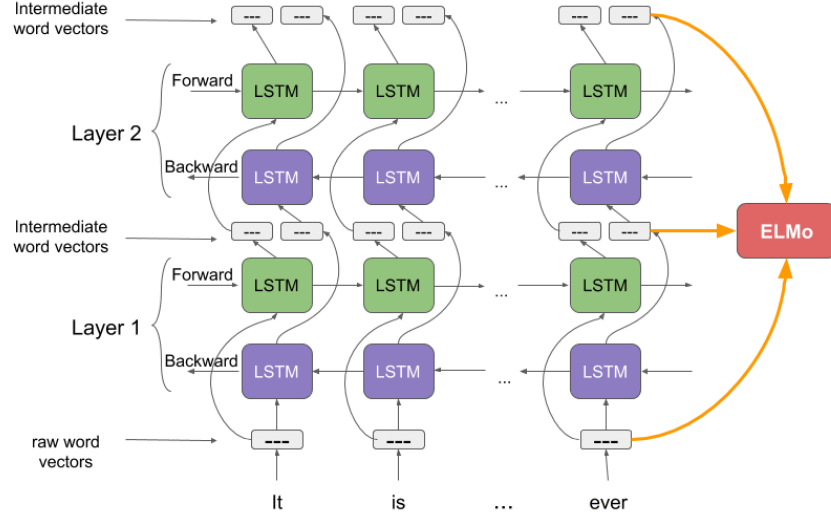


Figure 8: ELMo Vector Representation

ELMo achieved 9% error reduction on the SQuAD (question-answering) dataset compared to then state-of-the-art, 16% on Ontonotes SRL dataset, 10% on Ontonotes coreference dataset and 4% on CoNLL 2003 dataset. It paved a huge path in the success of contextualized word representations for different tasks in NLP.

ULMFiT stands for Universal Language Model Fine Tuning, introduced the way of applying transfer learning on text classification problem (Howard & Ruder 2018). It does so in three main steps: first, train a general domain language model on large corpus of text (mainly Wikipedia); second, fine tune the language model on task specific target dataset; and third, use the again fine tune the fine-tuned language model as classifier by adding a softmax activation on top with target dataset. An illustration of the three steps of ULMFiT is shown in fig. 9.

ULMFiT achieved better results for text classification on six different datasets ranging from topic classification to sentiment analysis. It did so by learning the general rules of grammar from huge corpus of text and then transferring that learning with fine tuning on a task-specific dataset providing better results than state-of-the-arts (Howard & Ruder 2018).

3.3.8 Transformers

Almost all of the models we discussed have recurrent behaviour, which can not be trained parallelly. This imposes a huge problem of time taken to train a model from scratch. In the work (Vaswani et al. 2017), authors proposed a new neural architecture called Transformers which uses a combination of self attention and feed-forward network in its encoder-decoder model and doesn't require any recurrent or convolutional elements. This new seq2seq model was a huge success where it gained better performance on various sequential NLP tasks. To name one, it improved the machine translation performance by 10% on WMT EN-FR and WMT EN-DE datasets. It also reduced the training time by large margin benefiting its non-recurrent nature (Vaswani et al. 2017).

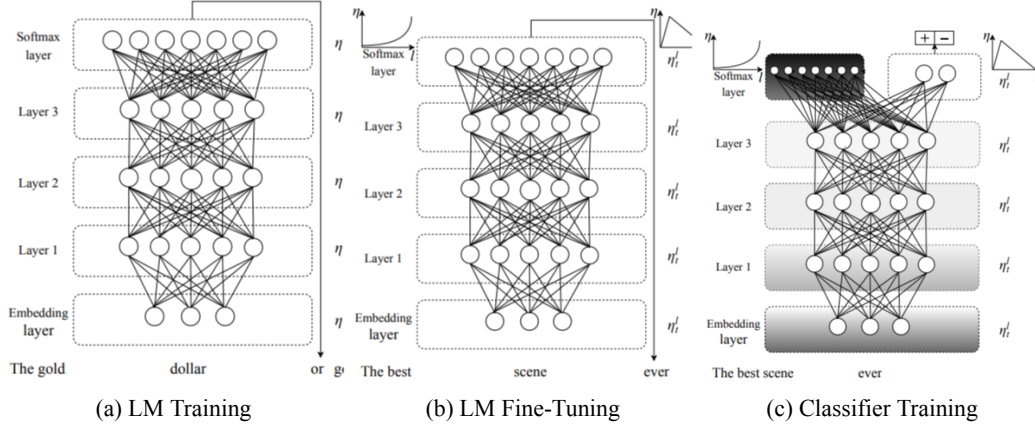


Figure 9: ULMFiT Steps (Howard & Ruder 2018)

The success of transformer architecture paved the way for development of new models to solve the sequential tasks. It helped NLP researchers to utilize its non-recurrent nature in transfer learning where the transformer is used for general pre-training of a language model on large corpus of text which can be used for fine-tuning on domain specific dataset for downstream tasks. An encoder-decoder model of Transformer architecture is shown in the fig. 10.

