**University of Southern Denmark**

# *Innovation Tendency Identifier*

**Sangramsing Nathusing Kayte**

*Supervisor:* **Peter Schneider-Kamp**

**Department of Mathematics and Computer Science (IMADA)**
**University of Southern Denmark (SDU),**
**Odense M, Denmark 5230**

**September 24, 2020**

## Abstract

This research project proposes a new method of automatized text generation and subsequent classification of the European Union (EU) Tender Electronic Daily (TED) text documents into predefined technological categories of the dataset. The TED dataset provides information about the respective tenders includes features like name of project, Title, Description, Types of contract, Common procurement vocabulary (CPV) code, and Additional CPV codes. The dataset is obtained from the SIMAP-Information system for the European public procurement website, which is comprised of tenders described in XML files. The dataset was pre-processed using tokenization, removal of stop words, removal of punctuation marks etc. We implemented a neural machine learning model based on Long Short-Term Memory (LSTM) nodes for text generation and subsequent code classification. Text generation means that given a single line or just two or three words of the title, the model generates the sequence of a whole sentence. After generating the title, the model predicts the main applicable CPV code for that title. The LSTM model reaches an accuracy of 97% for the text generation and 95% for code classification. This experiment is a first step towards developing a system that based on TED data is able to auto-generate and code classify tender documents, easing the process of creating and disseminating tender information to TED and ultimately relevant vendors. The development and automation of this system will future vision and understand current undergoing projects and the deliveries by a SIMAP-Information system for European public procurement tenders organisation based on the tenders published by it.

## 1 Introduction

Natural language processing (NLP) is a wide area concerned with the application of computers to analyze, understand, and derive meaning from human language in a smart and useful way. NLP can organize and structure knowledge to perform tasks such as automatic summarization, translation, named entity recognition, relationship extraction, sentiment analysis, speech recognition, and topic segmentation. The unswerving increase of categories for tender documents has resulted in the need for methods that can automatically classify new documents within sub categories of the text. In this research project we attempt to make the categorization process automated using a machine learning approach, i.e., to learn from the European public procurement tender data in Tender Electronic Daily (TED) database. Given text describing the tender, the model predicts codes using the common procurement vocabulary (CPV) that are relevant for this text. We commence this research project with an overview of European Union (EU) TED and then move to the introduction of a few terminologies which are relevant to this research project. TED works under the Policies of European Public procurement as drawn by the Directorate General for Internal Market, Industry, Entrepreneurship, etc. A tender is a document that a purchasing agent publishes to announce its request for certain goods or services. The whole electronic tendering system is similar to that of a traditional one, i.e. a selling agent submits a bid to the purchasing organisation against the tender enquiry, which evaluates the quoting vendors based on the technical specifications as well as financial terms and conditions.

Then the purchasing agent announces the successful bidder. We carried out the classification on the SIMAP-Information system for European public procurement tenders available on its website ($https://ted.europa.eu/TED/main/HomePage.do$). eNotices is the online tool for preparing public procurement notices and publishing them in the Supplement to the Official Journal of the EU.

There is a standard format for the calling of tenders in public procurement in European nations, which are referred online with the process, the Tender European Daily (TED) database. In this, there are different sections allotted to various categories like construction, financing, operation, and distribution of water bottling facilities, etc. The syntactically-parsed information is maintained in these TED tenders

**Table 1:** *Project features of data*

| Features | Data type | Values |
|---|---|---|
| Name | Text | singe line free text |
| Description | Text | multiple line free text |
| Types of contract | Text | one of WORKS, SUPPLIES, or SERVICES |
| CPV code | Number | from predefined hierarchical catalog of codes |
| additional CPV codes | List of numbers | from predefined hierarchical catalog of codes |

The primary goal of this research is to classify tenders according to (parts of) the title of the project. Titles are generated from the user-given text. The implemented system achieves an accuracy of 97% when generating titles from user-given text and an accuracy of 95% when predicting CPV codes from the generated titles. After a deep inspection of the dataset, it becomes apparent that the CPV code describes the uniqueness of each tender. Here, we tried to extract information about certain tenders in TED based on year-wise allotment along with their CPV codes. So when the CPV code is given it retrieves the information about the project.

## 2 Aims and Objectives

**Aims:**

The primary aim of this project involves the collection of Tender based database. After a deep study of this database, we found 6 attributes of the tender as follows: Title, Description, services, CPV Main code, and additional CPV. We have applied the NLP techniques like Extracting entities, tokenization, Encoder-Decoder recurrent neural network architecture, and LSTM-Sequence model. We are implementing Deep learning techniques like Recurrent neural networks (RNNs) and Long short-term memory (LSTM) for text prediction also we use for text (Cpv main) category classification using the Linear Support vector machine (SVM).

**Objectives:**

The objective is to accomplish the task of predicting the text sequence and its extraction Cpv main related code information from the tender database. The objectives for this project are defined as follows:

1. Application of neural network model approaches to overcome the disadvantages of traditional machine learning approaches like SVM or Logistic regression.

