Focused domain contextual AI chatbot framework for resource poor languages

Anirudha Paul, Asiful Haque Latif, Foysal Amin Adnan and Rashedur M Rahman

Department of Electrical and Computer Engineering, North South University, Dhaka, Bangladesh

ABSTRACT
In today’s business world, providing reliable customer service is equally important as delivering better products for maintaining a sustainable business model. As providing customer service requires human resource and money, businesses are often shifting towards artificial intelligence system for necessary customer interaction. However, these traditional chatbot architectures depend heavily on natural language processing (NLP), it is not feasible to implement for the languages with little to no prior NLP backbone. In this work, we propose a semi-supervised artificially intelligent chatbot framework that can automate parts of primary interaction and customer service. The primary focus of this work is to build a chatbot which can generate contextualized responses in any language without depending much on rich NLP background and a vast number of a prior data set. This system is designed in such a way that with a dictionary of a language and regular customer interaction dataset, it can provide customer services for any business in any language. This architecture has been used to build a customer service bot for an electric shop, and different analysis has been done to evaluate the performance of individual components of the framework to show its competence to provide reliable response generation in comparison with other approaches.

ARTICLE HISTORY
Received 14 January 2018
Accepted 8 December 2018

KEYWORDS
Machine learning; convolutional neural network; natural language processing; chatbot; customer service; context awareness; Bangla; English

Introduction
E-Commerce is a growing sector in the world. With the growth of social media, any person can easily market and sell their products online. However, as the business grows, it is quite challenging to answer queries, take and confirm orders coming from a vast number of people. So, most of the time a business owner relies on a separate division or company to provide this kind of customer service. Even in a small country like Bangladesh; this customer service provider market was around 13 million dollars in 2013 and half of the revenue was collected from the domestic market (Hasan, 2018). These customer care services cannot always guarantee that customers do not need to wait for getting a reply. So, business owners are leaning towards AI (Artificial Intelligence) Bot that can handle natural language conversation with the user and serve him/her with the desired service.
But if a business owner wants to use existing enterprise-level solutions like – Chatfuel (2017), Flowxo (2017), Motion.ai (2017), Botsify (2017), Morph.ai (2017), ChattyPeople (2017) etc. for our country, Bangladesh, it is not possible at all. All of them rely heavily on already existing Natural Language Processing methods for English and some other limited number of resourceful languages. They cannot work on a completely resource-poor language like Bangla which is the native language in Bangladesh. Another problem attached to these platforms is that most of them use rule-based architecture to identify different kinds of request. Generating those rules is a difficult task for any business owner, and they do not always guarantee proper responses. Additionally, these chatbots force the user to take a specific path of the conversation by providing conversation button of next query rather than actual natural chat conversation. Another aspect of these chatbots is that most of them do not take advantage of already existing data owned by the business owners and they do not provide any scientific evaluation metrics to evaluate the performance of their chatbots before investing on them.

The primary motivation of our work is to avoid the incompetence of the existing chatbots and provide an easy to implement multilingual context-aware chatbot for any domain. As most of the chatbot relies heavily on NLP (thus faces the problem with language dependency), has to maintain a predefined path of conversation, cannot use existing dataset and most of them do not provide contextualization in conversation; we have addressed these limitations by designing our chatbot around a simple NLP solution rather than a complicated one like parts of speech tagging, semantic analysis to avoid language dependency. The structure of the data set for training the chatbot was well thought so that anyone can take advantage of the currently available data as well as new data generated from time to time. Ease of use is considered while designing the training data structure. To automate the need of manual rule generation for finding different patterns in a conversation; we have tested some well-established data mining and machine learning techniques such as – Centroid, K nearest neighbour, Support Vector Machine, various Neural Networks and evaluated them with renowned techniques to find out the performance of each of these methods in domain-specific scenario. Finally, we have tried different techniques to maintain the context of the conversations which makes the conversation with the chatbot much more natural.

We have tested, and analysed different components of the algorithms used this framework for better performance and scalability. The end product was implemented on a server with limited resources to push the optimization. Web and mobile application were made to test its performance in real life.

The result of this work provides a performance wise optimized framework for a chatbot which can provide contextual answers to various queries of customers without using grammar induction, morphological segmentation, part-of-speech tagging, terminology extraction, lexical analysing, etc like resource heavy natural language processing techniques. This framework is only dependent on training dataset and a dictionary of that language which makes it easily adaptable across various languages. This model can easily adopt to user data and can consistently improve overtime.

**Related works**

In recent years various works have been done to test various aspect of conversational AI bot.
Eisman, Navarro, and Castro (2016) presented a method to build a multi-agent architecture for conversations that can analyse heterogeneous data sources despite the data being scattered across different sources and with different format and structure. However, this model relies heavily on thousands of regular expressions, Natural language processing and rule-based architecture which is not an ideal model for ease of portability of the structure.

In the next paper, Chakrabarti and Luger (2015) built a chatterbot that can contextualize the conversation and overcome the limitation of utterance-exchange type conversations. This bot is specialized for handling customer service. But it relies heavily on finite state automata, semantic analysis and complex goal fulfilment map which make it very complicated to implement for a domain. Kandasamy, Bachrach, Tomioka, Tarlow, and Carter (2017) presented a method of a policy-based batch gradient method that can help a batch reinforcement learning to train chatbots. This algorithm efficiently used minimal labelled data. However, the lack of labelling increases the constant need of human supervision which decreases the usefulness of a chatbot (Kandasamy et al., 2017). Setiaji and Wibowo (2016) proposed a way to store the knowledge base of a chatbot in a relational database. They structured the database as an entity-relationship diagram and pattern matching was done using structured query language. The problem with this architecture is that it relies heavily on its database. But to deploy a chatbot, a vast knowledge database might not be present in the first place. Another problem with this model is that it is too sensitive to the misspelling of words. Li et al. (2017) introduced a reinforcement learning framework that can train a neural network to create responses by stimulating dialogues created between two agents. The model was straightforward but diverse enough to create a sustainable conversation. This model requires a balanced reward system to perform accordingly for a given scenario.

