RESEARCH ARTICLE OPEN ACCESS

Accelerating Information Retrieval using Natural Language Processing

Vignesh Venkatesh
Software Engineer II
Verizon – India

ABSTRACT

A pivotal aspect that stimulates an **artificial intelligence** application more effectuate is having the ability to understand the intent and presenting the relevant results which plays an imperative role in information retrieval. This paper shows various effective approaches in every layer of Information Retrieval using Natural Language Processing inclined towards higher accuracy and inference and a study of recent advancements in the field of Information Retrieval using Word Embeddings, Machine Learning, Deep Learning and Neural Networks. These approaches apply to natural language understanding, natural language generation, machine translation, feature extraction, image captioning and transfer learning.

Keywords :— Information Extraction, Information Retrieval, Natural Language Processing, Deep Learning, Artificial Intelligence, Knowledge Representation, Machine Learning.

I. INTRODUCTION

Natural Language Processing is used for building systems that can understand language and **Machine Learning** is used for building systems that can learn from experience. In this paper we explain the process involved in building these systems that help solving the problems ranging from text classification, sentiment analysis, auto summarisation, cognitive search, machine translation, conference resolution, question and answering, conversational agents to advanced systems that are intersection of computer vision and natural language processing like image captioning, visual question and answering.

Recently, the approach to build such systems has gone far beyond using information retrieval techniques to machine learning and deep learning which are highly insightful than a usual rule based model or binary retrieval model.

In order to build such systems, we need labelled data, which consists of documents and their proportionate categories, tags or topics.

Most of the NLP problems relate to classification except dialog systems that use natural language interaction that are built using modern neural network based sequence to sequence approach.

The purpose of text classification is to assign documents that contain unstructured data to one or multiple categories. Such categories can be logs, reviews about a product or the language in which the document was typed or any data that is unstructured and need to be analysed.

Accuracy of these systems are proportionate to training data.

Here, we suggest approaches which apply to all layers starting from automatic speech recognition, natural entity recognition, natural language understanding, natural language generation to most recent advancements such as question and answering systems, text summarisation.

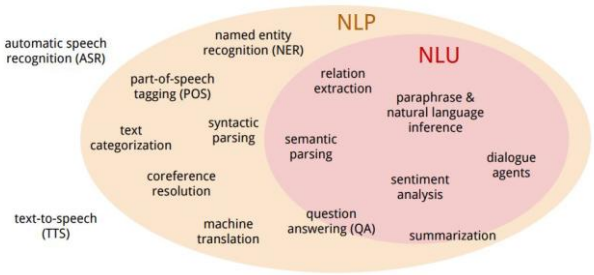

Fig. 1 Layers involved in Information Retrieval using Natural Language Processing and Natural Language Understanding.

II. DATA PREPARATION

Data preparation covers an array of operations, data scientists will use to get their data into a form more appropriate for what they want to do with it.

Training data depends on the use case of the application and it requires datasets that can be reused or custom datasets that has to be prepared for specific type of applications such as conversational agents for conference resolutions or processing and clustering unstructured data into meaningful information using classification.

If you need specialized, custom training data like conversations with some unique specifics for training a chat bot, you would need to create that yourself.

For annotating/generating your own training data, a script to expand the grammar file generated using JSGF can be used.
The Java Speech Grammar Format (JSGF) defines a platform-independent, vendor-independent way of describing one type of grammar, a rule grammar (also known as a command and control grammar or regular grammar). It uses a textual representation that is readable and editable by both developers and computers, and can be included in source code.

A rule grammar specifies the types of utterances a user might say (a spoken utterance is similar to a written sentence). For example, a simple window control grammar might listen for "open a file", "close the window", and similar commands.

What the user can say depends upon the context is the user controlling an email application, reading a credit card number, or selecting a font. Applications know the context, so applications are responsible for providing a speech recognizer with appropriate grammars.

The following is the specification for the java speech grammar format. The grammar header declares the grammar name and lists the imported rules and grammars.

The grammar body defines the rules of this grammar as combinations of utterable text and references to other rules. The examples of grammar declarations are provided below.

```
#JSGF V1.0 ISO8859-1 en;

grammar com.acme.commands;
import <com.acme.politeness.startPolite>;
import <com.acme.politeness.endPolite>;

/**
  * Basic command.
  * @example please move the window
  * @example open a file
  */

public <basicCmd> = <startPolite> <command> <endPolite>;

<command> = <action> <object>;
<action> = /10/ open |/2/ close |/1/ delete |/1/ move;
<object> = [the | a] (window | file | menu);
```

Fig. 2 Text generation using JSpeech Format

III. DATA PRE PROCESSING

Data Pre-processing is a technique that is used to convert the raw data into a clean data set. In other words, whenever the data is gathered from different sources it is collected in raw format which is not feasible for the classification.

The dataset should be formatted in such a way that more than one machine learning and deep learning algorithms are executed in one data set, and best out of them can be chosen.

Most commonly used pre-processing techniques are data cleaning, data annotation, data transformation, missing value imputation, encoding categorical variables, scaling and normalization.

A. **Data Cleaning:**

The process of cleaning the data involves removal of stop words that have no meaningful information that can be used for entity recognition and data classification.

B. **Data Integration:**

This process is applicable for information derived from multiple data sources are utilised for classification. The main purpose is to maintain consistency and remove redundant data from being used for returning the results.

C. **Data Transformation:**

This process involves normalizing, aggregating and generalize the data by converting the data by parsing, stemming and lemmatization. The goal of both stemming and lemmatization is to reduce inflectional forms and sometimes derivationally related forms of a word to a common base form.

Parser is an algorithm for analysing the query in a grammar structure and it establishes the relationship between the words. Parsers play a vital role in many applications such as opinion mining, information extraction, machine translation, question answering. The dependencies derived from the parser can be used to improve the grammar of the sentence.

A popular method for dependency parsing is transition-based parsing and it derives a dependency parse tree by determining a transition sequence from an initial configuration to some terminal configuration. The most probable transition is chosen at every step of parsing based on the current configuration available to the parser.

Stanford Dependency Parser and spacy parser, the fastest and accurate parsers available are transition-based which produce typed dependency parses of natural language sentences. The parsers are powered by a neural network which accepts word embedding inputs from pre-trained models. An example of parsing a sentence using a transition based parser is shown below,

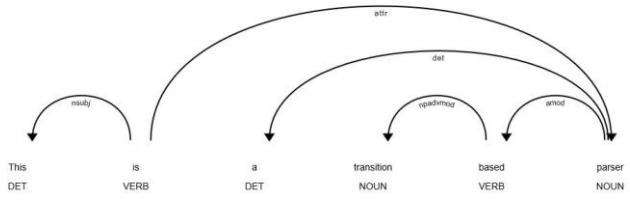

Fig. 3 Transition based parsing

Missing Data is common in all research, and below are the ways to impute the missing data,

Mean Value: Replace the missing value with the mean value for the particular feature.

Regression Substitution: The missing value can be replaced with the historical value from similar data.

Matching Imputation: Each missing value y can be substituted with the value x by determining the similar values or co-occurrence of the values in the data.

Maximum Likelihood is also used to find the missing data.
Deep learning models, work with missing data better than other approaches.

D. **Data Reduction:**

This process involves tasks that tend to reduce the dimensionality of the data and removing information that have less influence on the classification, which results in the data set to be a reduced form.

The size of the resultant dataset becomes smaller after pre-processing thus, enhancing the precision of the data to draw meaningful insights.