Generative Pre-Training One of the earliest works in using Transformers for pre-training of language model was presented in (Radford et al. 2018). Following the idea from ELMo (Peters et al. 2018), authors proposed a language model using transformer decoder trained on large corpus and then fine-tuned on task specific dataset. The main difference of GPT from ELMo is that ELMo uses two independent LSTM language models to capture the forwards and backward context whereas in case of GPT, it uses a uni-directional multi-layer transformer language model capable of capturing context due to its attentive nature.

ELMo takes a feature based approach of feeding feature vectors for different tasks into different models, whereas GPT takes a fine-tuning based approach where same language model trained on huge corpus is fine-tuned for downstream tasks without changing the architecture. An illustration of a GPT model used for pre-training is shown in fig. 11 ¹¹.

Bidirectional Encoder Representation from Transformers BERT (Devlin et al. 2018) is another example of the success of transfer learning in NLP. BERT is a bidirectional transformer language model trained on a large text corpus that can be fine-tuned on any domain specific dataset for the downstream tasks like text classification or named entity recognition. BERT mainly differs from other models like GPT and ELMo because of the pre-training tasks used during the unsupervised training of language model. It involves two tasks; first, Masked Language Model (MLM) (Taylor 1953) or prediction of the masked word in a sentence, and second, prediction of next sentence from the corpora.

For the first task of Masked Language Model, let's say we have a sentence 'Boris Johnson is the Prime Minister of UK'. So instead of training for prediction of next word in the sentence as a general Language Model, BERT pre-training replaces 15% of the words with a [MASK] token and learns to predict the correct

¹¹Taken from <https://openai.com/blog/language-unsupervised/>

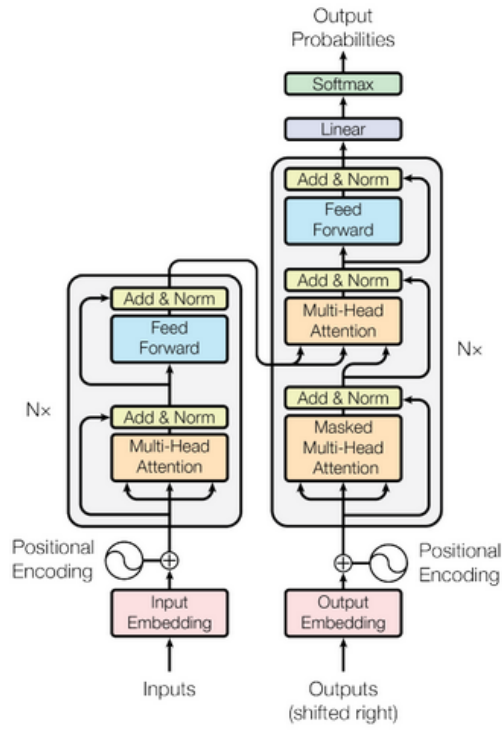


Figure 10: Transformer architecture (Vaswani et al. 2017).

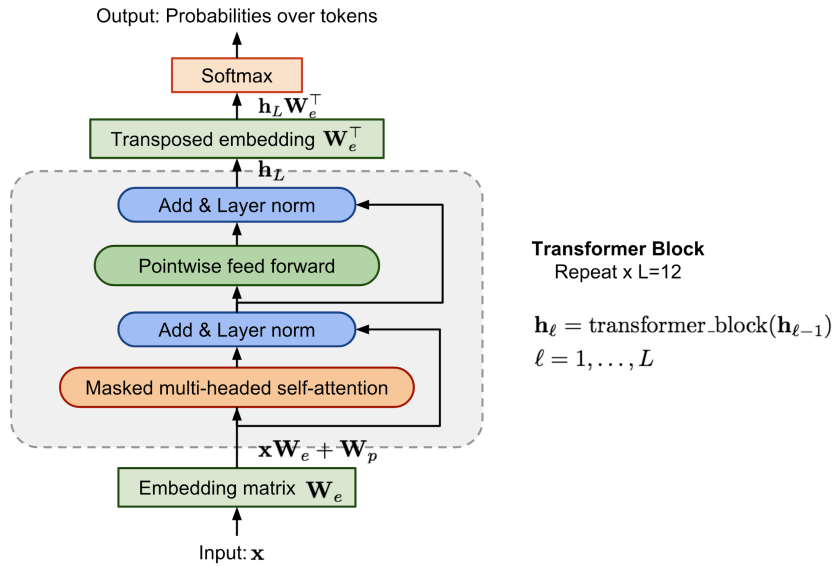


Figure 11: GPT architecture.