Chang, Yin, and Zhang (2017) developed a sequence to sequence model framework for advertisement recommendation, query understanding and user interaction. They gained some significantly better result in query rewriting that maximizes information gain. It requires correct grammatical sentences to perform well. Pavlić, Dovedan Han, and Jakupović (2015) developed a conceptual framework to represent text expressed knowledge. They had to un-contextualize the text for processing, and that is the limitation of this work. Yue et al. (2017) improved dynamic memory networks to build a text-based QA system that can simultaneously extract global and hierarchical salient features. Their system performed better than previously used dynamic memory networks both in term of stability and accuracy. This approach requires a dense dataset to answer properly.

The goal of our work is to overcome these limitations like DBMS dependency, lack of context awareness, dependency on rule-based structures like finite state automata, constant human supervision, etc. The focus of this work is to make a cross-language context-aware chatbot that can be easily implemented for any domain without making custom rules for each use case scenario to increase the ease of use.

Development

**Designing the training data**

Designing a domain-specific contextual chatbot requires a dataset to test and validate the performance of that architecture. The data characteristic is needed to be versatile;
covering various topics in several depths. Because of these reasons, we chose to build the bot around an electronics shop for both training and testing purpose. The reason for selecting electronics shop as a specified domain is that – electronics shops have product from multiple categories from multiple brands which ensure the breadth of the dataset. Not only that but the typical conversation with the customers can get into several depths including the price, specifications, warranty, delivery, etc. related information. This kind of multi-depth conversation is necessary to test out the contextualization capability of the framework.

After consulting with various electronics shop owners, we narrowed down the dataset to cover 91 topics total. These include two main categories – products and services. And branched into subbranches like- brands, specification, warranty, etc. on products side. Delivery, discount, address, payment information, etc. on the services side. The dataset was built using two different language – Bangla and English. Bangla represents an example of a resource-poor language for which no standard dataset, stemming method, NLP library, and word list is available in any programming language. On the other hand, English has all kind of support for implementing and testing various methods in any standard NLP library.

The dataset information was retrieved from the official website of various electronics shop (Chowdhury Electronics, 2017). The sentence structure and query style were retrieved from the official Facebook page of the consulting companies. The translation method between the two methods was oblique translation without altering the base information.

We structured this data into this following model.

**Tag:** a unique name for each topic that the model covers

**Sample Patterns:** Example of sample query of that topic

**Possible Responses:** Sample responses of the topics. After identifying the topic, one of these responses will be randomly chosen as an answer to the query

Then some basic contextual elements were added.

**Current Context:** Changes the state/context of the conversation, if needed.

**Related Context:** Helps to filter out the result based on the context set

Sample instance of data is shown in Table 1.

This model to structure chat data has two main advantages. Firstly, if someone has existing chat data, it can mark every query to associate with a tag. In this way, already existing query data and response can be attached to the training dataset. Secondly, if someone has no prior dataset; anyone can create and add data in this format. There is no hassle to

| Table 1. Sample data set. |
|---------------------------|
| **Tag**                   | Sony_TV_Price_WC  |
| Sample Patterns           | ‘Price?’, ‘How much?’, ‘What is the cost?’ |
| Possible Responses        | ‘Sony 24-inch TV costs 22 thousand taka’, ‘Sony 32-inch TV costs 27 thousand taka’, ‘Sony 40-inch TV costs 34 thousand taka’ |
| Current Context           | ‘Sony_TV’ |
| Related Context           | ‘television’ |
specify the flow of the conversation. If anyone wants to mark a conversation related to other information; it was kept track through ‘Related Context’ tag. All the inputs in these 5 input categories can be set by the person who wants to build the chatbot through this framework. We can see that there is no need to preset any conditional flow of the conversation. As every class in this structure of the dataset is discrete; if new data occurs, it can be easily entered one these following classes or in a new class without worrying about maintaining the relationship between the new data with other classes. Finally, total 1731 pattern queries in total were added to these 91 topics. Same data was translated in both English and Bangla.

**System design**

In this section, we are going to explain the approach of our system design.

Firstly, we have a domain-specific data to train the chatbot. The data is pre-classified, and every class has some sample examples and some other data to track the flow of the conversation. So, when a new query comes from a user, we need to associate the query to our pre-existing classes based on the sample that already exists in each class. Based on that topic identification; we give the user appropriate answers to their queries (Figure 1).

So now the whole problem becomes a text classification problem. The only difference between this and a standard text classifier scenario is the corpora characteristic. Unlike regular text classification, the chatbot dataset has a vast majority of classes under the same domain. Each class has only 5–10 sample sizes, and each sample contain only one sentence instead of a whole paragraph or document. Even the input sentence is one or two lines of sentences. This sparsity of both training sample and test data makes it difficult to classify the data even for the well-established text classification techniques.

To find out the most suitable methods to tackle this problem, we divided the query identification process into two major part – Pre-processing and Text Classification.

Pre-processing is needed to avoid variation of same words to be considered as different, and both training and testing examples need to go through the same process. Our goal here was to keep the system free from language-specific dependency as much as possible.

After the pre-processing; different text classification methods were tested against our dataset to find out the effectiveness of different algorithms for finding out the best possible relation between a new query and already existing dataset. The primary goal here was to find out the algorithm which can classify queries within the constraints of limited sparse data and limited computational resource.

![Figure 1. Chatbot structure.](image)
So as shown in Figure 2 a dataset is designed to train the chatbot, it will go through the finalized preprocessing system for stemming, and then it will train the classification model which will be used for identifying the class label of a new query after preprocessing to give it appropriate response as shown in Figure 3.

After testing different algorithms in both sections and comparing results with standardized methods, we have tried to find out the optimal solution for building the chatbot for any resource-poor languages.

**Development**

*Selecting and designing the preprocessing model*

In an information retrieval system stemming is an important preprocessing step. The goal is to minimize inflectional forms of words and convert derivationally similar form of words to a common base form.

According to Jivani, stemming techniques can be classified into three types of methods- truncating, statistical and mixed and for the English language various truncating techniques such as Lovins, Porters, Paice/Husk, Dawson, etc. are the most popular ones because of their resource light nature (Jivani, 2011). However, the main drawbacks of these techniques are harder implementation and rule-based structure. Because of this for a language independent model, they are not suitable at all.