2. Keras LSTM model to make predictions is to first start off with a seed sequence as input, generate the next character then update the seed sequence to add the generated character on the end and trim off the first character. This process is repeated for as long as we want to predict new characters for text prediction.

3. The objective is to predict a word based on its context, typically using a shallow neural network.

4. Create a language model for generating natural language text by implement and training state-of-the-art Recurrent Neural Network.

5. The objective of this model is to generate new text, given that some input text is present.

6. Word2Vec is embedded for syntactical and semantic information from the given text database.

7. The Extraction information form tender database and extract information with using as per tender Cpv main code.

8. Generate text sequence given short sentence

## 3 Literature Review

The neurons interconnected with each other simulate the network structure of neurons in the human brain

This allows them to deal with a time sequence with temporal relations such as speech recognition

For NLP, it is useful to analyze the distributional relations of word occurrences in documents

## 4 Natural language processing

Natural language processing (NLP) is the interpretation and use of any human language with the help of computer. There is scope for different programs that are designed to read and emit specialized languages to handle efficient and unambiguous parsing by simple programs. Generally, used languages are often ambiguous and defy formal description. NLP includes applications such as machine translation, text predicting, text segmentation etc. NLP applications are based on language models that define a probability distribution over sequences of words, characters or bytes in a natural language.

As with the other applications discussed in this project report, very generic neural network techniques can be successfully applied to NLP. However, to achieve excellent performance and to scale well to large applications, some domain-specific strategies become important. To build an efficient model of natural language, we need to implement techniques that are specialized for processing sequential data. Building of natural language model as a sequence of words is more preferred rather than a sequence of individual characters or bytes.Several strategies have been developed to make models of such a space efficient, both in a computational and in a statistical field.

1. **n-grams:-**

   A language model defines a probability distribution as per the sequences of tokens in a natural language. Depending on how the model is designed, a token is defined as a word, a character, or even a byte with discrete entities. The earliest successful language models were based on models of fixed-length sequences of tokens called $n - grams$. An $n - gram$ is a sequence of n tokens. Models based on $n - grams$ define the conditional probability of the $n - th$ token given the preceding $n1$ tokens. The model uses products of these conditional distributions to define the probability distribution over longer sequences:

   $$P(x_1, \ldots, x_\tau) = P(x_1, \ldots, x_{n-1}) \prod_{t=n}^{\tau} P(x_t | x_{t-n+1}, \ldots, x_{t-1}) \tag{1}$$

   This decomposition is justified by the chain rule of probability. The probability distribution over the initial sequence $P(x_1, \ldots, x_{n-1})$ may be modeled by a different model with $a$ smaller value of $n$.

   Training $n{-}gram$ models is straightforward because the maximum likelihood estimate can be computed simply by counting how many times each possible n gram occurs in the training set. Models based on n-grams have been the core building block of statistical language modeling for many decades

   The statistical accuracy of $n - gram$ models, class-based language models introduce the notion of word categories and then share statistical strength between words that are in the same category. The clustering algorithm is applied to partition the set of words into clusters or classes, based on their co-occurrence frequencies with other words. The model can then use word class IDs rather than individual word IDs to represent the context on the right side of the conditioning bar. Composite models combining word-based and class-based models via mixing or back-off are also possible. Although word classes provide a way to generalize between sequences in which some word is replaced by another of the same class, much information is lost in this representation.

2. **Neural Language Models**:-

   $Neural\ language\ models$ or NLMs are a class of language model that uses a distributed representation of words which loopholes the curse of dimensional problem for modeling natural language sequences.

   We sometimes call these word representations word embeddings. In this interpretation, we view the raw symbols as points in a space of dimension equal to the vocabulary size. The word representations embed those points in a feature space of lower dimension. In the original space, every word is represented by a one-hot vector, so every pair of words is at Euclidean distance 2 from each other. In the embedding space, words that frequently appear in similar contexts (or any pair of words sharing some features

