**Capstone Project - AIML**

**(2019-20)**

**Automatic Ticket Assignment**

**Mentored By: Naga Pavan Kumar Kalepu**

**Prepared By:**

* Shivalik Chhabra
* Nishant Joshi
* Srikanth Rammnath
* Mansi Garg

Summary of problem statement, data and findings

**PROBLEM STATEMENT**

In any IT industry, Incident (an unplanned interruption to an IT service or reduction in the quality of an IT service that affects the Users and the Business) Management plays an important role in delivering quality support to customers. The main goal of this management process is to provide a quick fix / workarounds or solutions that resolves the interruption and restores the service to its full capacity to ensure no business impact. Whenever an incident is created, it reaches the Service desk team and then it gets assigned to the respective teams to work on the incident.

The manual assignment of these incidents might have below disadvantages:

* Time consuming and requires human efforts
* Increases human errors and resource consumption, as it is carried out ineffectively because of the misaddressing.
* Increases the response and resolution times which result in user satisfaction deterioration / poor customer service

If this ticket assignment is automated, it can be more cost-effective, less resolution time and the Service Desk team can focus on other productive tasks.

**OBJECTIVE**

The goal is to build a classifier that can classify the tickets by analysing text.

**DATA DESCRIPTION**

The given dataset consists of the following four attributes:

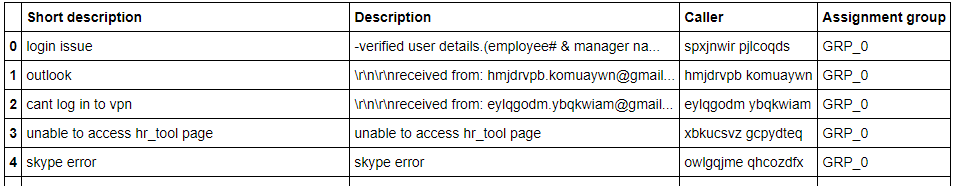
1. Short Description (a summary of the issue faced by the user)

2.Description (detailed description of the issue)

3.Caller (ID of the caller)

4.Assignment group (GRP\_0 ~ GRP\_73 i.e., total 74 classes of Assignment group) – Target class

**SAMPLE DATA**



**FINDINGS**

Followings are the general observation from the given dataset:

* Caller ID are present in a random manner (may not be useful for training data)
* Languages other than English for example- German, etc. are also present in the dataset
* Non-English languages are also found in the data
* Email/chat format with symbols in description
* Hyperlinks and URLS are found in the description
* Blank records are present in either short description or description
* Few descriptions are replica of the short description
* Few words were combined together
* Spelling mistakes and typo errors are found

**CURRENT PROCESS**

Currently the incidents are created by various stakeholders (Business Users, IT Users and Monitoring Tools) within IT Service Management Tool and are assigned to Service Desk teams (L1 / L2 teams). This team will review the incidents for right ticket categorization, priorities and then carry out initial diagnosis to see if they can resolve. Around ~54% of the incidents are resolved by L1 / L2 teams. Incase L1 / L2 is unable to resolve, they will then escalate / assign the tickets to Functional teams from Applications and Infrastructure (L3 teams). Some portions of incidents are directly assigned to L3 teams by either Monitoring tools or Callers / Requestors. L3 teams will carry out detailed diagnosis and resolve the incidents. Around ~56% of incidents are resolved by Functional / L3 teams. Incase if vendor support is needed, they will reach out for their support towards incident closure.

L1 / L2 needs to spend time reviewing Standard Operating Procedures (SOPs) before assigning to Functional teams (Minimum ~25-30% of incidents needs to be reviewed for SOPs before ticket assignment). 15 min is being spent for SOP review for each incident. Minimum of ~1 FTE effort needed only for incident assignment to L3 teams.