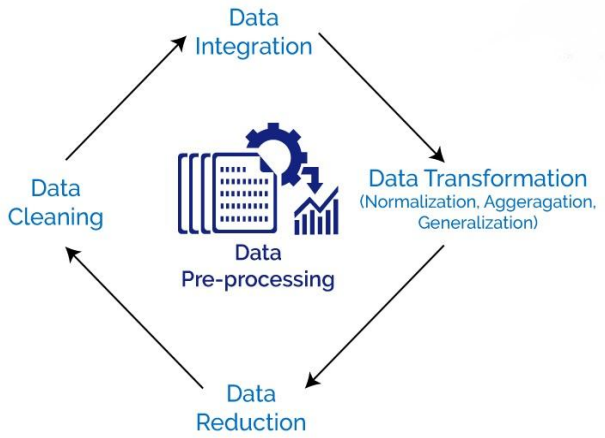

Fig. 4 Phases involved in Data Pre-Processing

The structured data sets are high dimensional i.e. large rows and columns. In a way, text expands the universe of data manifolds. Hence, to avoid long training time, you should be careful in algorithm that scales for text data analysis.

IV. DATA REPRESENTATION

Data Representation involves representing data in an appropriate form for the machine learning algorithms.

Text cannot be used as an input to machine learning algorithms, text must be tokenized and the words need to be encoded as integers or floating point values for use as input to a machine learning algorithm.

Modern machine learning algorithms represent words as vectors, which can be used to find semantics and analogies of the input query. The vectors can be binary, multinomial, or continuous. This improves the performance of the text classifiers. In text classification, each document is considered as an "input" and a class label is the "output" of the algorithm.

Query Expansion plays a vital role in text classification and similarity functions. Queries can be expanded by estimating the distance between the word vectors and it expands the query to more number of instances.

The following techniques have pre-trained vectors which aims at enhancing the process of the understanding the query in different context.

A. **Bag of Words:**

A simple and effective model for text documents in machine learning is called the Bag-of-Words Model, or BoW. It exhibits the order of information in the words and focuses on the occurrence of words in a document.

This can be done by assigning each word a unique number. Then any document can be encoded as a fixed-length vector with the length of the vocabulary of known words. The value in each position in the vector could be filled with a count or frequency of each word in the encoded document.

This model is concerned with encoding schemes that represent what words are present or the degree to which they are present in encoded documents without any information about order.

There are many ways to extend this simple method, both by better clarifying what a "word" is and in defining what to encode about each word in the vector.

Things got much easier with recurrent neural networks and recursive neural networks. With them, you can shift the focus from representing entire sentences and paragraphs as vectors and instead treat them by dealing with individual words iteratively or recursively. You can use an actual tree of word embeddings as a representation.

B. **GloVe:**

Global Vectors, is a model for distributed word representation. The model is an unsupervised learning algorithm for obtaining vector representations for words.

Training is performed on aggregated global word-word co-occurrence statistics from a corpus, and the resulting representations showcase interesting linear substructures of the word vector space.

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

Fig. 5 Co-occurrence probabilities for target words ice and steam with selected context words from a 6 billion token corpus. Only in the ratio does noise from non-discriminative words like water and fashion (https://nlp.stanford.edu/)

C. **fastText:**

Fast text is a Library for efficient text classification and representation learning of word embeddings created by Facebook's AI Research (FAIR) lab.

The skipgram model learns to predict a target word thanks to a nearby word. On the other hand, the c-bow model predicts the target word according to its context and it is represented as a bag of the words contained in a fixed size window around the target word. The model is an unsupervised learning algorithm for obtaining vector representations for words.
Facebook makes available pre-trained models for 294 languages.
The most important parameters of the model are its dimension and the range of size for the sub-words. The dimension (dim) controls the size of the vectors, the larger they are the more information they can capture but requires more data to be learned.

D. **Word2Vec:**

Word2vec is a computationally-efficient predictive model in learning the meaning behind the words. It supports Continuous Bag-of-Words model (CBOW) and the Skip-Gram model.

Algorithmically, these models are similar, except that CBOW predicts target words from source context words, while the skip-gram does the inverse and predicts source context-words from the target words. This inversion might seem like an arbitrary choice, but statistically it has the effect that CBOW over a lot of the distributional information (by treating an entire context as one observation) [14].

For the most part, this turns out to be a useful thing for smaller datasets. However, skip-gram treats each context-target pair as a new observation, and this tends to do better when we have larger datasets.

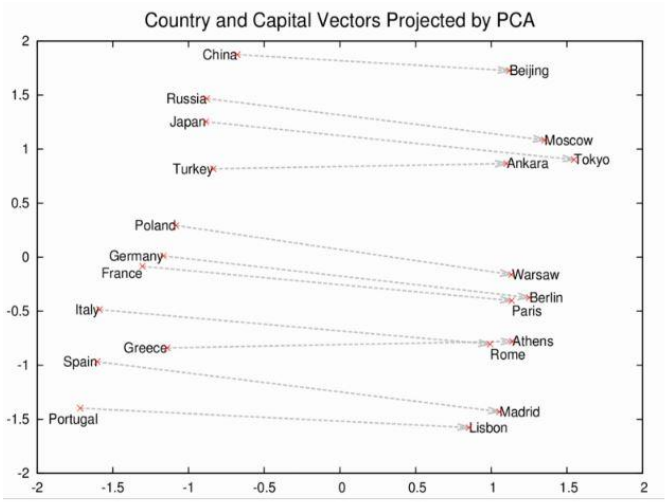

Fig. 6Vectors Projected using PCA [14]

Word2vec word vector representations and t-SNE dimensionality reduction can be used to provide a bird's-eye view of different text sources, including text summaries and their source material. This enables users to explore a text source like a geographical map [14].

For all unique non-frequent words, the vector representation of words are collected from the word2vec model from the gensim Python library. Each word is represented by an N-dimensional vector [14].

Word2vec uses distributed representations of text to capture similarities among concepts. For example, it understands that Paris and France are related the same way Berlin and Germany are (capital and country), and not the same way Madrid and Italy are. This chart shows how well it can learn the concept of capital cities, just by reading lots of news articles -- with no human supervision [1].

This has a very broad range of potential applications: knowledge representation and extraction; machine translation; question answering, conversational systems, and many others.

E. **Visualizing Embeddings:**

Google AI Research has open sourced the tool for visualizing the high dimensional data. The data needed to train machine learning systems comes in a form that computers don't immediately understand. To translate the things we understand naturally (e.g. words, sounds, or videos) to a form that the algorithms can process, we use embeddings, a mathematical vector representation that captures different facets (dimensions) of the data [1].
With the Embedding Projector, you can navigate through views of data in either a 2D or a 3D mode, zooming, rotating, and panning using natural click-and-drag gestures.

PCA is effective at exploring the internal structure of the embeddings, revealing the most influential dimensions in the

data. t-SNE, on the other hand, is useful for exploring local neighborhoods and finding clusters, allowing developers to make sure that an embedding preserves the meaning in the data (e.g. in the MNIST dataset, seeing that the same digits are clustered together).

Custom linear projections can help discover meaningful "directions" in data sets - such as the distinction between a formal and casual tone in a language generation model which would allow the design of more adaptable ML systems [1].

The words visualised are represented using the Vectors from Word2Vec 10K dataset. The standalone projector provides the option to visualize the word embedding using t-SNE, PCA. It can also be customised to compare the vectors in a linear projection. These embeddings can also be visualized by projecting a 3-D text scatter plot using t-sne and textscatter. The clusters can be visualized using a 2-D t-SNE data coordinates.