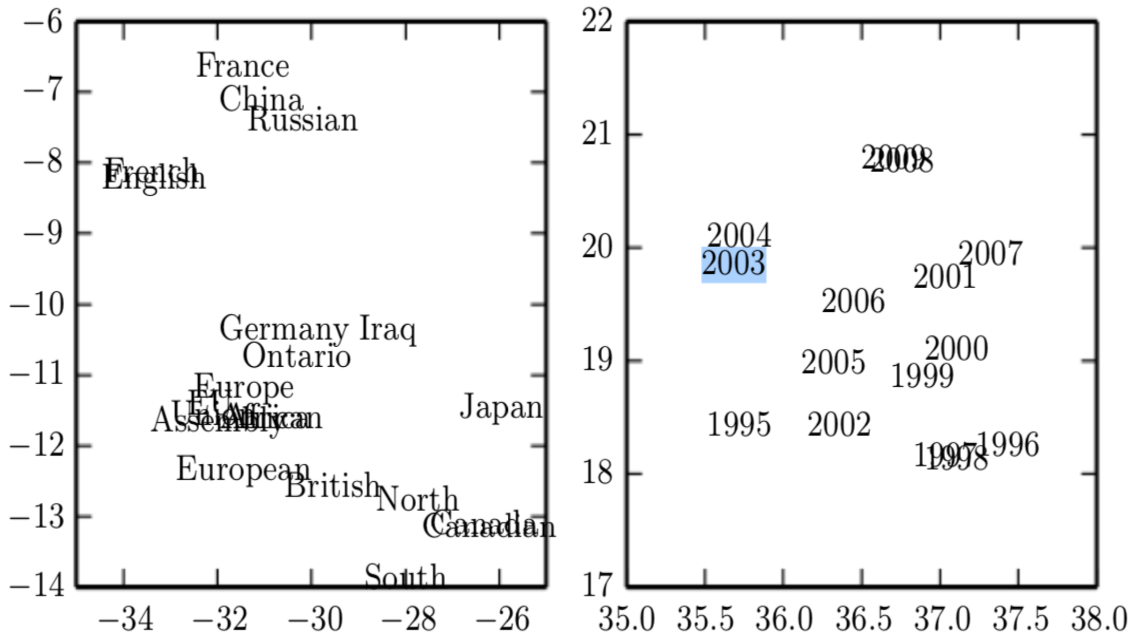


**Figure 1:** *Two-dimensional visualizations of word embeddings obtained from a neural machine translation model, zooming in on specific areas where semantically related words have embedding vectors that are close to each other*

learned by the model) are close to each other. This often results in words with similar meanings being neighbors. Figure 1 zooms in on specific areas of a learned word embedding space to show how semantically similar words map to representations that are close to each other.

Neural networks in other domains also define embeddings. For example, a hidden layer of a convolutional network provides an image embedding. Usually NLP practitioners are much more interested in this idea of embeddings because natural language does not originally lie in a real-valued vector space. The hidden layer has provided a more qualitatively dramatic change in the way the data is represented. The basic idea of using distributed representations to improve models for NLP is not restricted to neural networks. It may also be used with graphical models that have distributed representations in the form of multiple latent variables

3. **High-Dimensional Outputs :-**

   In many natural language applications, we often want our models to produce words (rather than characters) as the fundamental unit of the output. For large vocabularies, it can be very computationally expensive to represent an output distribution over the choice of a word, because the vocabulary size is large. In many applications, $V$ contains hundreds of thousands of words. The naive approach to representing such a distribution is to apply an affine transformation from a hidden representation to the output space, then apply the SoftMax function. Suppose we have a vocabulary $V$ with size $|V|$. The weight matrix describing the linear component of this affine transformation is very large, because its output dimension is $|V|$.. This imposes a high memory cost to represent the matrix, and a high computational cost to multiply by it. Because the SoftMax is normalized across all $|V|$ outputs, it is necessary to perform the full matrix multiplication at training time as well as test timewe cannot calculate only the dot product with the weight vector for the correct output. The high computational costs of the output layer thus arise both at training time (to compute the likelihood and its gradient) and at test time (to compute probabilities for all or selected words). For specialized loss functions, the gradient can be computed efficiently, but the standard cross-entropy loss applied to a traditional softmax output layer poses many difficulties

Suppose that $h$ is the top hidden layer used to predict the output probabilities $y$. If we parametrize the transformation from $h$ to $y$ with learned weights $W$ and learned biases $b$, then the affine-SoftMax output layer performs the following computations:

$$a_i = b_i + \sum_j W_{ij} h_j \quad \forall i \in \{1, \ldots, |\mathbb{V}|\} \tag{2}$$

$$\hat{y}_i = \frac{e^{a_i}}{\sum_{i'=1}^{|\mathbb{V}|} e^{a_{i'}}} \tag{3}$$

If $h$ contains $n_h$ elements then the above operation is $O(|V|\mathrm{n}_h). With \mathrm{n}_h$ in the thousands and $|V|$ in the hundreds of thousands, this operation dominates the computation of most neural language models.

4. **Combining Neural Language Models with n-grams:-**

A major advantage of $n-gram$ models over neural networks is that $n-gram$ models achieve high model capacity (by storing the frequencies of very many tuples) while requiring very little computation to process an example (by looking up only a few tuples that match the current context). If we use hash tables or trees to access the counts, the computation used for $n-gram$ is almost independent of capacity. In comparison, doubling a neural networks number of parameters typically also roughly doubles its computation time. Exceptions include models that avoid using all parameters on each pass. Embedding layers index only a single embedding in each pass, so we can increase the vocabulary size without increasing the computation time per example. Some other models, such as tiled convolutional networks, can add parameters while reducing the degree of parameter sharing in order to maintain the same amount of computation. However, typical neural network layers based on matrix multiplication use an amount of computation proportional to the number of parameters.