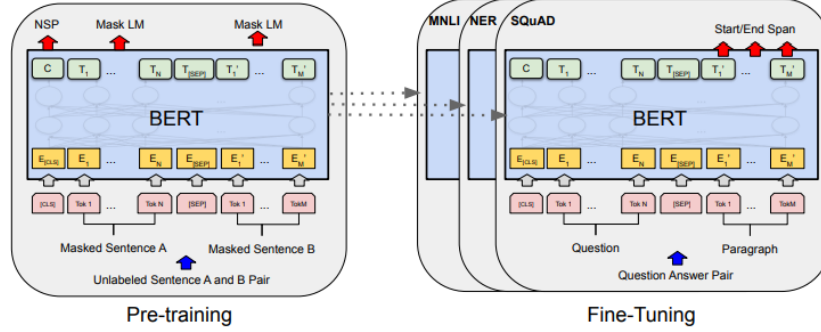


Figure 12: BERT (Devlin et al. 2018)

word at the position of [MASK] token. In the second task of Next Sentence Prediction, the model is trained to learn the relationship between sentences where for a given sentence pair A & B, the model is asked to predict if the sentence B is actually the next sentence that comes after A or not?

BERT improved the fine-tuning based approach of GPT by using a bidirectional transformer for masked language modelling by learning the both left and right context which is a huge improved over GPT's uni-directional approach specially for token-level tasks like Question Answering where the answer depends on both left and right contexts. An illustrated diagram of BERT pre-training and fine-tuning is shown in fig. 12.

4 Proposed Methodology

In order to achieve our objectives we hope to adopt the following methodologies.

4.1 Objective 1

"Identifying the NLP related problems in business processes that are not well addressed by academic research"

Academic research can overly simplify a real-world problem in order to gain good performance. For example, the data used may have training samples which are different to those in real world business process. In real scenarios, getting enough labelled data is usually a problem as the data generation is also a part of the process. We might get very less data during the start of the process or even if the data is available that will be unlabelled. The quality of the data may be poor, and we need to apply pre-processing before we can use that data for any experiment. We will identify these kind of problems which arise during the implementation of machine learning methods in real-world application.

We plan to investigate two approaches for identifying these problems faced by business processes. First, by investigation of literature where we will do extensive review of current state-of-the-art methods from academic research and try to find out where they can fail while implementing them in real-world scenario. Second, by talking to the representatives of different industries to establish at first hand where the key challenges lie in adopting the state-of-the-art methods from academic research in their tasks.

Deliverable: At the end of this phase, we will disseminate our findings in form of review paper or report through journal publications.

4.2 Objective 2

“Identifying the common use-cases from business processes and gathering data for those use-cases”

There are various use-cases in business process like text classification, natural language generation or question answering. We will identify some use-cases that are very common in business processes in terms of text mining throughout the industry. This will again be achieved by either contemplation or by talking to people from industry.

While selecting the use-case, focus will also be on the availability of public datasets that can be used to easily simulate the real-world scenario. We will try to ask some companies if they are willing to share their data in exchange to better methods for their business process.

Deliverable: At the end of this phase, any dataset gathered will be made available to the community through open repositories like UCI ML Repo ¹².

4.3 Objective 3

“Setting up the benchmark performance by implementing state-of-the-art algorithms for the real-world scenario”

We will use public datasets to simulate the real-world scenario and apply the current state-of-the-art. For example, adding timestamps to a pre-processed public datasets can help in simulating a real-world scenario. Then with time-stamps we can simulate the cold-start scenario where initially we will have very less data to start our process and then the labelled data will be generated as a part of the process.

We will use this real-world scenario to test the various state-of-the-art methods from literature review on the use-cases selected from objective 2.

Deliverable: At the end of this phase, we will publish our findings in form of through various workshops and conferences focused on solving NLP related problems in business processes.

4.4 Objective 4

“Developing novel methods to outperform the current state-of-the-art algorithms and evaluating them with real world data on different use-cases”

One of the possible problems can be the unavailability of labelled data for training the learning models. To gain better performance with less training data, we can either develop techniques suitable for working with less training data; or develop techniques for generating the new training data.

¹²<https://archive.ics.uci.edu/ml/index.php>

4.4.1 Reducing the Data Requirement

We can take an approach to develop novel techniques that can give good performance on less data. Chunking the time period based on the availability of labelled data can be one way, where we will use different set of algorithms for based on the amount of labelled data available for training.