WordCloud (MATLAB) function adds support for creating word clouds directly from string arrays, and creating word clouds from bag-of-words models and LDA topics.
Below is a figure showing the nearest points to the embedding for a word after training a TensorFlow model using the Word2Vec [1].

The below image represents the word embeddings and nearest points to the word “Research” and clusters of words using 10000 points plotted in 200 dimensions in google’s standalone word embedding projector (http://projector.tensorflow.org/).

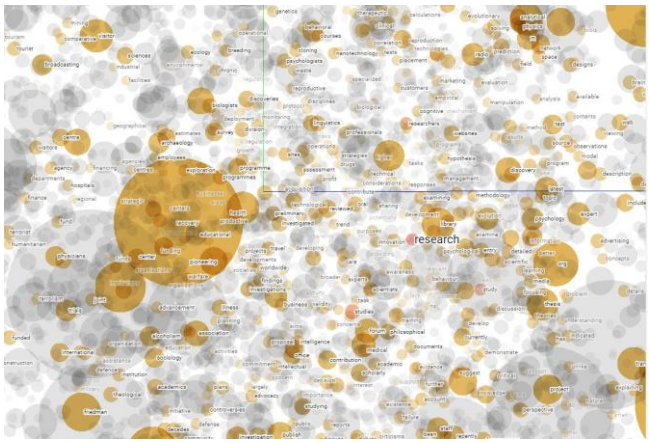

Fig. 7 Visualizing the word embeddings using PCA.

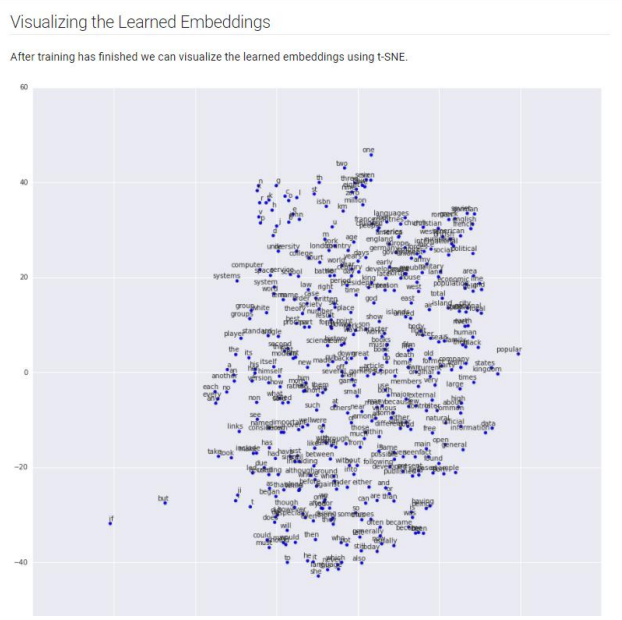

Fig. 8 Visualizing the word embeddings using t-SNE.

V. FEATURE EXTRACTION

Although there are number of NER systems available, MITIE from MIT and spacy NER from explosion.ai work on vectors and achieved higher accuracy for named entity recognition at a large scale.

A. **MITIE :**

MITIE (https://github.com/mit-nlp/MITIE) uses Structural SVM to perform named entity classification [13]. It is a C++ library that provides APIs in C, C++, Java, R and Python (2.7).

It is open-source and has been proven to be in par with Stanford NLP on the Name Entity Recognition task using the CoNLL 2003 corpus (testb). MITIE displayed an F1 score of 88.10% while Stanford NLP 86.31% (https://github.com/mit-nlp/MITIE/wiki/Evaluation).

It is also fast in comparison to other models that attempt to solve the task of named entity recognition. MITIE chunks each sentence into entities and each entity is labelled by a multi-class classifier. In order to classify each chunk, MITIE creates 500K dimensional vector which is the input to the multi-class classifier. The classifier learns one linear function for each class plus one for the “not an entity class” [13].

The feature extraction source code can be found in the ner__feature_extraction.cpp file (https://github.com/mit-nlp/MITIE/blob/master/mitielib/src/ner_feature_extraction.cpp). It uses the Dlib toolkit(https://dlib.net/) which is used in C++ for machine learning[13].

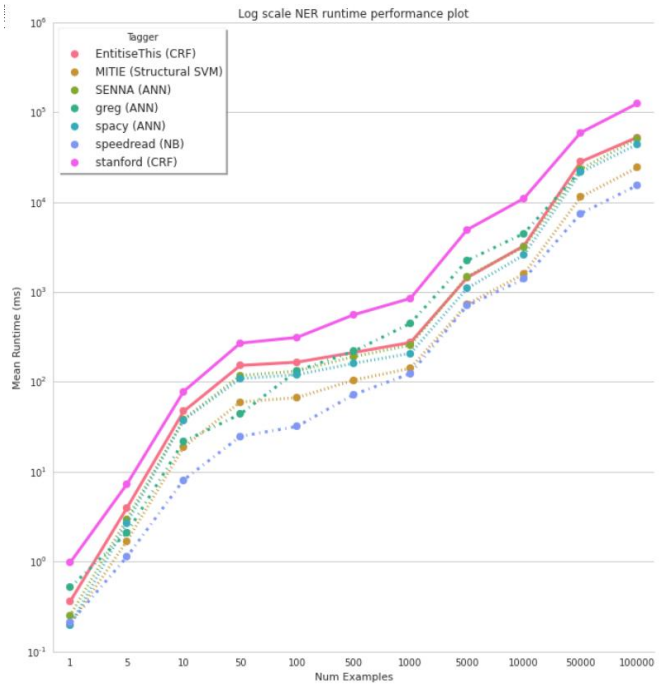

Fig 9 Performance Comparison of various open source libraries providing NER [13].

If one has defined N classes, the classifier has 500K*(N+1) values to learn [13].

B. **spaCy NER :**

spaCy is a complete NLP toolkit written in Python (Cython) (https://github.com/explosion/spaCy).
Many Independent Researchers have confirmed that spaCy is the fastest in the industry.

spaCy is proficient in extracting information at a large scale and scales extremely apex for applications such as search engines, translation systems and applications on the conduct of seq2seq approach.

spaCy features an extremely fast statistical entity recognition system and it assigns labels to spans of tokens and has built in visualizers for determining syntax and NER. Depending on the scale of the application it has models for NER ranging from small, medium and large.

The pre-trained models can be updated to support custom representation of data. The models contain the pre trained vector representation that enhance the entity recognition process. It can recognise various types of named entities in a document, by asking the model for a prediction.

spaCy v2.0 features new neural models for tagging, parsing and entity recognition.

C. **VW Learning2Search:**

John Langford (Microsoft Research) and Hal Daume III (University of Maryland) presented the Learning2Search [2] approach in their tutorial in "Advances in Structured Prediction" in ICML2015 (https://hunch.net/~l2s/merged.pdf). Learning to search method for solving complex joint prediction problems based on learning to search through a problem-defined search space [2].

The major difference of the learning2search (L2S) to the rest of models used in the state-of-the-art is on the way it approaches the task of structured prediction [2].

The majority of the state-of-the-art approaches can be characterised as "global models", having the advantage that they have clean underlying semantics and the disadvantage that they are computationally costly and introduce difficulties in the implementation. On the other hand, L2S treats the problem as a sequential decision making process [2].

Sequential decision making approaches have been recently used in dependency parsing.