Among the statistical stemming approaches HMM stemming is complex and over stemming may occur in this method. On the other hand, YASS stemmer requires significant computing power and deciding the threshold can be difficult (Jivani, 2011). These problems lead us to use N-Gram Stemming technique which is also a statistical approach, but the main advantage is – this method is entirely language independent because it only uses a word list to find the stem (Mayfield & McNamee, 2003). No language-specific rules are needed for this method, so it works universally for all the language if the given word list contains the necessary words.
As we were using two languages to build and test our model – to perform N-Gram stemming we used the Kolkata Bangla Academy Dictionary for Bangla Language which contains about 98525 words (Bangla Word List, n.d.). However, for the English language, we kept the traditional widely used porter stemming to see the end result performance difference between a resource-poor language and a resource-rich language.

So, we built a system to find out the maximum N-Gram match from the list for any given words.

There are two approaches we have tried to implement N-Gram maximum string-matching stemming algorithm.

**Dynamic programming**
The dynamic programming approach to find max N-Gram string is given below.

```
INPUT:
D = {W1, W2 W3 ...}: Collection of root word list
W: Input word
OUTPUT:
Stem: appropriate stem of W
Stem_length = length of the stem size initially zero
for every word Wc in D do
  Lc = length of Wc
  Li = length of Wi
  LCS[Lc][Li]: Two-dimensional array initialized with 0
  longest: variable initialized with zero
  started_from: variable initialized with zero
  for x = 1 to Lc do
    for y = 1 to Li do
      if Lc[x] == Li[y] then
        LCS[x][y] = LCS[x-1] [y-1] + 1
        if LCS[x][y] > longest then
          longest = LCS[x][y]
          started_from = y
      else
        LCS[x][y] = 0
      if longest > stem_length then
        Stem = Wc[started_from to longest]
        Stem_length = longest
  return Stem
```

This method is the straightforward approach to N-Gram stem using dynamic programming. If we observe the outer loop, it is iterating over the whole word list for every stemming. But the dictionary size is enormous for every language. There can be millions of words presented in a dictionary. So, if the number of iterations needed for each stemming goes to million, it might not always be a feasible solution to use N-Gram stemming in this way. So, we tried different architecture to make it more suitable for our purpose.

**Trie data structure**
If we look closely to our problem, we can see that for a given word we are trying to match its’ max N-Gram to the N-Gram of the words in the dictionary. This dictionary is static for all the stem. The only variable here is the query word. We used this property to speed up the N-Gram stemming process by using the Trie data structure. Trie is a suffix tree which stores
a string in a link list structure shown in Figure 4. This data structure searching is not dependent on the size of the dictionary.

So, to find the max N-Gram, we inserted all n-grams of the words in the dictionary to this suffix tree. Max number of n-gram for a word of length t can be t * t. If the dictionary size is L, the time complexity for building the suffix tree is O (L * t * t) which is the same as the dynamic programming. But this time this is a onetime process. So, for every query, it will take almost O (1) time to find out the N-Gram stem as the average word length is roughly 10. So this is a time efficient solution for stemming words for all languages. Only the problem is that it is not a memory efficient solution. But because of the limitations of other stemming techniques, we opted for this stemming method.

We tested both of these methods in terms of training data preprocessing and also testing query preprocessing to find out if the second implementation made any significant impact.

**Designing the text classification model**

**Method 1: Support vector machine (SVM)**

SVM is one of the most popular text categorization methods based on Structural Risk minimization principle which was introduced by Vapnik (Burges, 1998). We made a bag of words which is a vector representation of specific word occurrence (Zhang, Jin, & Zhou, 2010). The target was to associate unique words with unique intent by considering the occurrence of each word in a certain classification. So, the input was a vector of numbers; where zero indicated the word is not present in the current input and nonzero number indicated the word occurrences in the current sentence. Since specific words of all training intents were included in the vector, it became very sparse for each training intent. As a result, most of the elements in the vector were zero. For building the linear SVM, we started with a combination of ‘hinge’ loss function and ‘L2’ regularization method.

Next, we substituted that combination with another combination of ‘hinge’ loss function and ‘elastic net’ regularization method based on our result and previous studies (Joachims, 1998; Özgür, Özgür, & Güngör, 2005). With that SVM, we found almost the same
results on the dataset. Then instead of changing the regularization method, we opted to change our loss function. According to previous related studies, ‘logistic’ loss function can deal with outlier and sparse data better than ‘hinge’ loss function (Rosasco, De Vito, Caponnetto, Piana, & Verri, 2004).

So, we have tried out two suitable loss function – hinge and logistic and two different regularization methods- L2 and elastic net to find the best result attainable.

**Method 2: centroid K nearest neighbour (CenKNN)**

Pang, Jin, and Jiang proposed and evaluated a way named CenKNN which works better in reducing sensitivity to imbalanced class distribution and avoids noisy term features. They experienced better performance than KNN and other scalable classifiers like Rocchio and Centroid. It even outperformed Support Vector Machines in a situation with limited classes with imbalanced corpora (Pang, Jin, & Jiang, 2015).

**Data representation**

To implement and test this architecture we represented our data in Vector Space Model which is used widely in various text classification model (Salton, Wong, & Yang, 1975). Our dataset is represented as a set of string-class tag pair like

\[
\text{Dataset} = \{(S_1, C_1), (S_2, C_2), \ldots, (S_q, C_q)\},
\]

here \(C_i\) belongs to the predefined tag which represents the topic of the sentence. \(C_i \in \{\text{Tag}_1, \text{Tag}_2, \ldots, \text{Tag}_m\}\).

Each sample sentence \(S_i\) is represented as a term vector where each index represents one unique term/word, \(S_i = \{\text{word}_1, \text{word}_2, \ldots, \text{word}_n\}\) and the weight of each dimension of this vector is defined by TF * IDF where TF = Term Frequency and IDF = Inverse Document Frequency.

Here, \(q\) is the number of total sample sentence in all classes, \(m\) is the number of tag/class, and \(n\) is the dictionary size. For our dataset \(q = 1731, m = 91\) and \(n = 1682\) because there was 91 class with 1731 with 1682 unique stemmed words present in the dataset.

This method uses two algorithms consecutively for classification.