In the initial phase, where we have very less amount of labelled data for training we can use rule-based traditional NLP techniques. After some time, when we develop more labelled data we can use feature engineering based machine learning techniques for better performance. After a long time, when we have a huge corpus of labelled data for training then we can deep learning techniques suitable for that amount of training data. A similar kind of initial work is explained in section 5.2.2.

4.4.2 Generating New Training Data

Another way can be augmenting data to increase the amount of training data. Data augmentation in NLP is still not very popular as compared to that in Computer Vision (Guo et al. 2018). The few techniques used in NLP for augmenting text data from literature that can be adopted and extended are;

Back Translation In this method, a sentence is translated from one language to other and then again back translated to the first language. In this process of back translation, some of the words are changed from the original sentence while maintaining the semantics as it is. Similar kind of work is done here (Xie et al. 2019) for consistency training.

Contextual Word Replacement Another good technique for augmentation is randomly replacing some of the words with their synonyms/antonyms with changing the context of the whole sentence. This technique was introduced in the work (Kobayashi 2018) which achieved comparatively better results on various datasets.

GANs Generative Adversarial Networks are very successful in computer vision for new data generation (Ali-Gombe & Elyan 2019). But they are not much investigated in text for the same purpose (Guo et al. 2018). There are few examples like MaskGAN and LeakGAN (Fedus et al. 2018, Guo et al. 2018) which can be tested and improved for our purpose.

Deliverable: At the end of this phase, we will disseminate our findings through different conferences and journals in form of technical papers and will also make our code open-source through GitHub ¹³ or some other open-source code sharing platforms.

5 Initial Work

We have done some initial work for addressing text mining problems in business process in two use cases:

¹³<https://github.com/>

5.1 Natural Language Generation (NLG)

5.1.1 Obituary Generation

In the ongoing Interface funded project of Obituary generation at CSDM school, we are working to generate the obituary of a deceased person based on the details given about the person. We have collected the data of obituaries from one of the most famous obituary website of UK named Funeral Notices ¹⁴. We have selected notices only from the Scotland area and those posted after 2015.

In the initial phase we are taking a Case-Based approach where the problem set is represented with the various features related to deceased person's life, family and funeral details. One of the commencing tasks for this approach is to develop a labelled dataset with all the features marked in the funeral notices.

The work is currently in progress and after the CBR implementation, we are going to take a Natural Language Generation (NLG) approach using deep learning to generate the new text based on the features given as the input to the model.

5.2 Text Classification

5.2.1 Compliance Management

In the recent OGIC funded AZOTH project at CSDM school, we worked on the knowledge extraction and requirement mapping on Oil and Gas regulatory documents. One of the problem statement was to identify if an extracted sentence from a document is a Compliance Requirement (CR) or not? The dataset consists of 2,600 sentences which have been determined as:

1. accept (known CR);
2. postpone (possible CR but requires further review); and
3. reject (known non-CR); with respect to their CR status.

As a result, this is considered a classification problem of three classes. Given the arbitrary nature of the postpone class ('postpone' can be later determined as 'accept' or 'reject'), we modify the CR status to give two class problems. Three variants of CR status for classification are as follows:

1. Accept - Reject - Postpone (3 classes): as in the original dataset as 'accept', 'postpone' and 'reject' respectively.
2. Not reject - Reject (2 classes): 'postpone' is added to the 'accept' class.
3. Accept - Reject (2 classes): 'postpone' is added to the 'reject' class.

We compared three different representation of the text to solve this problem; Linguistic features (section 3.2.2), TF-IDF vectors (section 3.2.2) and Word2Vec Word Embeddings (section 3.3.1). The results are shown in table 3.

¹⁴<http://funeral-notices.co.uk/>

Table 3: Compliance Requirement Verification on different class combinations.

Combination	Representation	Accuracy	Macro F1
All 3 Classes	TF-IDF	67.4%	59.3%
	Word2Vec	74.9%	71.5%
	Linguistic Features	85.0%	69.0%
Reject vs Not-Reject	TF-IDF	89.8%	80.5%
	Word2Vec	91.5%	86.4%
	Linguistic Features	94.0%	92.0%
Accept vs Reject	TF-IDF	87.9%	85.5%
	Word2Vec	90.6%	88.1%
	Linguistic Features	89.0%	88.0%

5.2.2 Weighted Ensemble of Text Classifiers

We have done some work in the direction of developing techniques for suitable for performing with less training data in text classification as well. Here we have used a weighted ensemble of different machine learning classifiers to classify different types of text datasets with number of training examples varying from 5k (Reuters-21578) to 5.6M (Dbpedia). The results are shown in fig. 13 where we have compared our WETC algorithm with various baselines and two deep learning benchmarks BERT and ULMFiT. Newsgroup has 8k of training samples whereas Reuters has 5k and Deception has 1.2k samples only. Ag News and Dbpedia have 1.2M and 5.6M training samples respectively.