The goal for Learning2Search was to create a model that has the following characteristics:

1. Lower programming complexity.
2. Good prediction accuracy.
3. Efficiency in terms of both train and test speed.

The following graph displays a comparison in terms of lines of code between Conditional Random Field approaches (CRFSGD, CRF++) and Structured SVM (S-SVM) and Learning2Search [2].

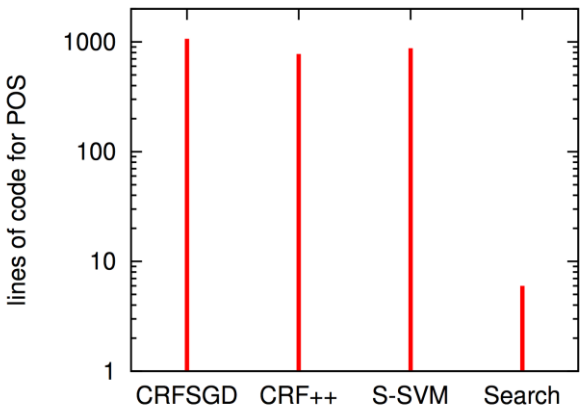

Fig. 10 Programming Complexity of L2S VS State-of-the-art [2]

The following graph displays a comparison in terms of accuracy and training time between Conditional Random Field approaches (CRFSGD, CRF++), Structured

Perceptron(https://en.wikipedia.org/wiki/Structured_prediction), Structured SVM (S-SVM) and Learning2Search [2].

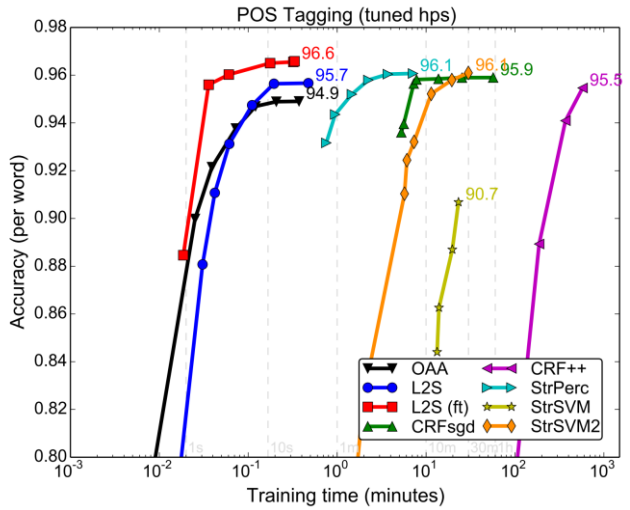

Fig. 11 Training time and accuracy of L2S VS State-of-the-art [2]

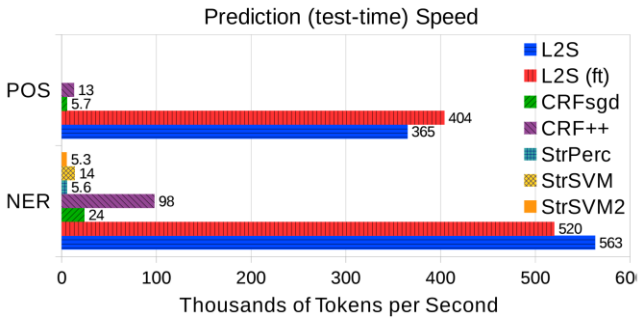

Fig. 12 Training time and accuracy of L2S VS State-of-the-art [2]

From the above mentioned analysis, Learning2Search was the best model. Structured SVM based MITIE also performed better than the traditional methods for feature extraction and classification.

VI. CLASSIFICATION

Classification can be categorised as binary classification, multi-class classification and multi-label classification or clustering. Traditional classification methods were labelled manually, or rule based or statistical based. These methods had limitations of maintaining it and required high computational resources for building these systems and the process was expensive.

Accuracy of classification depends on the algorithm used to train and build a model. Binary or Boolean models are commonly used for exact matches and partial matches. They work based on (AND, OR, NOT) operators for the given query.

Vector space models can be used for extending it to semantic searches. Vector based models project the query in n dimensions, and similarity can be calculated based on the distance between the vector representations, which increases the performance of the classification.

Supervised Learning	Unsupervised Learning
Logistic Regression	Clustering
Naïve Bayes	Hidden Markov Model
Stochastic Gradient Descent	
K-Nearest Neighbours	
Decision Trees	
Random Forest	
Support Vector Machines	

Table. 1 Classification algorithms

The above table explains the classification algorithms categorised as supervised and unsupervised and are explained below,

A. **Supervised Learning:**

1. Logistic Regression:

Logistic Regression is part of a larger class of algorithms known as Generalized Linear Model (glm) and it predicts a binary outcome for a given set of independent variables using probability of occurrence of an event by fitting data to a logistic function.

The fundamental equation of generalized linear model is:

$$g(E(y)) = \alpha + \beta x1 + \gamma x2$$

In the above equation, g() is the link function, E(y) is the expectation of target variable and $\alpha + \beta x1 + \gamma x2$ is the linear predictor (α,β,γ to be predicted). The role of link function is to 'link' the expectation of y to linear predictor.

For measuring the performance of a logistic regression model, Akaike information criteria, residual deviance, confusion matrix, ROC Curve, and likelihood function can be used.

2. Naïve Bayes:

Naïve Bayes represents a supervised learning method as well as a statistical method for classification and is highly scalable. It uses the Bayes Theorem. It predicts membership probabilities for each class such as the probability that given record or data point belongs to a particular class. The class

with the highest probability is considered as the most likely class. This is also known as Maximum A Posteriori (MAP).
Types of Naïve Bayes Algorithms are,

Gaussian Naive Bayes:

When attribute values are continuous, an assumption is made that the values associated with each class are distributed according to Gaussian i.e., Normal Distribution.

Multinomial Naive Bayes:

Multinomial Naive Bayes is preferred to use on data that is multinomially distributed. It is one of the standard classic algorithms which is used in text classification. Each event in text classification represents the occurrence of a word in a document.

Bernoulli Naive Bayes:

Bernoulli Naive Bayes is used on the data that is distributed according to multivariate Bernoulli distributions, multiple features can be there, but each one is assumed to be a binary valued (Bernoulli, boolean) variable. So, it requires features to be binary valued.

3. Stochastic Gradient Descent:

Stochastic gradient descent is a simple and very efficient approach to fit linear models. It is particularly useful when the number of samples is very large. It supports different loss functions and penalties for classification.

In gradient descent, a batch is the total number of examples you use to calculate the gradient in a single iteration. So far, we've assumed that the batch has been the entire data set. The term "stochastic" indicates that the one example comprising each batch is chosen at random.

4. K-Nearest Neighbours:

K-Nearest neighbours can be used for both classification and regression tasks. It is a non-parametric. The model structure is determined from the data and is suitable for distributed data. The output of this classification algorithm is determined by the number of data points or neighbours nearer to the data.

The Implementation of the algorithm involves the following steps,

1. Load the input data.
2. Initialise the value of k.
3. Iterate from 1 to total number of training data points.
4. Calculate the distance between test data and each row of training data using distance between the data points and sort the calculated distances in ascending order based on distance values.
5. Get top k rows from the sorted array.
6. Get the most frequent class of these rows.
7. Return the determined class.

5. Decision Trees:

It works for both categorical and continuous input and output variables. In this technique. the sample is split into two or more homogeneous sets based on most significant differentiator in input variables.