During the process of incident assignments by L1 / L2 teams to functional groups, there were multiple instances of incidents getting assigned to wrong functional groups. Around ~25% of Incidents are wrongly assigned to functional teams. Additional effort needed for Functional teams to re-assign to right functional groups. During this process, some of the incidents are in queue and not addressed timely resulting in poor customer service.

**GOAL**

The goal here is to create NLP based classifier that could automatically classify any ticket raised by analysing ticket description to the suitable Assignment group, this could be integrated with any ticket management service like Service Now

Based on the ticket description our model would assign a probability of it to being assigned to one of the 74 Groups.

This project intends to reduce the manual effort of IT support teams by automating the process of ticket assignment.

Summary of Data

**Data Source**

* Details about the data and dataset files are given in below link,
  + <https://drive.google.com/open?id=1OZNJm81JXucV3HmZroMq6qCT2m7ez7IJ>
* The data set contains 4 string columns

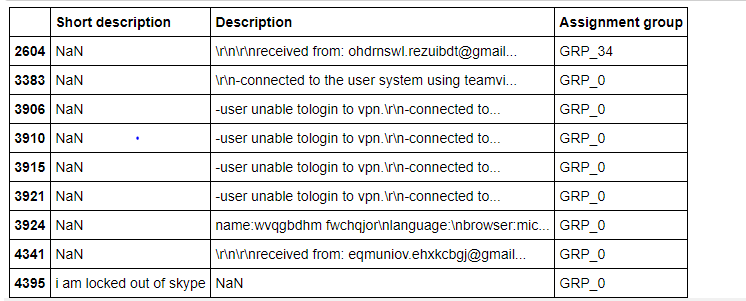
|  |  |  |
| --- | --- | --- |
| **Column** | **Description** | **Data type** |
| Short description | Short description for problem for which the incident is being raised | 8492 non-null object |
| Description | Detailed description of the problem for which the incident is being raised | 8499 non-null object |
| Caller | Masked user name | 8500 non-null object |
| Assignment Group | IT Support Group to whichthe incident has to be assigned | 8500 non-null object |

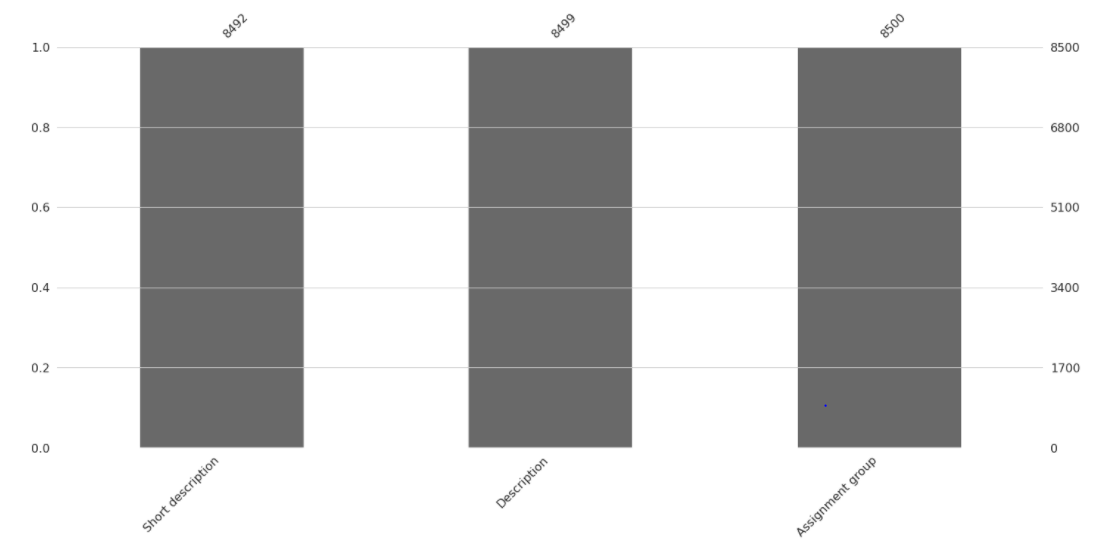
* The dataset is divided into two parts, features and the target classes.
* For given dataset, features are Short description, Description and Caller .
* For given dataset, the class variable name is Assignment group.
* There are totally 8500 rows
* There seems to be missing values in Short description and Description columns, which needs to be looked into and handled.
  + There are 8 null/missing values present in the Short description and 1 null/missing values present in the description column
* Caller columns mainly contain the details of the user who raised the incident and is of not much use in our analysis and can be dropped.
* "Short Description" and "Description" could be concatenated into a single column, so that we don’t miss information about the tickets.