One easy way to add capacity is thus to combine both approaches in an ensemble consisting of a neural language model and an $n - gram$ language model

5. **Neural Machine Translation :-**

   Machine translation is the task of reading a sentence in one natural language and emitting a sentence with the equivalent meaning in another language. Machine translation systems often involve many components. At a high level, there is often one component that proposes many candidate translations. Many of these translations will not be grammatical due to differences between the languages. For example, many languages put adjectives after nouns, so when translated to English directly they yield phrases such as apple red. The proposal mechanism suggests many variants of the suggested translation, ideally including red apple. A second component of the translation system, a language model, evaluates the proposed translations, and can score red apple as better than apple red. The earliest use of neural networks for machine translation was to upgrade the language model of a translation system by using a neural language model

   Traditional language models simply report the probability of a natural language sentence. Because machine translation involves producing an output sentence given an input sentence, it makes sense to extend the natural language model to be conditional.

6. **Historical Perspective :-**

   The idea of distributed representations for symbols was introduced in one of the first explorations of back-propagation, with symbols corresponding to the identity of family members and the neural network capturing the relationships between family members, with training examples forming triplets such as (Colin, Mother, Victoria)

   The idea of forming an embedding for a symbol was extended to the idea of an embedding

   The history of NLP is marked by transitions in the popularity of different ways of representing the input to the model. Following this early work on symbols or words, some of the earliest applications of neural networks to NLP represented the input as a sequence of characters

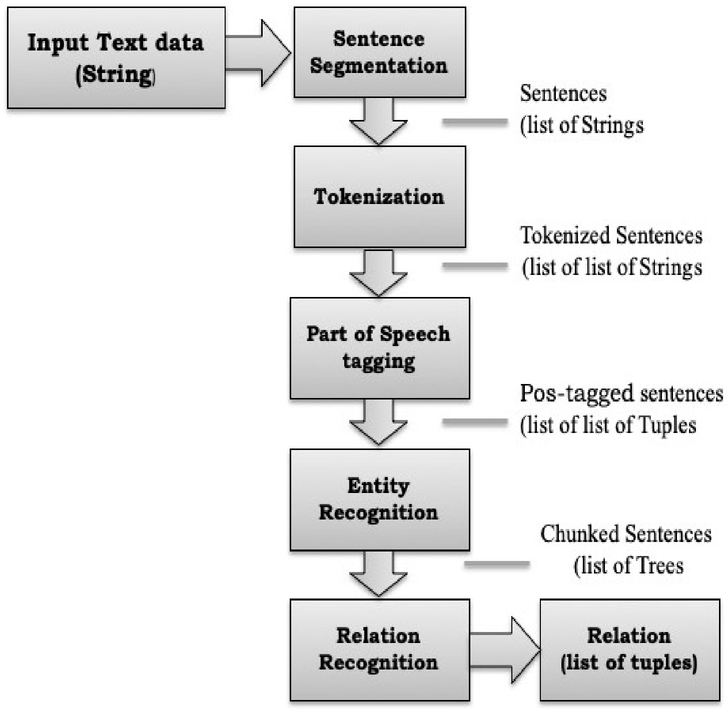

Input Text data (String)
Sentence Segmentation
Sentences (list of Strings
Tokenization
Tokenized Sentences (list of list of Strings
Part of Speech tagging
Pos-tagged sentences (list of list of Tuples
Entity Recognition
Chunked Sentences (list of Trees
Relation Recognition
Relation (list of tuples)

**Figure 2:** *Pipeline For Natural Language Processing*

## 4.1 Pipeline of Natural Language Processing

NLP facilitates the use of computers to process or to understand human languages for applications like information filtering, language identification, readability assessment, and sentiment analysis, etc. NLP is an interdisciplinary field that is a combination of the subfield of computer science, information engineering, and artificial intelligence. From a scientific perspective, NLP aims to model the cognitive mechanisms underlying the understanding and production of human languages

(a) **Text Segmentation**:-

This method involves the sentence is broken into a sequence of contiguous parts called segments. There exist another sequence segmentation like sentence segmentation breaking a piece of string into sentences which is an important post-processing stage for speech transcription and chunking also known as shallow parsing to find important phrases from sentences, such as noun phrases

(b) **Named Entity Recognition (NER)**

NER is the process of finding and tagging named entities existing in the given text into pre-defined categories. The NER task is hugely dependent on the knowledge base used to train the

NER extraction algorithm, so the results vary on the provided dataset it was trained on.