Decision trees are also nonparametric because they do not require assumptions about the distribution of the variables in each class. Decision trees can be of two types,

Categorical Variable Decision Tree: Decision Tree which has categorical target variable then it called as categorical variable decision tree.

Continuous Variable Decision Tree: Decision Tree has continuous target variable then it is called as Continuous Variable Decision Tree.

The decision of making strategic splits heavily affects a tree's accuracy. gini Index, chi-square, information gain and reduction in variance are the algorithms used in decision trees.

The following image describes the visualisation of classification using decision trees.

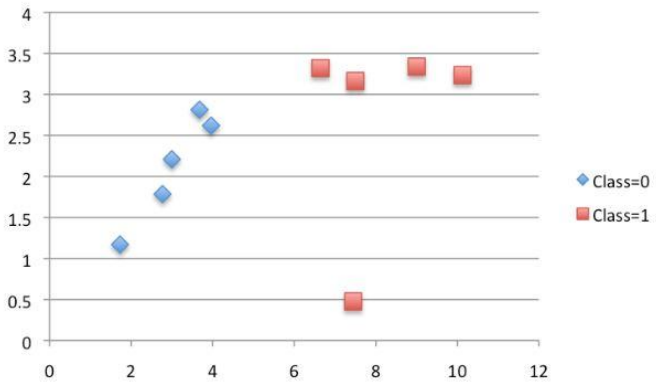

Fig. 13. Visualisation of Classification using decision trees

6. Random Forest:

Random Forest, an ensembled algorithm and versatile machine learning method is capable of performing both regression and classification tasks. Ensembled algorithms are those which combines more than one algorithms of same or different kind for classifying objects. For example, running prediction over Naive Bayes, SVM and Decision Tree and then taking vote for final consideration of class for test object. The most important tuning parameters to be considered for high performance are as follows,

max_features: these are the maximum number of features random forest is allowed to try in individual tree.

n_estimators: If you have built a decision tree before, you can appreciate the importance of minimum sample leaf size. Leaf is the end node of a decision tree. A smaller leaf makes the model more prone to capturing noise in train data. Generally a minimum leaf size of more than 50 is preffered. However, multiple leaf sizes should be tried to find the most optimum.

n_jobs, random_state, oob_score are the other parameters to be considered for tuning the performance of the model.

7. Support Vector Machines:

Support Vector Machine (SVM) is an algorithm defined by a separating hyperplane. The algorithm outputs an optimal hyperplane which categorizes new examples using the trained labelled data. The Hyperplane is a line dividing the plane in a two dimensional space. Kernel, Regularization, Gamma and Margin are the tuning parameters and are described below.

Regularization:

The Regularization parameter tells the SVM optimization how much you want to avoid misclassifying each training example.

For large values of parameter, the optimization will choose a smaller margin hyperplane if that hyperplane does a better job of getting all the training points classified correctly. Conversely, a very small value of parameter will cause the optimizer to look for a larger margin separating hyperplane, even if that hyperplane misclassifies more points.

Kernel:

The learning of the hyperplane in linear SVM is done by transforming the problem using linear algebra. This is where the kernel plays role.

For linear kernel the equation for prediction for a new input using the dot product between the input (x) and each support vector (xi) is calculated as follows:

$$f(x) = B(0) + sum(ai * (x,xi))$$

This is an equation that involves calculating the inner products of a new input vector (x) with all support vectors in training data. The coefficients B0 and ai (for each input) must be estimated from the training data by the learning algorithm.

Gamma:

The gamma parameter defines how far the influence of a single training example reaches, with low values meaning ‘far’ and high values meaning ‘close’. In other words, with low gamma, points far away from plausible seperation line are considered in calculation for the seperation line where as high gamma means the points close to plausible line are considered in calculation.

Margin:

A margin is a separation of line to the closest class points. A good margin is one where this separation is larger for both the classes. A good margin allows the points to be in their respective classes without crossing to other class.

B. **Unsupervised Learning :**

In unsupervised learning, the model is not provided with the correct results during the training. It can be used to cluster the input data in classes on the basis of their statistical properties.

The labelling can be carried out even if the labels are only available for a small number of objects representative of the desired classes and it should be able to generalize and give the correct results when new data are given in input without knowing a prior the target.

The construction of a proper training, validation and test set is crucial for increasing the performance of the algorithm.

1. Clustering:

K - means :

k-means clustering is a method of vector quantization, originally derived from signal processing, that is popular for cluster analysis in data mining.

It aims to partition n observations into k clusters in which each observation belongs to the cluster with the nearest mean, serving as a prototype of the cluster. It is easy to implement and apply even on large data sets.

It has been successfully used in various topics, including market segmentation, computer vision, astronomy and agriculture. It often is used as a pre-processing step for other algorithms [16].

A balance between two variables, the number of clusters and the average variance of the clusters are determined. Learning rate, neighbourhood, radius are some of the parameters to be considered.

k-means and k-medoids clustering partitions data into k number of mutually exclusive clusters. These techniques assign each observation to a cluster by minimizing the distance from the data point to the mean or median location of its assigned cluster, respectively.

The below image describes the visualization of data clusters using k-means.

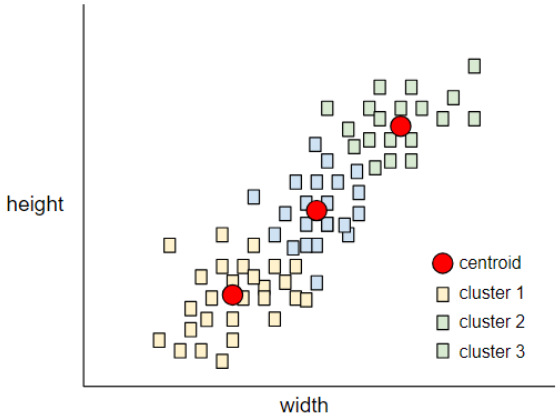

Fig. 14 Clustering using k-means algorithm

Hierarchial Clustering:

Hierarchical clustering (also called hierarchical cluster analysis or HCA) is a method of cluster analysis which seeks to build a hierarchy of clusters. Strategies for hierarchical clustering generally fall into two types,

Agglomerative: This is a "bottom up" approach, each observation starts in its own cluster, and pairs of clusters are merged as one moves up the hierarchy.

Divisive: This is a "top down" approach, all observations start in one cluster, and splits are performed recursively as one moves down the hierarchy [15].

The decision on which clusters to be combined or split is determined by the distance between the sets of observations of the data using dissimilarity functions. Commonly used functions for distance metrics are euclidean distance, squared euclidean distance, manhattan distance, maximum distance, mahalanobis distance.

2. Hidden Markov Models:

Markov processes are examples of stochastic processes that generate random sequences of outcomes or states according to certain probabilities. Markov processes are distinguished by their next state depends only on their current state. Hidden Markov Models (HMM) seek to recover the sequence of states that generated a given set of observed data.

C. **Similarity :**

People would be surprised by the time that actually takes working in a machine learning project doing the actual core machine learning.

Process like collecting data, integration of data, cleaning data and pre-processing the data consumes a significant amount of time and it forecasts the amount of trial and error that have been involved in the feature design.

Also, machine learning is not considered to be a one step process of preparing a data set and running a machine learning algorithm on the data set. It is considered as an iterative process which involves reviewing the results, altering the data, and re-iterating it. Learning takes less amount of time in this process as re-iteration makes it well versed.