Summary of Approach to EDA and Pre-processing

**Step-by-step walk through the solution with the observations**

* Loaded the input csv file into pandas’ data frame.
* EDA has been performed on the dataset and following are the observations from that:
* All columns are of type object containing textual information.
* Assignment group is our predictor / target column with multiple classes. So, this is a Multiclass Classification problem.
* There are **8 null/missing values** present in the Short description and **1 null/missing values** present in the description column.



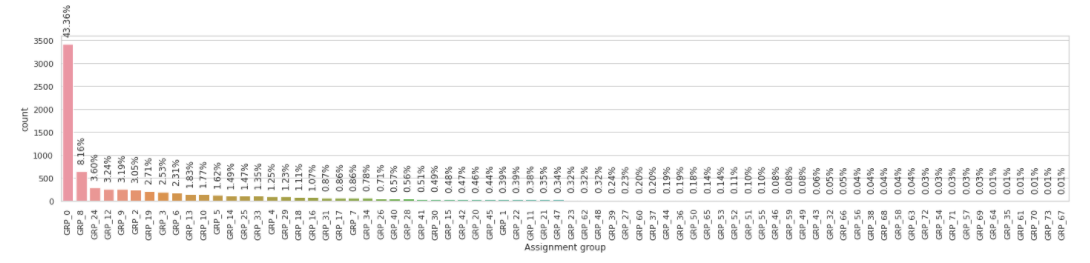
Visually the number of missing values are as shown below:

* Dropped the duplicates entries of the incidents. Thus, the size of the dataset gets reduced to (7909,4)
* Dropped the caller attribute as the data was not found to be useful for analysis
* Replaced Null values in Short description & description with space.
* Merged Short Description & Description fields for analysis

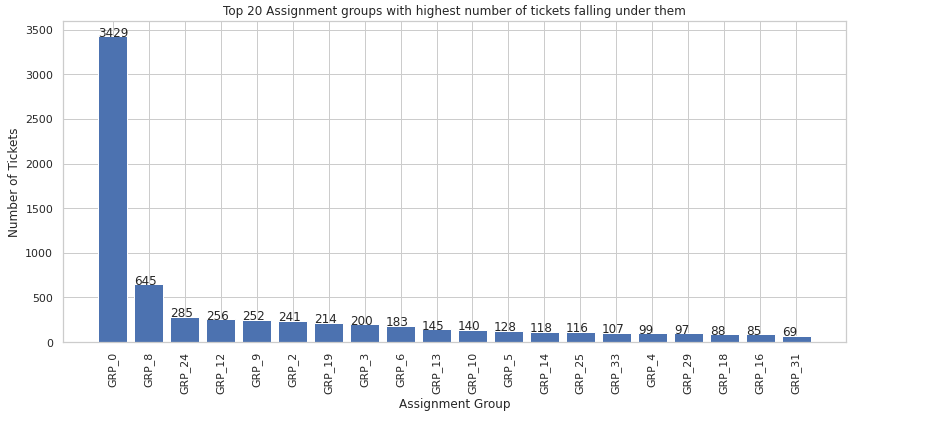
**OBSERVATIONS REGARDING TARGET CLASS**

* A large number of entries belonged to GRP\_0 (mounting to 3429 which account for ~ more than 40% of the data )