(c) **Word Embedding**

A word embedding learning method is used to calculate the embeddings using distributional semantics. The way to evaluate the approximate meaning of a word by considering how often it co-occurs with other words in the corpus. These embedding-based metrics usually approximate sentence-level embeddings using some heuristic to combine the vectors of the individual words in the sentence. The sentence-level embedding between the generated and reference response are compared using a measure such as a cosine distance

(d) **Word2vec**

Word2vec is a two-layer neural net used in the recurrent neural network that processes text. Its input is a text corpus and its output is a set of vectors. It gives the best results for any kind of sequence neural network models to learn node and sequence representations based on node sequences

(e) **Bag-of-Words (BOW)**

The concept of BOW is to predict the target word with surrounding context words. For convenient, the surrounding words are symmetric (so as in skip-gram), i.e., a window with size m is predefined and the task is to predict the target word $w_c$ with a sequence of words $(wcm, ..., wc1, wc{+}1, ..., wc{+}m)$, where $w_i$ denotes the word at position $c$

(f) **Text Cleaning**

The text cleaning is step for converting the text into a particular input format. This helps to remove the tagged data, punctuation, stop words, abbreviations from the given input text

i. Removing the HTML tags

ii. Removing Punctuation and Stopwords

iii. Stemming

iv. Conversion of data text to lower case

## 5 Corpus Description

The corpus consists of the information about the announcements and calls related to tenders in TED. The data is stored in an open XML format. We extracted it to a Comma-Separated Values (CSV) format, containing the information in form of text and numbers as indicated in Table 1. The CSV file contains information like the title of the project and CPV code as an important entity along with other sub-entities like description, reference of the project. The other entities like types of contract and a detailed description of a particular project are also part of the dataset. In this way, we are having a dataset of approx. 40,000 texts for each year from 2015 to 2019. This data will be helpful for training and implementing the machine learning model for the required system.

## 6 Proposed Framework

The proposed framework consists of NLP library for pre-processing such using Natural Language Toolkit (NLTK) library and unidirectional LSTM architecture for sequence prediction and classification. We have also applied linear support vector machine (SVM) classifier to classify CPV main code category from other relevant fields, which is shown in the architecture in Figure 3.

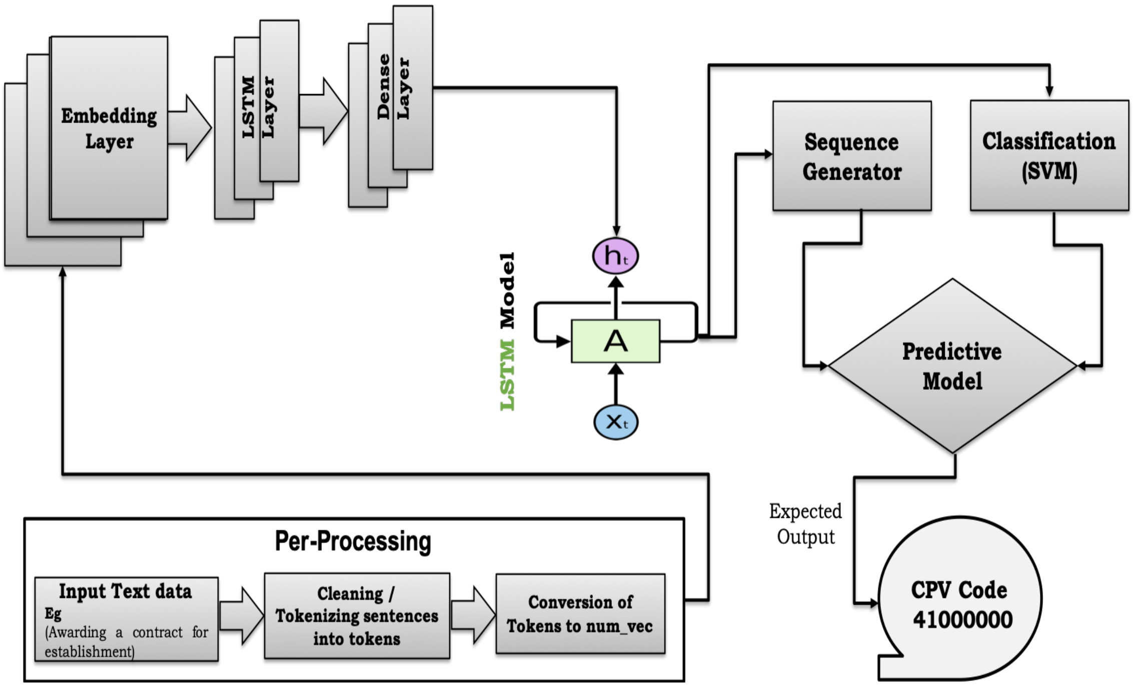


**Figure 3:** *The system architecture of the proposed approach*

The tender category of the title describe overall information about the particular product. Initially, first 4-6 words from the title are given as input for the system, the LSTM model generates the overall text information. At the first stage, pre-processing and word features are extracted. Secondly, the

Word2Vec word embedding model is used to learn word representations as vectors.