From the fig. 13 we can see that our ensemble method is working better for less training data whereas deep learning algorithms are better in case of datasets with large training samples.

6 Identified Impact

This research will introduce new state-of-the-art methodologies for solving the NLP related problems in business processes. Machine learning research has evolved rapidly with the introduction of deep learning techniques build upon the support of tremendous computational power. But still there use in business processes is very few due to the difference between nature of problems in real-world applications and academic research. We will bring attention to the academic research community towards the problems faced by business processes that are not well addressed right now. Our work will help industries make their workforce allocation better as many tasks which do not need much human intervention will be handled in an automated manner. This will free up the human resource of a company for more creative and innovate tasks. Eventually leading in the cost reduction as they have to focus on quality of their workforce instead of quantity.

Right now the work done by few industries to automate their business process is not made available in public domain. There are a few conglomerates who work on developing methods to address some of their problems but they tend not to open-source their findings in order to gain financial profits. The findings from our project will be made public through various publications and code sharing website like GitHub¹⁵ that can help industries with lesser resources to enhance their performance. Instead of using their resources on automating their text related processes they can focus on their core issues & ideas whereas our work can help them in automation.

¹⁵<https://github.com/>

	Newsgroup	Reuters	Deception	AG News	Dbpedia
RF TF	0.9166	0.9396	0.8675	0.8747	0.9636
NB TFIDF	0.9584	0.9182	0.865	0.8777	0.9455
LR TF	0.9431	0.9648	0.845	0.8853	0.9722
RF TFIDF	0.9188	0.9465	0.87	0.8731	0.9639
NB TF	0.953	0.9652	0.875	0.8765	0.9506
LR TFIDF	0.9559	0.9588	0.8775	0.8963	0.9718
Xg TF	0.8774	0.9543	0.8225	0.7898	0.9324
Xg TFIDF	0.8792	0.9492	0.83	0.7888	0.9343
SVM TF	0.9229	0.957	0.83		
SVM TFIDF	0.9571	0.9725	0.8775		
ULMFiT	0.9429	0.9575	0.7875	0.9499	0.9915
BERT	0.9289	0.9548	0.8675	0.923	0.9936
WETC (RF, NB, LR)	0.9605	0.968	0.8775	0.9007	0.974
WETC (RF, NB, LR, XgB)	0.956	0.9648	0.8725	0.8769	0.9706
WETC (RF, NB, LR, SVM)	0.9609	0.9684	0.865		
WETC (RF, NB, LR, SVM, XgB)	0.9586	0.9784	0.88		

Figure 13: Weighted Ensemble of Text Classifiers Result

7 Time Management Plan

Time Management Plan	Year 1				Year 2				Year 3			
Objective 1												
Review literature to identify the research gaps												
Talking to industry representatives for identifying gaps												
Objective 2												
Identifying use-case and gathering data												
Objective 3												
Implementing state-of-the-arts methods on gathered data												
Objective 4												
Proposing new algorithms to outperform the benchmark performance												
Evaluating proposed methods on real-world data												
Final												
Finalizing Thesis Submission												