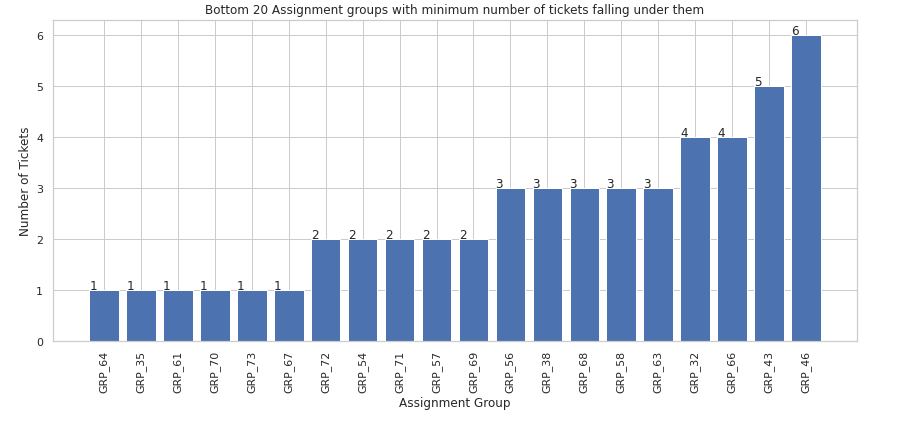
**Distribution of Target class for all groups**



* The Target class distribution is extremely skewed
* The data is too much biased towards a single group and seems to be highly imbalanced, with majority of incidents are from Group 0 followed by Group 8 , 24 , 12 , 9 , 2 and so on
* There are few classes which just have less than 10 incidents per class and even classes with just 1 or 2 incidents (samples), need to see if we can drop those rows due to the lack of samples representing those classes. They might not be of much help as a predictor
* Top 20 Assignment groups having the highest number of tickets for training the data.



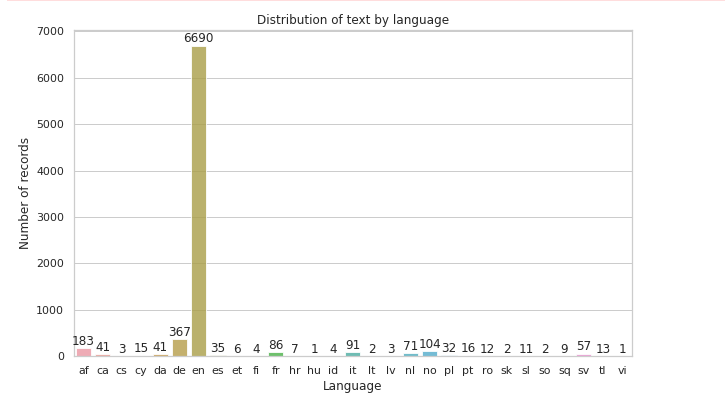
* Following are the Tickets with less number of tickets per Assignment groups.



**DATA PRE-PROCESSING**

Below steps have been performed for initial pre-processing and clean-up of data:

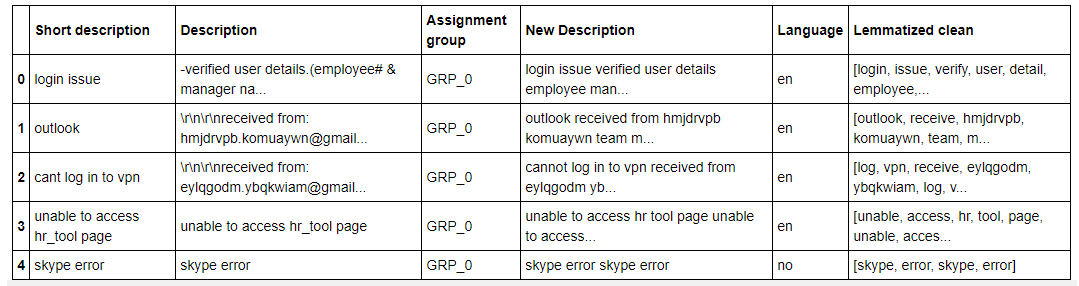
* Replaced the gibberish text using **FTFY**
* Contraction words found in the merged Description are removed for ease of word modelling
* Changed the case sensitivity of words to the common one
* Removed Hashtags and kept the words, Hyperlinks, URLs, HTML tags & non-ASCII symbols from merged fields.
* There were quite few entries with languages different from English.