The external input gate unit $g_i^{(t)}$ is computed similarly to the forget gate (with a sigmoid unit to obtain a getting value between 0 and 1), but with its own parameters of weights and model:

$$g_i(t) = \sigma\left(b_i^g + \sum_j U_{i,j}^g x_j^{(t)} + \sum_j W_{i,j}^g h_j^{(t-1)}\right) \tag{4}$$

The output of the LSTM cell can also be shut off, via the output gate $q_i^{(t)}$, which also uses a sigmoid unit for gating:

$$q_i(t) = \sigma\left(b_i^o + \sum_j U_{i,j}^o x_j^{(t)} + \sum_j W_{i,j}^o h_j^{(t-1)}\right) \tag{5}$$

which has parameters $b^o, U^o, W^o$ for its biases, input weights and recurrent weights, respectively. Among the variants, one can choose to use the cell state $q_i^{(t)}$ as an extra input (with its weight) into the three gates of the $i - th$ unit, as shown in the above equation. Thus the LSTM model takes the recurrent input of weights and concurrently gives the output to the cell state

(a) **Preprocessing**

In this stage, we filtered the text data into required format needed for the training of the system. The pre-processing is divided into four different steps:

i. **Cleaning** consists of getting rid of the less useful parts of a text through stopword removal, dealing with capitalization and characters, and other minor details.

ii. **Annotation** This process involves the application of a scheme to texts which include structural markup and part-of-speech tagging.

iii. **Normalization** consists of the translation (mapping) of terms in the scheme or linguistic reductions through Stemming, Lemmatization, and other forms of standardization.

iv. **Analysis** consists of statistically probing, manipulating, and generalizing from the dataset for feature analysis.

(b) **Tokenization** Tokenization is the process of splitting text fields into meaningful segments by locating the word boundaries like CPV code, title from the given text database. The points where one word ends and another begins. For computational linguistic purposes, the words thus identified are frequently referred to as tokens. In written languages where no word boundaries are explicitly marked in the writing system, tokenization is also known as word segmentation, and this term is frequently used synonymously with tokenization.

(c) **Word Embedding** Word embeddings is the representation of title, description of words in the form of vectors. The aim is to represent words via vectors such that similar text words or words used in a similar context are close to each other while antonyms end up far apart in the vector space.

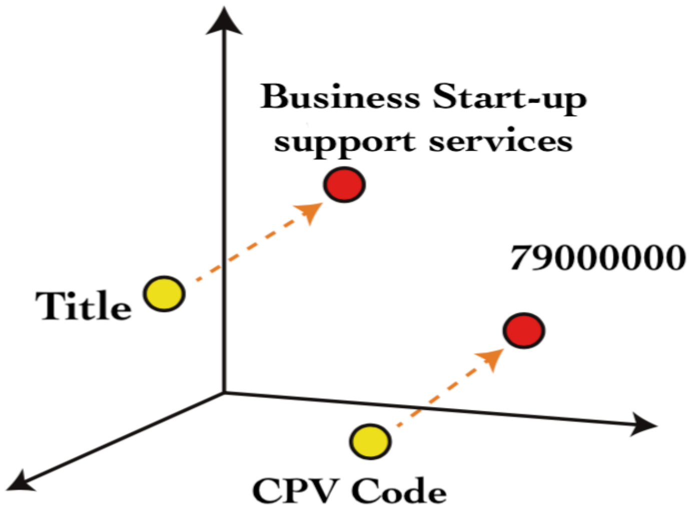


**Figure 4:** *Visualization of Word Embeddings*

The conversion of word vectors into a high-dimensional space can be envisioned as shown in Figure 4

(d) **Dense Layer** The dense layer contains a linear operation in which every input is connected to every output by weight (so there are n_inputs * n_outputs weights - which can be a lot!). Generally, this linear operation is followed by a non-linear activation function.

(e) **Support Vector Machines**

SVM is text classification approach which can be applied in both types of classification and regression problems. The feature of SVM model find a maximum marginal hyperplane(MMH) that best divides the dataset into classes and minimization of error. The classifier identifies the unique data points using a hyperplane with the largest amount of margin using training data.SVM finds an optimal hyperplane which helps in classifying new data points and so also referred as discriminative classifier

The decision function $g$ in SVM framework is defined as:

$$g(x) = sgn(f(x))) \tag{6}$$

$$f(x) = \sum_{i=1}^{l} y_i \alpha K(x_i, x) + b \tag{7}$$

where $K$ is a kernel function and $\sigma i$ are weights

## 6.1 Long Short-Term Memory (LSTM)

RNNs have a powerful sequence study capability, which can finely describe the dependence relationship of title and CPV code data

The LSTM deals especially in handling sequences that have a large gap between relevant information for learning or predictions of new features in the training set. The attractive feature of LSTM is that it effectively deals with the long-term dependence problem in time sequences