Feature engineering is more difficult because it's domain-specific, while learners can be largely general-purpose. However, there is no sharp frontier between the two, and this is another reason the most useful learners are those that facilitate incorporating knowledge. Of course, one of the holy grails of machine learning is to automate more and more of the feature engineering process.

One way this is often done today is by automatically generating large numbers of candidate features and selecting the best by (say) their information gain with respect to the class. But bear in mind that features that look irrelevant in isolation may be relevant in combination.

For example, if the class is an XOR of k input features, each of them by itself carries no information about the class. (If you want to annoy machine learners, bring up XOR.) On the other hand, running a learner with a very large number of features to find out which ones are useful in combination may be too time-consuming, or cause overfitting. So there is ultimately no replacement for the smarts you put into feature engineering.

Below is the list of popular feature engineering methods used:

1. **n-grams**: In a given document, a word that occurs solely like (Person's name, Place name) is 1 gram and likewise can contain (Person's name and Description, Place name and Description) a bi-gram. The idea behind this technique is to explore the chances that when one or two or more words occurs together gives more information to the model.

2. **TF – IDF**: It is also known as Term Frequency - Inverse Document Frequency. This technique believes that, from a document corpus, a learning algorithm gets more information from the rarely occurring terms than frequently occurring terms. Using a weighted scheme, this technique helps to score the importance of terms. The terms occurring frequently are weighted lower and the terms occurring rarely get weighted higher. * TF is be calculated as: frequency of a term in a document / all the terms in the document. * IDF is calculated as: ratio of log (total documents in the corpus / number of documents with the 'term' in the corpus) * Finally, TF-IDF is calculated as: TF X IDF.

3. **Cosine Similarity**: This measure helps to find similar documents. It's one of the commonly used distance metric used in text analysis. For a given 2 vectors A and B of length n each, cosine similarity can be calculated as a dot product of two unit vectors.

4. **Jaccard Similarity**: This is another distance metric used in text analysis. For a given two vectors (A and B), it can be calculated as ratio of (terms which are available in both vectors / terms which are available in either of the vectors). It's formula is: (A ∩ B)/(A U B). To create features using distance metrics, first we'll create cluster of similar documents and assign a unique label to each document in a new column.

5. **Levenshtein Distance**: It creates a new feature based on distance between two strings. It finds the shorter string in longer texts and returns the maximum value as 1 if both the shorter string is found. For example: Calculating levenshtein distance for string "Mountain View 41" and "1st Block, Mountain View 41" will result in 1.

6. **Feature Hashing**: This technique implements the 'hashing trick' which helps in reducing the dimension of document matrix (lesser columns). It doesn't use the actual data, instead it uses the indexes[i,j] of the data, thus it processes data only when needed. And, that's why it takes lesser memory in computation.

7. **Vector Space Model**: This model converts the document or query to a vector in an 'n' dimensional space. Every dimension represents the tf-idf for the particular term. It outperforms in identifying semantically similar information in the documents using word2vec, which has the vectors projected for words to calculate similarity between documents.

VII. ADVANCEMENTS IN INFORMATION RETRIEVAL

Natural Language Processing employs computational techniques for the purpose of learning, understanding, and producing human language content.

Early computational approaches to language research focused on automating the analysis of the linguistic structure of language and developing basic technologies such as machine translation, speech recognition, and speech synthesis.

Researchers refine and make use of such tools in real-world applications, creating spoken dialogue systems and speech-to-speech translation engines, mining social media for information about health or finance, and identifying sentiment and emotion toward products and services.

This paper discusses successes and challenges in this rapidly advancing area.

1. Computational Graph:

Wolfram Alpha is a computational knowledge engine or answer engine developed by Wolfram Alpha LLC, a subsidiary of Wolfram Research. It is an online service that answers factual queries directly by computing the answer from externally sourced "curated data" rather than providing a list of documents or web pages that might contain the answer as a search engine might.

On the data front, Wolfram Alpha uses a very large set of data sources which are verified, tested, and highly structured. The company takes all that data and ensures that it's structured again in such a way that it can be "computable," which leads to the real power of Wolfram Alpha and explains why it's not simply a search engine or a more reliable version of Wikipedia.

Wolfram Alpha doesn't just return information, it analyses and does computation on your inputs and on its own data to provide "reports" instead of just "answers."

Users will also be able to use images as inputs.
Once an image is uploaded, the engine will provide interactive reports with histograms, EXIF data, lists of colours with the HTML identifiers, edge detection, optical character recognition for text, a few image filters and effects, and a bit more [9].

A key feature is the ability to upload many common file types and data including raw tabular data, images, audio, XML, and dozens of specialized scientific, medical, and mathematical formats for automatic analysis [9].

Other features include an extended keyboard, interactivity with CDF, data downloads, in-depth step by step solution, the ability to customize and save graphical and tabular results and extra computation time [9].

2. Knowledge Graph:

The Knowledge Graph enables you to search for things, people or places that Google knows about—landmarks, celebrities, cities, sports teams, buildings, geographical features, movies, celestial objects, works of art and more and instantly get information that's relevant to your query [10].

It's Google's systematic way of putting facts, people and places together, to create interconnected search results that are more accurate and relevant [10].

More specifically, the "knowledge graph" is a database that collects millions of pieces of data about keywords people frequently search for on the World Wide Web and the intent behind those keywords, based on the already available content.

With the knowledge graph, users can get information about people, facts and places that are interconnected in one way or the other as shown in the image below [10].

To make your learning easier, just go to Google and search for "what is the knowledge graph?" The answer is displayed right there – and that's also what the Knowledge Graph does [10].

Fig. 15 Search Results using **Knowledge Graph**

Some of the potential applications include: semantic search, automated fraud detection, intelligent chat bots, advanced drug discovery, dynamic risk analysis, content-based recommendation engines and knowledge management systems [10].

3. Deep Learning:

In previous years, Natural Language Processing was considered extinct from other fields in terms of adopting deep learning for applications.

Text does not have a spatial feature space suitable for convolutional nets, nor is it entirely unstructured as it is already encoded via commonly understood vocabulary, syntax and grammar rules and other conventions. However, this has changed over the last few years, thanks to the use of RNNs, specifically LSTMs, as well as word embeddings.

Deep learning models often require significant amounts of computational resources, memory and power to train and run, which presents an obstacle if you want them to perform well on mobile. Learn2Compress enables custom on-device deep learning models in Tensor Flow Lite that run efficiently on mobile devices, without developers having to worry about optimizing for memory and speed [1].

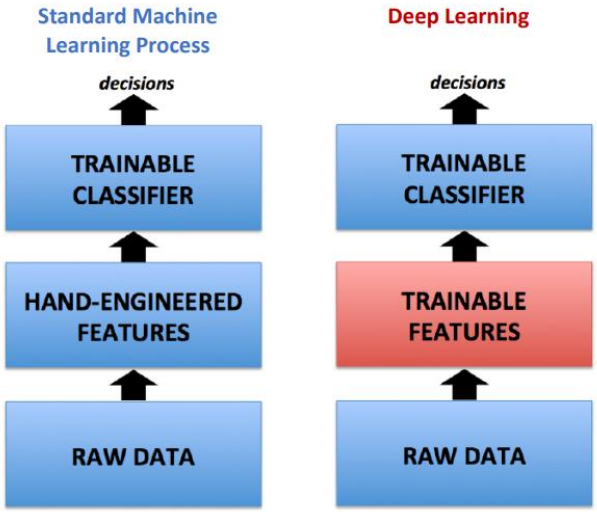

Fig. 16 Standard machine learning process and deep learning based process in Natural Language Processing [11].