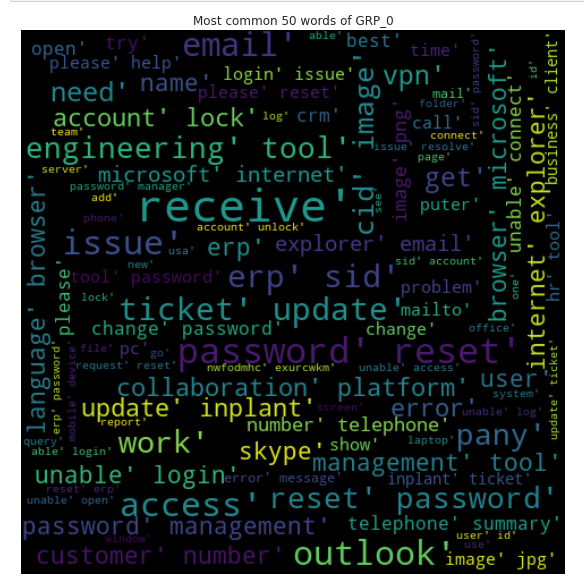
We can see that most of the tickets are in English, followed by tickets in German language.

* **Tokenization** of merged data
* **Stop words** have been removed using nltk corpus modules.
* **Lemmatization** is the process of grouping together the different inflected forms of a word so they can be analysed as a single item. Lemmatization is similar to Stemming but it brings context to the words. So, it links words with similar meanings to one word.

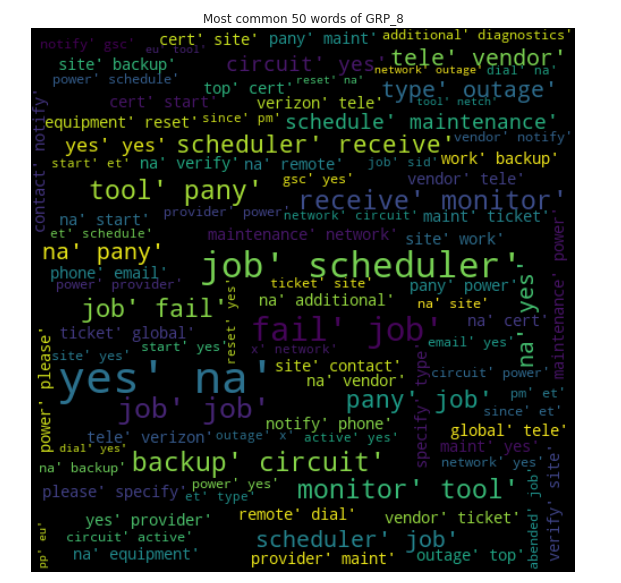
Here we have preferred Lemmatization over Stemming because lemmatization does morphological analysis of the words.



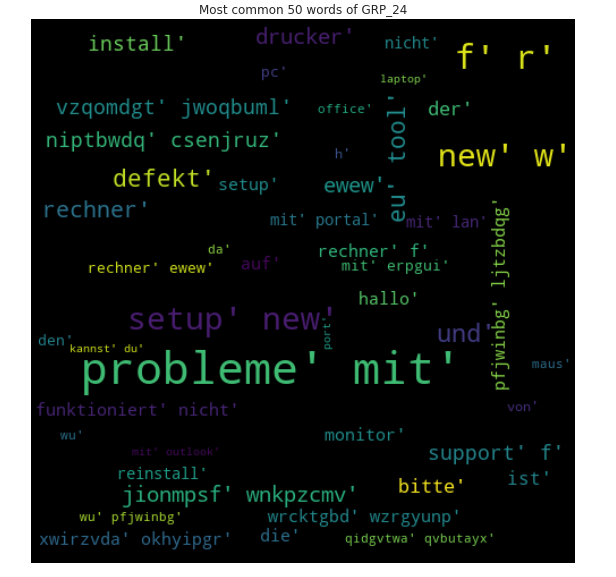
* **WordCloud** created for all available 50 groups to have more information specific to Assignment groups