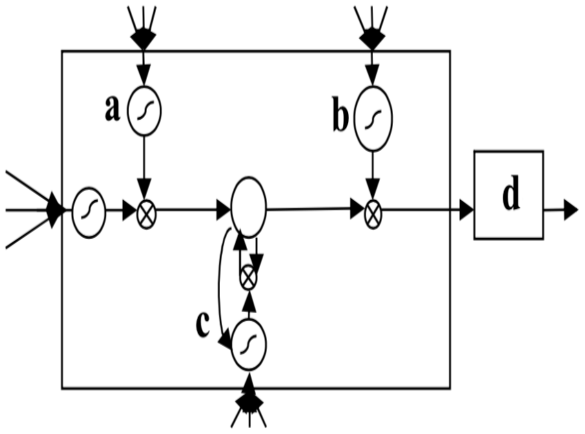


**Figure 5:** *The LSTM cell. a-Input gate, c-Forget gate, b-Output gate, d- output from the cell*

Figure 5 shows the structure of the LSTM network, where the repeated modules represent the hidden layer in each iteration. Each hidden layer contains a number of neurons. Each neuron performs a linear matrix calculation on the input vector and then outputs the corresponding results after the nonlinear action of the activation function

In each iteration, the output of the previous iteration interacts with the next word vector of the text determines the preservation or abandonment of the information and the update of the current state.

$h_t$ is the input of the hidden layer in this iteration. According to the current state information, the predicted output value of hidden layer $y$ is obtained, and the output vector $h_t$ is provided for the next hidden layer at the same time. Whenever there is a new word vector input in the network, the output of the hidden layer at the next moment is calculated in conjunction with the output of the hidden layer at the last moment. The hidden layer circulates and keeps the latest state

## 7 Experimental Evaluation

We have calculated the accuracy of our NLP model using text prediction and classification. This involves the concepts of true positives, true negatives, false positives, and false negatives are also evaluated. False positives are cases the model incorrectly labels as positive that are actually negative, or, in our example, the text which is not identified is considered a false negative. True positives are data points classified as positive by the model that actually are positive (meaning they are correct), and false negatives are data points the model identifies as negative that actually are positive (incorrect). The LSTM-based model we implemented achieves an accuracy of 97% for the text generation and 95% for the code classification.

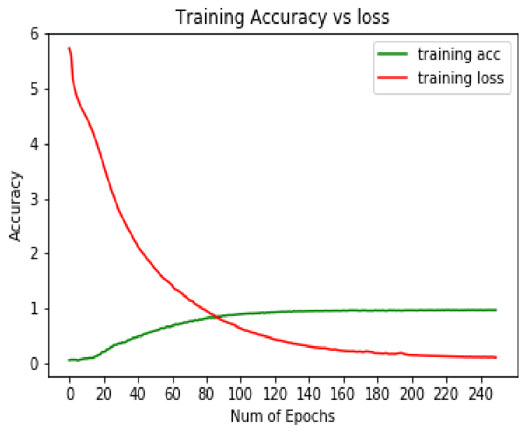


**Figure 6:** *ROC curve against the training accuracy and loss*

An Receiver operating characteristic (ROC) curve plots the accuracy on the x-axis against the number of epochs on y-axis. The true positive rate (TPR) is the recall and the false positive rate (FPR) is the probability of a false alarm. Figure 6 shows the ROC curve of training accuracy and training loss of the system against the number of epochs. We can read from observations that the training accuracy gradually increases and loss decreases as the number of epochs for training of the system progresses. This generates and classifies the data accurately from the given dataset of the model.

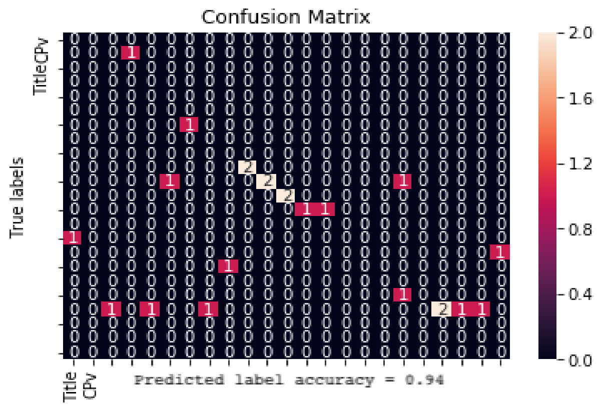


**Figure 7:** *Confusion Matrix training accuracy*

Figure 10 confusion matrix is a summary of prediction results on a classification of text data from given dataset. In this matrix, the number of correct and incorrect predictions like title of project and CPV code are summarized with count values and broken down by each class. The confusion matrix helps to identify the ways in which your classification model is confused when it makes predictions. It also clears the errors being made by your classifier but more importantly the types of errors that are being made. So in this project, the confusion matrix is applied for the classification of text and the percentage accuracy is 95% where the title and CPV code is exactly identified from the given text data.