The first algorithm is CentroidDR which is used to calculate the centroid of each class and then maps the sample sentences to the vector space based of the class centroid using the cosine similarity measure function. The algorithm is shown below.

```
Input: Given a set of training sample strings
Output: A projection of the data based on the centroid of the classes. \(D^* = \{(x_1, C_1), (x_2, C_2), \ldots, (x_q, C_q)\}\)
1. Determine the class centroid via this formula
   \[
   \text{Centroid}_k = \frac{1}{|\text{Tag}_k|} \sum_{S_i \in \text{Tag}_k} S_i
   \]
2. For each sample sentence, compute the similarity between all centroids and the sentence and state these similarity values to the dimension of the projected data using this formula.
   \[
   x^{(i)}_{(j)} = \text{sim}(S_i, \text{Centroid}_j) = \frac{< S_i, \text{Centroid}_j >}{\|S_i\| \|\text{Centroid}_j\|}
   \]
3. Retrieve the projected vector \(D^* = \{(x_1, C_1), (x_2, C_2), \ldots, (x_q, C_q)\}\)
```
The second algorithm CenKNN is used to classify a new sentence based on the projected data $D^*$ we retrieved using CentroidDR. The algorithm is shown below.

By following these steps described by Pang, Jin and Jiang we implemented and tested CenKNN on our dataset and tested it with our test data (Pang et al., 2015).

| Input: Given the projected data $D^*$ from training dataset and a test query $S_t$ |
| Output: The intended tag/class for the new query $S_t$ |
| (1) Normalize the projected data vector $D^*$ |
| (2) Based on the normalized vector $D^*$ build a k-d tree |
| (3) Project the test query $S_t$ on the class-centroid-based space. After normalizing it, we will obtain new vector $x_t$ |
| (4) For the vector, $x_t$, search the K nearest neighbours on the k-d tree. |
| (5) Classify the $S_t$ based on the KNN rule based on the following formula |

$$
\text{Class} = \arg \max_{x_i \in \text{KNN}_{x_t}} \left( 1 - \text{dist}(x_t, x_i) \right) \times I(x_i, \text{Tag}_j)
$$

Here $\text{KNN}_{x_t}$ indicates the set of K nearest neighbours of $x_t$, $\text{dist}(x_t, x_i)$ indicates the Euclidean distance between this two vector, and $I(x_i, \text{Tag}_j)$ indicates the indication function which returns 1 when $x_i$ belongs to $\text{Tag}_j$.

**Method 3: artificial neural network**

For building the chatbot architecture we focused and tested several variation of Feed Forward Neural Network instead of a Feedback Network. The main reason behind it is that our data set is discrete. As it was not designed as complete conversation session structure, it could not be used to train recurrent neural network which works better where the input is given sequentially, and output depends not only on the input but also on the sequence of the input. So, our input characteristic was suitable for Feed Forward Neural Network, and we used different variations of it to find the suitable neural network model.

We needed to restructure our dataset for using in Neural Network according to its structure. Basic feed forward neural network without convolution consists of multiple layers of small computational nodes. The first layer is the input layer, and the final layer is the output layer. All the middle layers are considered as the hidden layer. The computational nodes output values based on activation functions and they try to readjust every edge based on the given training data set (Figure 5).

The input layer is an R dimensional vector. $W^1$ is a matrix of weights that connect nodes to input $a^1 = f^1(W^1 p)$ … Here $f^1$ is a linear function for our network. Circle with plus sign means the product of $W^1$ & $p$. Function $f^2$ represents softmax/CReLU/ReLU and $f^3$ represents regression. As the input is an R dimensional vector, we needed to represent the data set in a vector space containing numerical values. We chose to use the method bag of words for data representation (Zhang, Jin, & Zhou, 2010). In this method, every sentence is represented in a single dimensional vector space containing binary values, where

![Figure 5. Feed forward neural network (Hagan, Demuth, Beale, & De Jesus, 2016).](image)
each index represents a unique word. The value of this index is zero for non-existent words in the input sentence, and one indicates the presence of the word in the input.

The output layer defines the output of the neural network. As our output is completely discrete because we are trying to predict the tag; we tried to estimate the probability of the relevance of each topic. This unique characteristic converted our scenario as a logistic regression problem. As softmax regression is suitable for this kind of multiclass prediction, we used it for our output layer. The format training output is a similar kind of vector like in input format, but this time instead of representing the existence of the word, each index indicates the associated class.

The main characteristic of a feed-forward network is defined by the activation function in its’ hidden layer. Some activation functions can work better or worse based on the given scenario, and we have tested most of the up to date and reliable ones for finding the suitable solution for our chatbot model. We have tested with six variations of activation functions, namely (i) linear activation, (ii) rectifier linear, (iii) leaky rectifier linear, (iv) parametric rectifier linear, (v) exponential linear, (vi) concatenated ReLU. More details of these could be found elsewhere (Clevert, Unterthiner, & Hochreiter, 2015; Dahl, Sainath, & Hinton, 2013; He, Zhang, Ren, & Sun, 2015; Maas, Hannun, & Ng, 2013; Shang, Sohn, Almeida, & Lee, 2016; Wu, 2017).

So, all these activation functions were tested with various numbers of hidden layers to find the best performance overall while implementing the chatbot with the Feed Forward model.

By using the convolutional neural network, some researchers are getting comparatively better results for classifying text. Ladilav and Pavel found out that even on Czech standard corpora CNN performs slightly better in Multi-label document classification than Feed Forward Network (Lenc & Král, 2017). Convolutional Neural Network is a special type of neural network where in addition with fully connected layers in traditional feedforward network, there are other additional layers (Figure 6).

In query classification, the convolutional layer extracts feature from the input query. It tries to filter out the important features in the sentence by analyzing the sentence in chunk by chunk. This enables the neural network to find out the correlation between words and set different importance in each feature word. Another additional layer in CNN is spatial pooling which reduces the dimensionality of the feature input but retaining the most important information. The way we designed our CNN was inspired from the design by Kim (2014). The input of the network is the same as the Feed Forward Network, a bag of word. Then in the second layer, we designed an embedding layer which represents the input query into a vector space with M row and N columns; where N is the size of

Figure 6. Convolutional neural network (Britz, 2015).
embedded vectors. In the next layer, we used our convolutional layer with the kernel size \( Q \times 1 \) which is a one-dimensional convolution over the embedded vector and each time it takes \( Q \) input words. The next layer was designed to perform spatial pooling. Then it was connected to the fully connected layers presented in the feedforward network, and the rest of the architecture was kept same. The different variables regarding convolution size, pooling size and embedding was tested and fine-tuned for maximum performance.