**Analysis on GRP\_0 which is the most frequent group to assign a ticket to reveals that this group deals with mostly the maintenance problems such as *password, receive, reset* ,outlook, *account lock* , *login issue* , *ticket update* etc.**



**GRP\_8 seems to have tickets related *to scheduler, job failures, monitoring tool* etc.**



**GRP\_24 - Tickets are mainly in german.**

Model Selection

## **Defining independent and dependent features**

We are concatenating the Short description and Description as “New\_Description”.

“New\_Description” is considered as ***Independent attribute*** and Target – “Assignment group” is ***Dependent attribute***.

## **Splitting Datasets**

We have used the train\_test\_split function for splitting a single dataset into training and testing in 70:30 ratios.

The testing subset is for building the model. The testing subset is for using the model on unknown data to evaluate the performance of the model.

## **Machine Learning Models**

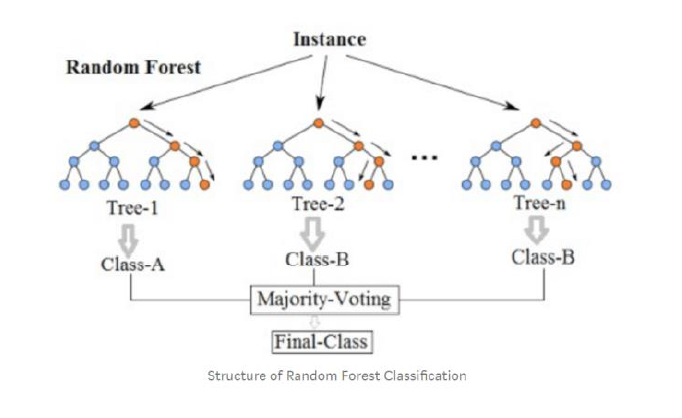
*We will be using classification algorithms, to start with we have used below basic*

*Machine Learning algorithm:*

**RANDOM FOREST CLASSIFIER**

Random forest, like its name implies, consists of a large number of individual decision trees that operate as an [ensemble](https://en.wikipedia.org/wiki/Ensemble_learning). Each individual tree in the random forest spits out a class prediction and the class with the most votes becomes our model’s prediction. The fundamental concept behind random forest is a simple but powerful one — the wisdom of crowds. In data science speak, the reason that the random forest model works so well is:

***A large number of relatively uncorrelated models (trees) operating as a committee will outperform any of the individual constituent models.***



**Below steps have been performed with the initial model:**

* Split the data into training and test set.
* Feed the data to the Random Classifier Model.
* Find the accuracy.



***As here accuracy is low, we would be trying some more models like using RNNs, LSTM, and would try to overcome imbalances in data set by doing up sampling or by assigning higher weights to class where we have less sample data.***

**Word Embedding**

As all our Machine Learning and Deep learning algorithms are incapable of processing strings or plain text in their raw form, word embeddings are used to convert the texts into numbers. There may be different numerical representations of the same text. It tries to map a word using a dictionary to a vector.

We have experimented below 2 types of embedding in our models with the dimension as 100.

**1. Word2Vector Embedding:**

Word2Vec models are shallow, two-layer neural networks that are trained to reconstruct linguistic contexts of words. Word2vec takes as its input a large corpus of text and produces a vector space, typically of several hundred dimensions, with each unique word in the corpus being assigned a corresponding vector in the space.