A. Neural Networks

1. LSTM (Long Short Term Memory):

Two commonly-used variants of the basic LSTM architecture are the Bidirectional LSTM and the Multilayer LSTM (also known as the stacked or deep LSTM) [3]. The mechanism followed in extending LSTM's are as follows,

1. Adding a forgetting mechanism. If a scene ends, for example, the model should forget the current scene location, the time of day, and reset any scene-specific information; however, if a character dies in the scene, it should continue remembering that he's no longer alive. Thus, we want the model to learn a separate forgetting/remembering mechanism: when new inputs come in, it needs to know which beliefs to keep or throw away [5].

2. Adding a saving mechanism. When the model sees a new image, it needs to learn whether any information about the image is worth using and saving [5].

3. When a new input comes in, the model first forgets any long-term information it decides it no longer needs. Then it learns which parts of the new input are worth using, and saves them into its long-term memory [5].

4. Focusing long-term memory into working memory.

Finally, the model needs to learn which parts of its long-term memory are immediately useful. For example, Bob's age may be a useful piece of information to keep in the long term (children are more likely to be crawling, adults are more likely to be working), but is probably irrelevant if he's not in the current scene. So instead of using the full long-term memory all the time, it learns which parts to focus on instead [5].

This, then, is a long short-term memory network. Whereas an RNN can overwrite its memory at each time step in a fairly uncontrolled fashion, an LSTM transforms its memory in a very precise way: by using specific learning mechanisms for which pieces of information to remember, which to update, and which to pay attention to. This helps it keep track of information over longer periods of time [5].

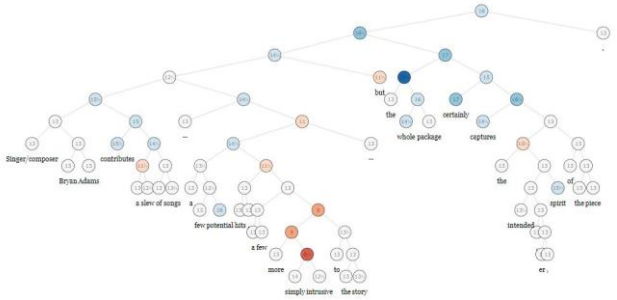

Fig. 17 Recursive neural networks applied on a sentence for sentiment classification at every node of a parse tree and capturing the negation and its scope in this sentence [4].

2. seq2seq:

It is a general end-to-end approach used in conversational applications to sequence learning that makes minimal assumptions on the sequence structure. The method uses a multilayered Long Short-Term Memory (LSTM) to map the input sequence to a vector of a fixed dimensionality, and then another deep LSTM to decode the target sequence from the vector.

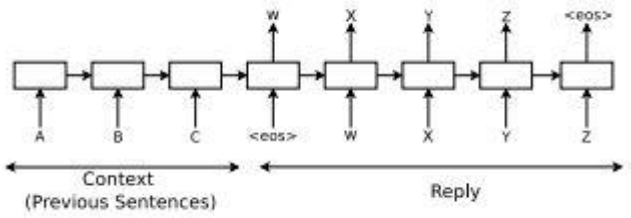

Fig. 18 Seq2Seq Learning and Neural Conversational Model

B. **Deep Neural Networks:**

In this section, we review recent research on achieving this goal with variational auto encoders (VAEs) and generative adversarial networks (GANs) [6]. Standard sentence auto encoders, as in the last section, do not impose any constraint on the latent space, as a result, they fail when generating realistic sentences from arbitrary latent representations [6].

The representations of these sentences may often occupy a small region in the hidden space and most of regions in the hidden space do not necessarily map to a realistic sentence [6].

They cannot be used to assign probabilities to sentences or to sample novel sentences. The VAE imposes a prior distribution on the hidden code space which makes it possible to draw proper samples from the model [6].

It modifies the auto encoder architecture by replacing the deterministic encoder function with a learned posterior.

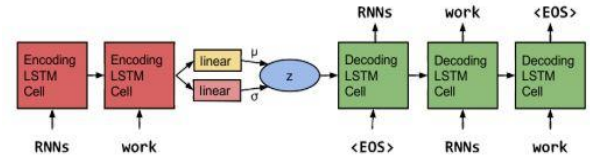

Fig. 19 RNN-based VAE for sentence generation (Figure source: Bowman et al. [6]).

Smart Compose, a neural network based model for composing email is a new feature in Gmail that uses machine learning to interactively offer sentence completion suggestions as you type, allowing you to draft emails faster [1].

Building upon technology developed for Smart Reply, Smart Compose offers a new way to help you compose messages — whether you are responding to an incoming email or drafting a new one from scratch [1].

It uses a combination a BoW model with an RNN-LM, which is faster than the seq2seq models with only a slight sacrifice to model prediction quality. In this hybrid approach, we encode the subject and previous email by averaging the word embeddings in each field [1].

It is then joined with averaged embeddings, and feeded to the target sequence RNN-LM at every decoding step, as the model diagram below shows [1].

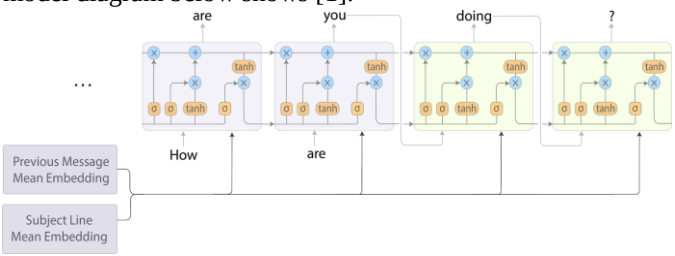

Fig 20 RNN-LM model architecture [1]

Subject and previous email message are encoded by averaging the word embeddings in each field. The averaged embeddings are then fed to the RNN-LM at each decoding step [1].

C. **Transfer Learning:**

Transfer Learning, a process of re-using the existing or pre-trained information for a current research is very popular in the field of Deep Learning. It can be used to train deep neural networks with comparatively less amount of data. This is very useful since most real-world problems typically do not have millions of labelled data points to train such complex models.

Recently, deep neural networks are emerging as the prevailing technical solution to almost every field in NLP. Although capable of learning highly nonlinear features, deep neural networks are very prone to overfitting, compared with traditional methods and therefore becomes even more important. Fortunately, neural networks can be trained in a transferable way by their incremental learning nature.

We can directly use trained (tuned) parameters from a source task to initialize the network in the target task. Alternatively, we may also train two tasks simultaneously with some parameters shared.

As neural networks are usually trained incrementally with gradient descent (or variants), it is steadfast to use gradient information in both source and target domains for optimization so as to accomplish knowledge transfer [7].

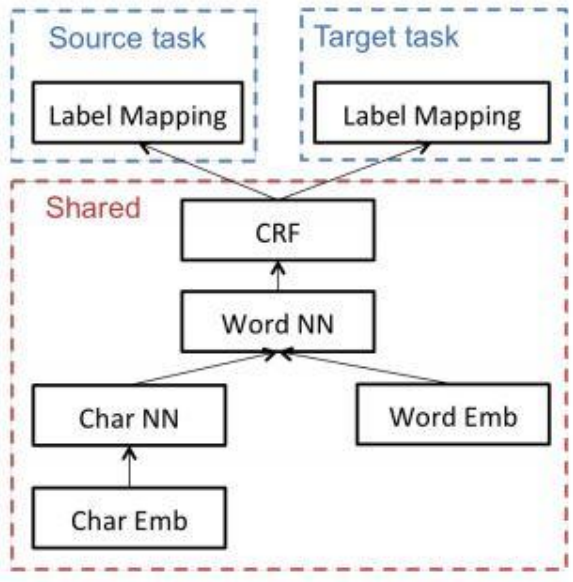

Fig. 21 Transfer model T-A: used for cross-domain transfer where label mapping is possible.: "Char NN" denotes character-level neural networks, "Word NN" denotes word-level neural networks, "Char Emb" and "Word Emb" refer to character embeddings and word embeddings respectively. [8]

Depending on how samples in source and target domains are scheduled, there are two main approaches to neural network-based transfer learning, MULT (Multi-task learning) and INIT. But their performance depends on the below factors.