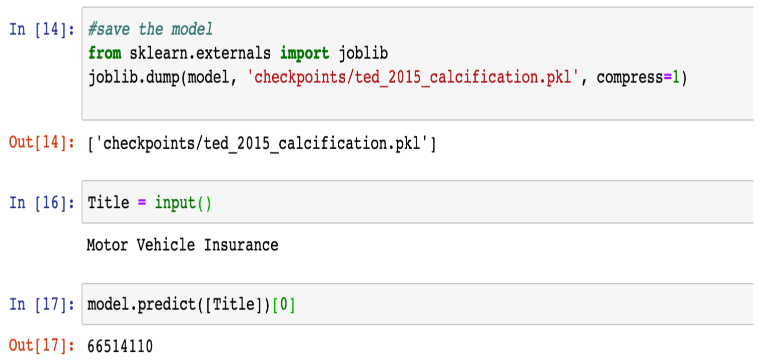
## 8 Output Result

### 8.1 Sequence Prediction

The Figure 8 shows the Sequence prediction output.

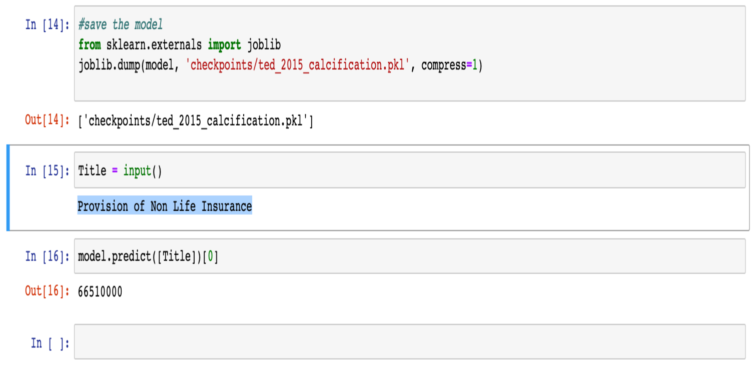
### 8.2 Calcification Model

The Figure 9 shows the Calcification Model output.



```
In [14]: #save the model
         from sklearn.externals import joblib
         joblib.dump(model, 'checkpoints/ted_2015_calcification.pkl', compress=1)

Out[14]: ['checkpoints/ted_2015_calcification.pkl']

In [16]: Title = input()

         Motor Vehicle Insurance

In [17]: model.predict([Title])[0]

Out[17]: 66514110
```

**Figure 8:** *Sequence Prediction*



```
In [14]: #save the model
         from sklearn.externals import joblib
         joblib.dump(model, 'checkpoints/ted_2015_calcification.pkl', compress=1)

Out[14]: ['checkpoints/ted_2015_calcification.pkl']

In [15]: Title = input()

         Provision of Non Life Insurance

In [16]: model.predict([Title])[0]

Out[16]: 66510000

In [ ]:
```

**Figure 9:** *calcification Model*

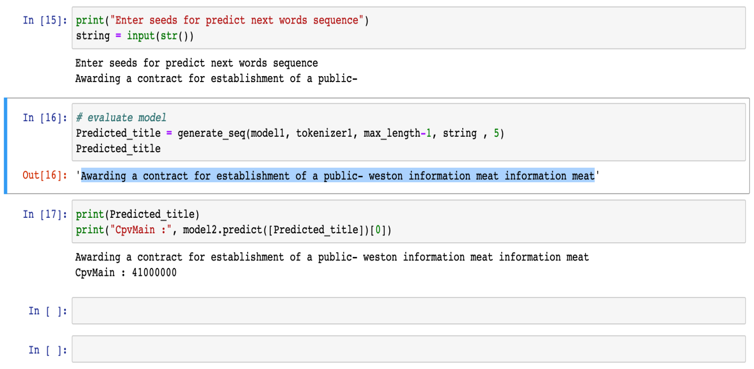
## 8.3 Prediction and Calcification

The Figure 10 shows the Prediction and Calcification output.

## 9 Conclusions

This experimental research project reported a high-accuracy approach based on an LSTM model for predicting CPV codes for TED tenders using a two step approach of generating texts and subsequently classifying the generated texts. Future work is to further increase the performance of the system as well as to automate the remaining steps of TED tender creation to the fullest extent possible.

## References



```
In [15]: print("Enter seeds for predict next words sequence")
         string = input(str())

         Enter seeds for predict next words sequence
         Awarding a contract for establishment of a public-

In [16]: # evaluate model
         Predicted_title = generate_seq(model1, tokenizer1, max_length-1, string , 5)
         Predicted_title

Out[16]: 'Awarding a contract for establishment of a public- weston information meat information meat'

In [17]: print(Predicted_title)
         print("CpvMain :", model2.predict([Predicted_title])[0])

         Awarding a contract for establishment of a public- weston information meat information meat
         CpvMain : 41000000

In [ ]:

In [ ]:
```

**Figure 10:** *Prediction and Calcification*