References

- Aamodt, A. & Plaza, E. (1994), ‘Case-based reasoning: Foundational issues, methodological variations, and system approaches’, *AI communications* **7**(1), 39–59.
- Ali-Gombe, A. & Elyan, E. (2019), ‘Mfc-gan: class-imbalanced dataset classification using multiple fake class generative adversarial network’, *Neurocomputing* **361**, 212–221.
- Bahdanau, D., Cho, K. & Bengio, Y. (2014), ‘Neural machine translation by jointly learning to align and translate’, *arXiv preprint arXiv:1409.0473*.
- Bandhakavi, A., Wiratunga, N., Massie, S. & Luhar, R. (2018), Context extraction for aspect-based sentiment analytics: Combining syntactic, lexical and sentiment knowledge, in ‘International Conference on Innovative Techniques and Applications of Artificial Intelligence’, Springer, pp. 357–371.
- Bandhakavi, A., Wiratunga, N., Padmanabhan, D. & Massie, S. (2017), ‘Lexicon based feature extraction for emotion text classification’, *Pattern recognition letters* **93**, 133–142.
- Barbu, A., Mayo, D., Alverio, J., Luo, W., Wang, C., Gutfreund, D., Tenenbaum, J. & Katz, B. (2019), Objectnet: A large-scale bias-controlled dataset for pushing the limits of object recognition models, in ‘Advances in Neural Information Processing Systems’, pp. 9448–9458.
- Bengio, Y., Simard, P., Frasconi, P. et al. (1994), ‘Learning long-term dependencies with gradient descent is difficult’, *IEEE transactions on neural networks* **5**(2), 157–166.
- Bojar, O., Buck, C., Federmann, C., Haddow, B., Koehn, P., Leveling, J., Monz, C., Pecina, P., Post, M., Saint-Amand, H., Soricut, R., Specia, L. & Tamchyna, A. s. (2014), Findings of the 2014 workshop on statistical machine translation, in ‘Proceedings of the Ninth Workshop on Statistical Machine Translation’, Association for Computational Linguistics, Baltimore, Maryland, USA, pp. 12–58.
URL: <http://www.aclweb.org/anthology/W/W14/W14-3302>
- Bordignon, A., Thom, L., Soares Silva, T., Stein Dani, V., Fantinato, M. & Ferreira, R. (2018), Natural language processing in business process identification and modeling: A systematic literature review.
- Bowman, S. R., Angeli, G., Potts, C. & Manning, C. D. (2015), ‘A large annotated corpus for learning natural language inference’, *arXiv preprint arXiv:1508.05326*.
- Breiman, L. (2001), ‘Random forests’, *Machine learning* **45**(1), 5–32.
- Brüninghaus, S. & Ashley, K. D. (2001), The role of information extraction for textual cbr, in ‘Proceedings of the 4th International Conference on Case-Based Reasoning: Case-Based Reasoning Research and Development’, ICCBR ’01, Springer-Verlag, Berlin, Heidelberg, p. 74–89.
- Budek, K. (2019), ‘A business guide to natural language processing (nlp)’.
URL: <https://deepsense.ai/a-business-guide-to-natural-language-processing-nlp/>
- Cambria, E. & White, B. (2014), ‘Jumping NLP curves: A review of natural language processing research’.
- Devlin, J., Chang, M.-W., Lee, K. & Toutanova, K. (2018), ‘Bert: Pre-training of deep bidirectional transformers for language understanding’, *arXiv preprint arXiv:1810.04805*.
- Elman, J. L. (1990), ‘Finding structure in time’, *Cognitive science* **14**(2), 179–211.

- Fedus, W., Goodfellow, I. & Dai, A. M. (2018), ‘Maskgan: better text generation via filling in the_’, *arXiv preprint arXiv:1801.07736*.
- Graves, A., Jaitly, N. & Mohamed, A.-r. (2013), Hybrid speech recognition with deep bidirectional lstm, in ‘2013 IEEE workshop on automatic speech recognition and understanding’, IEEE, pp. 273–278.
- Guo, J., Lu, S., Cai, H., Zhang, W., Yu, Y. & Wang, J. (2018), Long text generation via adversarial training with leaked information, in ‘Thirty-Second AAAI Conference on Artificial Intelligence’.
- Han, J., Pei, J. & Kamber, M. (2011), *Data mining: concepts and techniques*, Elsevier.
- Han, X., Zhu, H., Yu, P., Wang, Z., Yao, Y., Liu, Z. & Sun, M. (2018), ‘Fewrel: A large-scale supervised few-shot relation classification dataset with state-of-the-art evaluation’, *arXiv preprint arXiv:1810.10147*.
- He, K., Zhang, X., Ren, S. & Sun, J. (2016), Deep residual learning for image recognition, in ‘Proceedings of the IEEE conference on computer vision and pattern recognition’, pp. 770–778.
- Hochreiter, S. & Schmidhuber, J. (1997), ‘Long short-term memory’, *Neural computation* **9**(8), 1735–1780.
- Howard, J. & Ruder, S. (2018), ‘Universal language model fine-tuning for text classification’, *arXiv preprint arXiv:1801.06146*.
- Kim, Y. (2014), ‘Convolutional neural networks for sentence classification’, *arXiv preprint arXiv:1408.5882*.
- Kobayashi, S. (2018), ‘Contextual augmentation: Data augmentation by words with paradigmatic relations’, *arXiv preprint arXiv:1805.06201*.
- Krebs, A., Lenci, A. & Paperno, D. (2018), Semeval-2018 task 10: Capturing discriminative attributes, in ‘Proceedings of The 12th International Workshop on Semantic Evaluation’, pp. 732–740.
- Krizhevsky, A., Sutskever, I. & Hinton, G. E. (2012), Imagenet classification with deep convolutional neural networks, in ‘Advances in neural information processing systems’, pp. 1097–1105.
- LeCun, Y., Bottou, L., Bengio, Y., Haffner, P. et al. (1998), ‘Gradient-based learning applied to document recognition’, *Proceedings of the IEEE* **86**(11), 2278–2324.
- Manning, C. D. (2015), ‘Computational linguistics and deep learning’, *Computational Linguistics* **41**(4), 701–707.
- Marcus, M., Santorini, B. & Marcinkiewicz, M. A. (1993), ‘Building a large annotated corpus of english: The penn treebank’.
- Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S. & Dean, J. (2013), Distributed representations of words and phrases and their compositionality, in ‘Advances in neural information processing systems’, pp. 3111–3119.
- Mukras, R., Wiratunga, N. & Lothian, R. (2007), Selecting bi-tags for sentiment analysis of text, in ‘International Conference on Innovative Techniques and Applications of Artificial Intelligence’, Springer, pp. 181–194.