We tested this CNN over our test data and compared the result with other architectures.

**Implementing contextualization in prediction model**

While chatting, people tend to ask questions based on previous interactions. So, every query is not self-sufficient on their own; they have relation with the previous conversation. For saving the context of the conversation for better query answer we have tried different contextualization.

**Linear contextualization**

This method considers context as a linear list and only saves the context of the last conversation. As the neural network predicts answer and consider more than one tag; the highest probable tag which has the relevant context dependency gets the priority here (Figure 7).

But after implementing and testing, we found a problem with this method. This method works for most of the cases. But in real life scenario, context is not linear, but a tree-like structure shown in Figure 8. This linear contextualization faced a problem if context switch in the same domain but different branch. It failed to capture appropriate context there.

For example, if anyone goes into the conversation this way –

\[
\text{Product} \rightarrow \text{Brand 1} \rightarrow \text{Price}
\]

While asking price, if no brand is mentioned linear contextualization can assume that the query is about brand 1. But after that, if the person switches to Brand3, it will lose track which product he is talking about.

To solve this problem, we implemented a different method.

**Lowest common ancestor**

While searching for the appropriate context, in this method, we considered the lowest common ancestor as a deciding factor.

![Figure 7. Linear contextualization model.](image-url)
Lowest common ancestor means in a tree if we take two nodes and run a depth-first search on their parents, LCS will be the node which is the first common parent shared among them.

If we look at the previous problem switching from brand 1 price to brand 2 product info can lose track of the product itself if we save one step of linear data. Because before the brand 2 node; the price of brand 1 occurred. So, the only stage we saved in linear contextualization is the price of brand 1 which already lost the information about the product we are talking about.

Product → Brand 1 → Price → Brand 2

But if we consider lowest common ancestor of this case we can see that; the lowest ancestor between current node ‘Brand 2’ and previously saved stage ‘Price of Brand 1’ is ‘product.’ And this converges that the query about Brand 2 is about the same product as Brand 1. In this way, we created the domain sense in the chatbot.

This resulted in much better context capture sense in the bot.

**Result analysis**

We have conducted three separate testing to evaluate the performance of the query identification engine and the contextualization accuracy.

For comparison, we have decided to use precision, recall and f measure for measuring the performance of each of the segment of the whole framework. They are renowned techniques for measuring performance especially in pattern recognition and information retrieval tasks (Powers, 2011).

The first experiment was designed to analyze the performance of the different classification techniques in terms of query identification in a real conversation scenario. The dataset to recreate an unbiased conversation related to product query was built by 17 users from different educational background and age group. They were completely unaware of the system architecture to avoid biases. They were provided with 91 topics covered by our chatbot and asked them to ask questions related to those topics. This random data generation resulted in 769 different queries covering all the 91 topics presented in the training datasets. This test data was kept independent and never used in any part of the whole training process.
This dataset is used to measure the performance of different stemming techniques and query identification accuracy.

As described in the development section N-Gram stemming was used for the preprocessing part, and it was implemented using both dynamic programming and trie data structure. We have measured the time for preprocessing the whole training data and also average execution time for per query.

The performance difference we have found is quite significant as we can see in Table 2. The execution time for stemming the training data of Bangla is slow (2183.193 s) compared to the English training data (0.0931 s). But using of trie lowers down this execution time quite significantly. Another thing we have observed in the per-query execution time. For Bangla, using LCS takes about 2.817 s/query. Which is really not feasible execution time just for preprocessing. But using trie lowers it down to 0.0016 s/query which is even better than the execution time for English query which is 0.0025 s/query. We have concluded it from here that using a Dictionary based N-Gram stemming is a totally acceptable method for preprocessing if we use Trie data structure. This enables languages not to be affected by poor NLP backbone as N-Gram requires no custom rules to stem a word.

After deciding on the preprocessing part, we moved towards analyzing the prediction model. Firstly, while testing SVM, there are many ways to determine the methodology for Loss function and the regularization part. We have tried several of them and found that Hinge Loss function with L2 regularization and Logistic Loss Function with Elastic Net regularization were the two combinations which are giving better result among other combinations. But SVM performs poorly in our dataset. With an F measure score around 20.5%–27.03% it is not suitable for use in our model. Detailed result is shown in Table 3.

As Pang, Jin and Jiang have found out that using a combined method of K Nearest Neighbors and Centroid classifier can perform better in terms of execution time and classification than both KNN and Centroid (Pang et al., 2015). Most importantly they have also concluded that it performs better than Support Vector Machine if the dataset contains limited classes with imbalanced corpora which is our case. So, we have decided to test their proposed methodology cenKNN (Table 4).

The result we have found is quite promising. The f measure score increases to 62.39% for English test data and for Bangla test data we achieved the score 59.58%. This method performs overall better than SVM in terms of classifying sentences for chatbot model.

| Table 2. Execution time for different stemming method. |
|-----------------------------------------------|
| Language | Data type | Algorithm | Execution time (in seconds) |
| English | Training Data | Porter using Python NLTK library | 0.0931 |
| English | Per Query | Porter using Python NLTK library | 0.0025 |
| Bangla | Training Data | N Gram using LCS | 2182.193 |
| Bangla | Training Data | N Gram using Trie | 6.568 |
| Bangla | Per Query | N-Gram using LCS | 2.817 |
| Bangla | Per Query | N Gram using Trie | 0.0016 |

| Table 3. Performance of SVMs. |
|--------------------------------|
| Loss Function | Regularization | English test dataset | | Bangla test dataset | |
| | | Precision | Recall | F measure | Precision | Recall | F measure |
| Hinge | L2 | 20.4% | 26.5% | 23.05% | 22.1% | 30.7% | 25.69% |
| Logistic | Elasticnet | 22.8% | 33.1% | 27.01% | 23.2% | 32.3% | 27.03% |
Next, we have decided to test out Feed Forward Neural Network for designing the prediction model as researchers are getting the promising result using Neural Networks with various structures as discussed in the related work section. The result improved dramatically as we can see in Table 5.