Whether a neural network is transferable in NLP depends largely on how semantically similar the tasks are, which is different from the consensus in image processing [7].

The output layer is mainly specific to the dataset and not transferable. Word embeddings are likely to be transferable to semantically different tasks [7].

Transfer Learning is used widely in computer vision because training the systems is in-evident

VIII. TRAINING

The goal of the training is to minimize the total loss of model, but for evaluating model performance, we only look the loss of the main output.

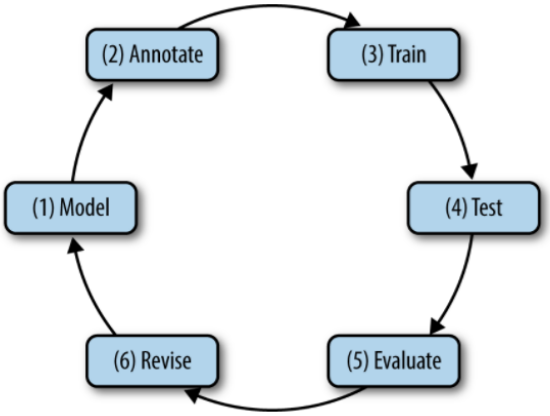

Fig. 22 Process involved in attaining a accurate model

It's crucial to train the model the with configurable parameters like number of training data to be considered and number of epochs, the model is trained in epochs, where the model sees all the input data at least once to determine the probability of the class of the query.
The number of epochs varies depending on the quality of dataset and has to be experimented number of times to finalise a model.

The steps involved in modelling, however, vary greatly, depending on the nature of the task you have defined for yourself. Essentially, the goal of designing a good model of the phenomenon (task) is that this is where you start for designing the features that go into your learning algorithm. The better the features, the better the performance of the algorithm.

To determine how well a natural language processing system performs, you also need labelled data. So the annotated texts are usually divided into 3 subsets:

A. Training Set:

Data used to train the ML algorithm. The developer may also look at this data to help design the system. This is usually the largest subset.

B. Tuning Set:

Data set aside to assess how well the program performs on unseen data and/or to set parameters. Helps to minimize overfitting.

C. Test Set:

Data set aside to perform a final evaluation of how well the program performs on new data. The developer should never look at these data.

Performance Measures

Some common performance measures are:

Accuracy: the percentage of instances assigned a correct label

Recall: for a specific category C, the percent of true instances of C that are correctly labeled ($\frac{\#\ correctly\ labeled\ as\ C}{\#\ true\ instances\ of\ C}$)

Precision: for a specific category C, the percent of instances assigned the label C that are correctly labeled ($\frac{\#\ correctly\ labeled\ as\ C}{\#\ labeled\ as\ C}$)

F Score: the harmonic mean of Recall and Precision

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

Fig. 23 Parameters to be considered for computing accuracy

Annotated Training:

Machine Learning Models can be improvised and attained using interactive mode of building it. **Prodigy** provides you with an interface to do it with ease. Public research datasets used for benchmarking the performance of natural language processing and computer vision algorithms are labelled by visually training it using an interface. This makes the process of training the model using unstructured data more insightful.

Although most of the services for conversational agents offered by cloud providers have the interface for training and defining the models and entities respectively, Open source tools don't extend a visual interface to train the models. The below tool allows the data engineers to annotate data using an interface.

Prodigy:

Prodigy is an annotation tool so efficient that data scientists can do the annotation themselves, enabling a new level of rapid iteration. Whether you're working on entity recognition, intent detection or image classification, Prodigy can help you train and evaluate your models faster. Stream in your own examples or real-world data from live APIs, update your model in real-time and chain models together to build more complex systems. (https://prodi.gy/) [12].

It can be used to extend the pre-trained models and also helps in annotating the data from image segmentation and object detection. The trained model can be exported as a versioned python package for deployment in production.

Prodigy fabricates the model building and testing process less enervated using reinforcement learning.

An example of training a model using prodigy can be found in (https://prodi.gy/demo?view_id=ner).

IX. CONCLUSION

This paper evinces approaches experimented in the industry for tasks involved in information retrieval from parsing to part of the speech tagging, classification, machine translation, transfer learning, question and answering systems and auto summarisation using natural language processing and deep learning. It also explains recent advancements in the research areas of information retrieval.

REFERENCES

[1] Google AI Research Blog https://ai.googleblog.com

[2] Elman, J. L. Finding structure in time. Cognitive science 14, 2 (1990), 179–211.

[3] K. S. Tai, R. Socher, and C. D. Manning, "Improved semantic representations from tree-structured long short-term memory networks," arXiv preprint arXiv:1503.00075, 2015.

[4] R. Socher, A. Perelygin, J. Y. Wu, J. Chuang, C. D. Manning, A. Y. Ng, C. Potts et al., "Recursive deep models for semantic compositionality over a sentiment treebank," in Proceedings of the conference on empirical methods in natural language processing (EMNLP), vol. 1631, 2013, p. 1642.

[5] Edwin Chen's Blog : http://blog.echen.me/2017/05/30/exploring-lstms/

[6] S. R. Bowman, L. Vilnis, O. Vinyals, A. M. Dai, R. Jozefowicz, and S. Bengio, "Generating sentences from a continuous space,"
arXiv preprint arXiv:1511.06349, 2015.

[7] Lili Mou, Zhao Meng, Rui Yan, Ge Li, Yan Xu, Lu Zhang, Zhi Jin.
"How Transferable are Neural Networks in NLP Applications."

[8] Zhilin Yang, Ruslan Salakhutdinov & William W. Cohen "Transfer Learning for sequence tagging with hierarchial recurrent networks"
http://www.cs.cmu.edu/~wcohen/postscript/iclr-2017-transfer.pdf

[9] Wolfram Alpha:
http://www.technologyreview.com/featuredstory/414017/search-me/

[10] Google Knowledge Graph :
https://www.google.com/intl/bn/insidesearch/features/search/knowledge.html

[11] Deep Learning for Natural Language Processing And Machine Translation :
https://pdfs.semanticscholar.org/presentation/297b/297951be2dde56bd0526f6fd1665fe3ada37.pdf

[12] Radically efficient machine teaching,
An annotation tool powered by active learning.
https://prodi.gy/

[13] https://booking.ai/named-entity-classification-d14d857cb0d5

[14] https://opensource.googleblog.com/2013/08/learning-meaning-behind-words.html

[15] Rokach, Lior, and Oded Maimon. "Clustering methods." Data mining and knowledge discovery handbook. Springer US, 2005. 321-352.

[16] Honarkhah, M; Caers, J (2010). "Stochastic Simulation of Patterns Using Distance-Based Pattern Modeling". Mathematical Geosciences. 42 (5): 487–517. doi:10.1007/s11004-010-9276-7.