- Pennington, J., Socher, R. & Manning, C. (2014), Glove: Global vectors for word representation, in ‘Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)’, pp. 1532–1543.
- Peters, M. E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K. & Zettlemoyer, L. (2018), ‘Deep contextualized word representations’, *arXiv preprint arXiv:1802.05365*.
- Radford, A., Narasimhan, K., Salimans, T. & Sutskever, I. (2018), ‘Improving language understanding by generative pre-training’, URL https://s3-us-west-2.amazonaws.com/openai-assets/researchcovers/languageunsupervised/language_understanding_paper.pdf.
- Raghunandan, M., Wiratunga, N., Chakraborti, S., Massie, S. & Khemani, D. (2008), Evaluation measures for tcsr systems, in ‘European Conference on Case-Based Reasoning’, Springer, pp. 444–458.
- Riedel, S., Yao, L. & McCallum, A. (2010), Modeling relations and their mentions without labeled text, in ‘Joint European Conference on Machine Learning and Knowledge Discovery in Databases’, Springer, pp. 148–163.
- Sang, E. F. & De Meulder, F. (2003), ‘Introduction to the conll-2003 shared task: Language-independent named entity recognition’, *arXiv preprint cs/0306050*.
- Silveira, N., Dozat, T., de Marneffe, M.-C., Bowman, S., Connor, M., Bauer, J. & Manning, C. D. (2014), A gold standard dependency corpus for English, in ‘Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC-2014)’.
- Simonyan, K. & Zisserman, A. (2014), ‘Very deep convolutional networks for large-scale image recognition’, *arXiv preprint arXiv:1409.1556*.
- Sintoris, K. & Vergidis, K. (2017), Extracting business process models using natural language processing (nlp) techniques, in ‘2017 IEEE 19th Conference on Business Informatics (CBI)’, Vol. 01, pp. 135–139.
- Sun, C., Qiu, X., Xu, Y. & Huang, X. (2019), How to fine-tune bert for text classification?, in ‘China National Conference on Chinese Computational Linguistics’, Springer, pp. 194–206.
- Sutskever, I., Vinyals, O. & Le, Q. V. (2014), Sequence to sequence learning with neural networks, in ‘Advances in neural information processing systems’, pp. 3104–3112.
- Taylor, W. L. (1953), “‘cloze procedure’: A new tool for measuring readability”, *Journalism Bulletin* **30**(4), 415–433.
- Tschabitscher, H. (2020), ‘19 fascinating email facts’, <https://www.lifewire.com/how-many-emails-are-sent-every-day-1171210>.
- Vapnik, V. (2013), *The nature of statistical learning theory*, Springer science & business media.
- Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł. & Polosukhin, I. (2017), Attention is all you need, in ‘Advances in neural information processing systems’, pp. 5998–6008.
- Weber, R. O., Ashley, K. D. & Brüninghaus, S. (2005), ‘Textual case-based reasoning’, *The Knowledge Engineering Review* **20**(3), 255–260.

- Williams, A., Nangia, N. & Bowman, S. R. (2017), ‘A broad-coverage challenge corpus for sentence understanding through inference’, *arXiv preprint arXiv:1704.05426* .
- Xie, Q., Dai, Z., Hovy, E., Luong, M.-T. & Le, Q. V. (2019), ‘Unsupervised Data Augmentation for Consistency Training’, pp. 1–19.
URL: <http://arxiv.org/abs/1904.12848>
- Yang, Z., Dai, Z., Yang, Y., Carbonell, J., Salakhutdinov, R. & Le, Q. V. (2019), ‘Xlnet: Generalized autoregressive pretraining for language understanding’, *arXiv preprint arXiv:1906.08237* .
- Young, T., Hazarika, D., Poria, S. & Cambria, E. (2018), ‘Recent trends in deep learning based natural language processing [Review Article]’, *IEEE Computational Intelligence Magazine* **13**(3), 55–75.