As activation function is one of the core components that impact the performance of FFN; we have tested various activation functions to find out the performance among them. All of them performed better than SVM and cenKNN. Concatenated Rectified Linear Unit (creLU) performed better than other activation functions with F measure score of 86.72% for English and 88.62% in Bangla as it handles both negative and positive information while enforcing non-linearity and non-saturation.

But adding a convolution layer on top of Feed Forward network can improve the performance of text classifier as found by Kim because it can better identify the local important feature set (Kim, 2014). And our testing shows this improvement too as shown in Table 6.

We have found that after using the convolutional and pooling layer performance improved by 1–5% for every method in normal Feed Forward Network. This performance improvement is a good reason to opt out for CNN instead of normal feedforward network.

If we compare methodologies to build the prediction model, we can see the difference clearly as shown in Figure 9 which represents their relative f measure score.

Based on the relative performance score, Convolutional Neural Network is the clear choice for designing the prediction model for the chatbot among the other 3 methods.

Table 4. Performance of centroid K nearest neighbor.

| Data                | Precision | Recall | F measure |
|---------------------|-----------|--------|-----------|
| English Test Dataset| 60.23%    | 64.72% | 62.39%    |
| Bangla Test Dataset | 57.37%    | 61.97% | 59.58%    |

Table 5. Performance of various activation function in FFN.

| Activation function                      | English test dataset | Bangla test dataset |
|------------------------------------------|----------------------|---------------------|
|                                          | Precision | Recall | F measure | Precision | Recall | F measure |
| Linear                                   | 88.73%    | 85.87% | 87.2%     | 89.87%    | 84.13% | 86.90%    |
| Rectified Linear Unit                    | 86.51%    | 82.87% | 84.65%    | 89.02%    | 82.93% | 85.86%    |
| Leaky Rectified Linear Unit              | 87.55%    | 86.61% | 87.07%    | 88.23%    | 81.89% | 84.94%    |
| Parametric Rectified Linear Unit         | 86.98%    | 84.82% | 85.88%    | 88.94%    | 83.82% | 86.30%    |
| Exponential Linear Unit                  | 85.69%    | 81.35% | 83.46%    | 89.52%    | 85.69% | 87.56%    |
| Concatenated ReLU                        | 89.43%    | 86.72% | 88.05%    | 90.33%    | 86.98% | 88.62%    |

Table 6. Performance of convolutional neural network.

| Activation function                      | English test dataset | Bangla test dataset |
|------------------------------------------|----------------------|---------------------|
|                                          | Precision | Recall | F measure | Precision | Recall | F measure |
| Linear                                   | 92.91%    | 87.87% | 90.31%    | 93.15%    | 86.26% | 89.57%    |
| Rectified Linear Unit                    | 89.25%    | 84.90% | 87.02%    | 92.59%    | 85.53% | 88.92%    |
| Leaky Rectified Linear Unit              | 90.55%    | 89.16% | 89.84%    | 91.03%    | 84.98% | 87.90%    |
| Parametric Rectified Linear Unit         | 89.42%    | 86.48% | 87.92%    | 90.96%    | 85.87% | 88.34%    |
| Exponential Linear Unit                  | 87.96%    | 86.75% | 87.35%    | 93.16%    | 87.26% | 90.11%    |
| Concatenated ReLU                        | 92.40%    | 88.42% | 90.36%    | 94.63%    | 89.08% | 91.77%    |
Another thing can be observed in this comparison. The difference between the performance of Bangla and English language is small. This is a good indication that the different methods applied in Bangla and English have not impacted the overall result. So, using the N-Gram model which works universally for all languages is a good choice for steaming if implemented efficiently.

The second experiment was conducted to do a synthetic test of the classification model on a standard dataset to measure the performance of the prediction model using CNN with crelu as the activation function of the fully connected layer.

We have compared this Convolutional Neural Network model with 20 Newsgroup data (Lang, 1995). Though our model is not designed to deal with this dataset, this is the only public dataset we have found which is divided into various classes, and each class contains sample messages/documents. So, it was closely formatted to the format we were using in our own model. Modifying the input stream, we were able to input the data set in our model. 60% of the dataset was used due to the large dataset size, as it was using an excessive amount of RAM at a time to represent itself in the Bag of Word method. After calculating the validation, we found out our model got 83.3% precision score and 80.39% recall score which resulted in 81.82% f measure score which is good despite handling a different situation which ensures the validity of our framework in terms of class prediction.

Final and the third test was conducted to test out the effectiveness of the contextualization methods.

To test this, 107 complete sessions of conversation was recorded through the test website. They contained 1112 individual sentences, but 87 of them were discarded from the dataset because the relevant data to answer those queries was not present in the training data. The rest 1025 sentences from 107 sessions of conversations were used to see if the error rate of query answering became any better. These sessions had enough query which needed the understanding of contextualization to answer properly.

For this phase of testing, we have took 4 different approaches. The first one is to output the answer based on the highest probability of the class that the CNN outputted as result for any given query. No contextualization is applied here. So, if someone queries – ‘What is the price?’ and the CNN outputs ‘Price of Samsung’ has the probability of 69% and ‘Price of Sony’ has the probability of 62% to be a potential answer; this method will choose ‘Price of Samsung’ regardless the context. The second method only saves the state of the final
query linearly. If any of the output is related to the last query; it is chosen though it might not hold the highest probability as output. So, if ‘Price of Samsung’ has the probability of 69% and ‘Price of Sony’ has the probability of 62% to be a potential answer. ‘Price of Sony’ will be chosen in this linear contextualization method if the last query is about ‘Models of Sony TV.’

The third approach shows the same characteristics as the second one but this time instead of considering the last query it considers the lowest common ancestor between the previous query and all the possible answers. It will give priority to the answer which has the maximum depth of LCA between the last query and that answer.