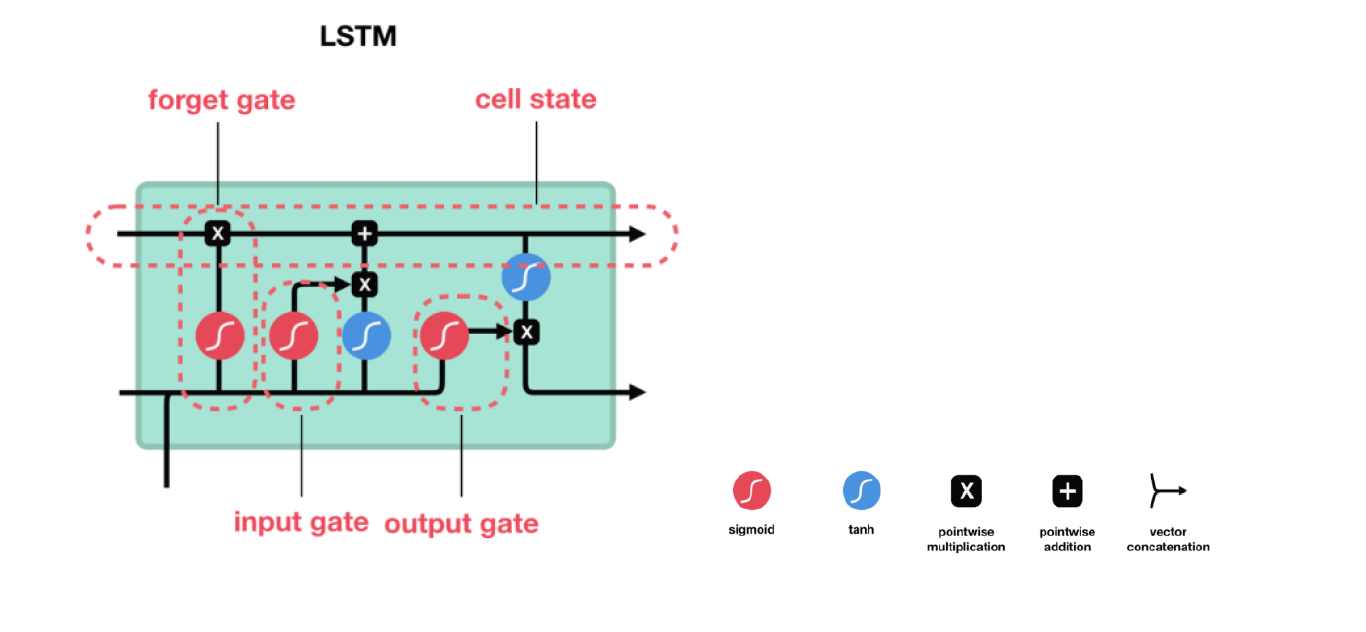
**2. GloVe (Global Vectors) Embedding:**

GloVe 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.

**Bi-directional LSTM Model**

Bidirectional LSTMs are an extension of traditional LSTMs that can improve model performance on classification problems.

In problems where all timesteps of the input sequence are available, Bidirectional LSTMs train two instead of one LSTMs on the input sequence. The first on the input sequence as-is and the second on a reversed copy of the input sequence. This can provide additional context to the network and result in faster and even fuller learning on the problem.



**CEC:** With the forget gate, influence of the state forward can be modulated such that it can be remembered for a long time, until the state or the input changes to make LSTM forget it. This ability or the path to pass the past-state unaltered to the future-state (and the gradient backward) is called constant error carrousel (CEC). It gives LSTM the ability to remember long term (hence, long short term memory)

**Blocks:** Since there are just too many weights to be learnt for a single state bit, several state bits can be combined into a single block such that the state bits in a block share gates

**Peepholes:** The state itself can be an input for the gate using peephole connections

**GRU:** In a variant of LSTM called gated recurrent unit (GRU), input gate can simply be one-minus-forget-gate. That is, if the state is being forgotten, then replace it by input, and if it is being remembered, then block the input

The flow of our Bi-directional LSTM model is as shown.