Appendices

A Vitae Personal Development Planner

myRDF - Action plan - ASHISH UPADHYAY

Your action plan provides an overview of the descriptors you have identified for development, and the current status of the phases you wish to progress within these descriptors. You may wish to share this report with your supervisor or another colleague to inform discussion about your professional development.

	Phase 1	Phase 2	Phase 3	Phase 4	Phase 5
Knowledge and intellectual abilities (A)					
Knowledge base (A1)					
Subject knowledge	Achieved	-		-	
Research methods - theoretical knowledge	Achieved	-	-	-	
Research methods - practical application	Achieved	-	-	-	
Information seeking	Achieved	Achieved		-	
Information literacy and management	Achieved	Achieved	-	-	
Languages	Achieved	-		-	
Academic literacy and numeracy	Achieved	-		-	
Cognitive abilities (A2)					
Analysing	Achieved	-		-	
Synthesising	Achieved	-		-	
Critical thinking	Achieved	-	-	-	
Evaluating	Achieved	-	-	-	
Problem solving	Achieved	-	-	-	
Creativity (A3)					
Inquiring mind	Achieved	-	-	-	
Intellectual insight	Achieved	-	-	-	-
Innovation	Achieved	-	-	-	-
Argument construction	Achieved	-		-	
Intellectual risk	Achieved	-		-	
Personal effectiveness (B)					
Personal qualities (B1)					
Enthusiasm	Achieved		-		-
Perseverance	Achieved		-	-	-
Integrity	Achieved	-	-	-	-
Self-confidence	Achieved	-	-	-	-
Self-reflection	Achieved	-		-	
Responsibility	Achieved	-	-	-	
Self-management (B2)					
Preparation and prioritisation	Achieved	-	-	-	
Commitment to research	Achieved	-	-	-	-
Time management	Achieved	-		-	
Responsiveness to change	Achieved	-	-	-	-
Work-life balance	Achieved	-		-	
Professional and career development (B3)					

myRDF - Action plan - ASHISH UPADHYAY

Your action plan provides an overview of the descriptors you have identified for development, and the current status of the phases you wish to progress within these descriptors. You may wish to share this report with your supervisor or another colleague to inform discussion about your professional development.

	Phase 1	Phase 2	Phase 3	Phase 4	Phase 5
Career management	Achieved	-	-	-	-
Continuing professional development	Achieved	-	-	-	
Responsiveness to opportunities	Achieved	-		-	
Networking	Achieved		-	-	-
Reputation and esteem	Achieved	-	-	-	-
Research organisation and governance (C)					
Professional conduct (C1)					
Health and safety	Achieved	-	-	-	-
Ethics, principles and sustainability	Achieved	-	-	-	-
Legal requirements	Achieved	-	-	-	-
IPR and copyright	Achieved	-	-		-
Respect and confidentiality	Achieved	-	-	-	-
Attribution and co-authorship	Achieved	-	-	-	-
Appropriate practice	Achieved	-	-	-	-
Research management (C2)					
Research strategy	Achieved	-		-	
Project planning and delivery	Achieved	-	-	-	
Risk management	Achieved	-	-	-	-
Finance, funding and resources (C3)					
Income and funding generation	Achieved	-	-		-
Financial management	Achieved	-	-	-	
Infrastructure and resources	Achieved	-	-	-	
Communication, influence and impact (D)					
Working with others (D1)					
Collegiality	Achieved	-	-	-	
Team working	Achieved	-	-	-	
People management	Achieved	-	-	-	
Supervision	Achieved	-		-	
Mentoring	Achieved	-	-	-	
Influence and leadership	Achieved	-	-	-	-
Collaboration	Achieved	-	-	-	
Equality and diversity	Achieved	-	-	-	
Communication and dissemination (D2)					
Communication methods	Achieved	-	-	-	
Communication media	Achieved	-	-	-	-
Publication	Achieved	-	-	-	-

myRDF - Action plan - ASHISH UPADHYAY

Your action plan provides an overview of the descriptors you have identified for development, and the current status of the phases you wish to progress within these descriptors. You may wish to share this report with your supervisor or another colleague to inform discussion about your professional development.

	Phase 1	Phase 2	Phase 3	Phase 4	Phase 5
Engagement and impact (D3)					
Teaching	Achieved	-	-	-	
Public engagement	Achieved	-	-	-	
Enterprise	Achieved	-	-	-	
Policy	Achieved	-	-	-	-
Society and culture	Achieved	-	-	-	
Global citizenship	Achieved	-	-	-	-