If we consider the situation shown in Figure 10, we can see that the last query was about Sony. So, when asking for the price out of three probable answers, it will choose Sony because the LCA between ‘Price of Sony’ and ‘Details about Sony’ is the node ‘Details about Sony’ which has a depth 2. On the other hand, the LCA between last query ‘Details about Sony’ and other 2 options are ‘Television’ for both cases which has a depth one. So, this method will choose ‘Price of Sony’ based on the depth of LCA. If depths are the same, it will consider the highest probability for decision making.

The fourth option is to give priority to the probability first then consider the LCA. So, if for some reason one possible answer has the highest depth in terms of LCA and its’ probability is 52% but the among all the possible answers, one answer has the probability of 93%, but the depth of its’ LCA is lower than the one with 52% probability. Because the latter answer has a drastically better probability of being the answer, it will be chosen regardless of the depth of LCA. If the probability is within 10%; it will then give priority to the LCA depth. The threshold 10% is chosen based on some observation among the probability of the similar queries.

We have tested all these four approaches with our session wise data which had 1112 different queries, and the result is shown in Table 7.

As we can see that the error percentage became very low with contextualization and LCA performed better by 6% than the normal linear contextualization. Upon close observation, we have found out that the errors in LCA + Probability threshold were mainly due

![Figure 10. Query scenario.](image-url)
classification errors. None of the errors in this method occurred due to the wrong contextualization.

In Table 8, we can see a sample of the point of failures for different Contextualization in conversational sessions.

The first query was a complete one and recognizable by all the methods. But in the second query, it was not specified if the Sony model is about a TV, home theatre, or phone. So, in no contextualized model; it failed to answer properly. When it jumped to another brand in the third query, the linear contextualization also failed as it already lost track of the original product which was TV. It had the information only about the last conversation which was about Sony. In the last query only prioritizing LCA go confused to answer as it answered about the cost of LG tv, though the CNN showed a greater probability for Home delivery service. But all these scenarios were recognized correctly by the LCA + Probability Threshold contextualization.

So, for the contextualization part, we have decided to use both LCA and the probability given as an output of CNN for better context awareness.

In case of mixed language interactions, this model can handle those scenarios without any problem. As this architecture uses bag of word to mark the presence of the words and use a separate data structure to keep track of the context; if a person switches his language in the middle of the conversation, this chatbot doesn’t lose track of the context. It detects the language accordingly and make proper response according to the language the person is using in his latest query.

If we compare this final architecture of the chatbot with previously mentioned systems in the related work section, the result is promising which is shown in Table 9.

This comparison clearly shows the advantage of using this architecture. The lack of dependency of various constraints makes this architecture much more practical to use and increase its portability across various domains and languages.

**Future work**

Eastern Bank Limited in Bangladesh has already launched a customer service chatbot on Facebook (EBL Dia, 2018) for its users. Though 98% of people in Bangladesh speak in

---

**Table 7.** The error rate in different contextualization.

| Methods                        | Failed to answer | Error  |
|--------------------------------|------------------|--------|
| Without contextualization     | 291              | 26.16% |
| Linear Contextualization      | 163              | 14.65% |
| Longest Common Ancestor       | 94               | 8.45%  |
| LCA + Probability Threshold   | 89               | 8.01%  |

**Table 8.** Points of failure in different contextualization.

| Sample Conversation                        | Without contextualization | Linear contextualization | Longest common ancestor | LCA + probability threshold |
|--------------------------------------------|---------------------------|--------------------------|-------------------------|-----------------------------|
| I want to buy a TV                         | ✔                         | ✔                        | ✓                       | ✓                           |
| What sony models do you guys have          | ❌                         | ✔                        | ✓                       |                             |
| Tell me about LG                           | ❌                         | ❌                       | ✓                       | ✓                           |
| Cost of home delivery?                     | ❌                         | ❌                       | ❌                       |                             |
Bangla, this chatbot cannot support Bangla as it is a resource poor language (Bangladesh, 2018). By integrating our framework in these kinds of scenario and working upon the feedback, we can further improve this work in the future.

The classification technique used in this work can be tested across other related areas like detecting offensive comments, flag inappropriate web articles for children etc. and as this is a language independent model, a proper architecture developed for those specific scenarios can easily be used in any language.

**Conclusion**

In this work, our goal was to build a framework for a domain focused context-aware multilingual easy to implement chatbot for resource-poor languages like Bangla with reasonable performance as the already available architectures are not compatible with this kind of languages.

The experiments show that only using an N-Gram stemming as the only NLP methods used in this architecture does not drag down its performance both in terms of accuracy of the response and the time required for preprocessing. But it enabled this architecture to easily be compatible with every language.

| Architectures                        | Dependency on regular expression | Dependency on complex natural language processing | Dependency on DBMS | Dependency of constant human supervision | Dependency on correct grammatical sentence | Requires dense dataset |
|--------------------------------------|----------------------------------|--------------------------------------------------|--------------------|------------------------------------------|--------------------------------------------|----------------------|
| Multi Agent Architecture (Eisman et al., 2016) | ✔                               | ✖                                                | ✖                  | ✖                                        | ✖                                          | ✔                    |
| Chatterbot Architecture (Chakrabarti & Lugur, 2015) | ✔                               | ✖                                                | ✖                  | ✖                                        | ✔                                          | ✔                    |
| Policy-Based Batch Gradient Method (Kandasamy et al., 2017) | ✖                               | ✔                                                | ✖                  | ✖                                        | ✖                                          | ✖                    |
| Knowledge in Database architecture (Setiaji & Wibowo, 2016) | ✔                               | ✖                                                | ✔                  | ✖                                        | ✖                                          | ✔                    |
| Deep Reinforcement Learning (Li et al., 2017) | ✔                               | ✖                                                | ✔                  | ✖                                        | ✖                                          | ✔                    |
| Seq 2 Seq Deepprobe Architecture (Chatfuel.com, 2017) | ✖                               | ✔                                                | ✖                  | ✖                                        | ✔                                          | ✔                    |
| Dynamic Memory Network Architecture (Dahl et al., 2013) | ✖                               | ✔                                                | ✖                  | ✖                                        | ✖                                          | ✖                    |
| Our Architecture                       | ✖                               | ✖                                                | ✖                  | ✖                                        | ✖                                          | ✖                    |
Using a convolutional neural network for query identification part showed significantly better performance than other available methodology.

Finally using LCA for context tracking showed the promising result in terms of keeping track of the flow of natural conversation.

The simplified structure of the training data made this architecture easily accessible to the general people to deploy a customer service bot with their already existing dataset. If needed, new data can also be created and added to the training phase without any problem.

As a proof of concept, one demo chatbot has been deployed on Amazon Web Server. The setup is a server instance with minimal computational resource such as 1 GB RAM. This demo is accessible to everybody on the website (metsys.io, 2017). As a result, anyone can visit the URL of the website in their web browser and interact with it.

However, we believe there is still room for further research and development. Some other classifier like Naive Bayes etc. can be tested in this framework. There is a possibility that feedback neural networks like recurrent networks can also improve the performance here. More real-world exposure of the chatbot and implementing it for various domains can further evaluate this chatbot framework.

Disclosure statement

No potential conflict of interest was reported by the authors.

Notes on contributors

Anirudha Paul has completed his Bachelor in Computer Science & Engineering from North South University in 2018. He is currently working as a software engineer at Samsung R&D Institute Bangladesh. His research work is primarily focused on news validation and summarization from multiple in native languages, recommendation system based on loosely connected data and human-machine interaction through natural language processing. His research interests include blockchain, distributed system, artificial intelligence and their applications on natural verbal communication processing and modern security system.

Asiful Haque Latif is currently working as a Software Engineer at Enosis Solutions in Bangladesh. He received his B.Sc. degree in Computer Science & Engineering from North South University in 2018. During his Bachelor’s degree studies, he directly contributed to many research projects with topics such as news article summarization, movie recommendation and performance of satellite-based navigation in civil aviation. Currently, his research interests include developing ways to implement natural language processing methods for Bangla, knowledge extraction from unstructured text and application of artificial intelligence in distributed systems.

Foysal Amin Adnan has received his B.Sc. degree in Computer Science & Engineering from North South University in 2018. He is currently working as a Software Engineer at Enosis Solutions in Bangladesh. At the time of his studies, he worked in a number of mobile application projects with LICT and Samsung R&D Institute Bangladesh. He contributed to research projects related to facial recognition, image compression, news article summarization using natural language processing in Bangla and applications of artificial intelligence.

Rashedur M. Rahman is working as a Professor in Electrical and Computer Engineering Department in North South University, Dhaka, Bangladesh. He received his Ph.D. in Computer Science from the University of Calgary, Canada and Masters from the University of Manitoba, Canada in 2007 and 2003, respectively. He has authored more than 150 peer-reviewed research papers in journals or conference proceedings in the area of parallel, distributed, grid and cloud computing, knowledge and
data engineering. His current research interest is in data science, data replication on grid, cloud load characterization, optimization of cloud resource placements, computational finance, deep learning, etc. He has been serving on the editorial board of a number of journals in the knowledge and data engineering field. He also serves as a member of organizing committee of different international conferences.

References

Bangladesh. (2018). [online] Retrieved from https://www.everyculture.com/A-Bo/Bangladesh.html
Bangla Word List. (n.d.). [PDF] West Bengal Bangla Academy, Kolkata.
Botsify.com. (2017). Botsify - create artificial intelligent chatbots without coding. [online] Retrieved from https://botsify.com/
Britz, D. (2015). Understanding convolutional neural networks for NLP. [online] WildML. Retrieved from http://www.wildml.com/2015/11/understanding-convolutional-neural-networks-for-nlp
Burges, C. J. C. (1998). A tutorial on support vector machines for pattern recognition. Data Mining and Knowledge Discovery, 2(2), 121–167.
Chakrabarti, C., & Luger, G. (2015). Artificial conversations for customer service chatter bots: Architecture, algorithms, and evaluation metrics. Expert Systems with Applications, 42(20), 6878–6897.
Chang, K., Yin, Z., & Zhang, R. (2017). DeepProbe: Information directed sequence understanding and Chatbot design via recurrent Neural Networks. KDD.
Chatfuel.com. (2017). Create AI chat bot for Facebook. [online] Retrieved from https://chatfuel.com/
Chattypeople.com. (2017). ChattyPeople. [online] Retrieved from https://www.chattypeople.com/
Chowdhury Electronics. (2017). Chowdhury Electronics. [online] Retrieved from http://www.chowdhurelectronics.com/
Clevert, D. A., Unterthiner, T., & Hochreiter, S. (2015). Fast and accurate deep network learning by exponential linear units (elu). arXiv preprint arXiv:1511.07289.
Dahl, G., Sainath, T., & Hinton, G. (2013). Improving deep neural networks for LVCSR using rectified linear units and dropout. 2013 IEEE International Conference on Acoustics, speech and Signal processing.
EBL Dia. (2018). [online] Retrieved from https://www.ebl.com.bd/home/ebl_dia
Eisman, E., Navarro, M., & Castro, J. (2016). A multi-agent conversational system with heterogeneous data sources access. Expert Systems with Applications, 53, 172–191.
Flowxo.com. (2017). Flow XO: Chatbot platform with easy-to-use editor and 100+ integrations. [online] Retrieved from https://flowxo.com/ [Accessed 28 Nov. 2017]
Fredkin, E. (1960). Trie memory. Communications of the ACM, 3(9), 490–499.
Hagan, M., Demuth, H., Beale, M., & De Jesús, O. (2016). Neural network design. [S. l.: s. n.], p.51.
Hasan, M. (2018). Call centres on the rebound. Retrieved from https://www.thedailystar.net/news-detail-119715
He, K., Zhang, X., Ren, S., & Sun, J. (2015). Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In Proceedings of the IEEE international conference on computer vision (pp. 1026–1034), Chicago.
Jivani, A. (2011). A comparative study of stemming algorithms. International Journal of Computer Technology and Applications, 2, 1930–1938.
Joachims, T. (1998). Text categorization with support vector machines: Learning with many relevant features. Machine learning: ECML-98, pp.137–142.
Kandasamy, K., Bachrach, Y., Tomioka, R., Tarlow, D., & Carter, D. (2017). Batch policy gradient methods for improving neural conversational models. International Conference on learning Representations.
Kim, Y. (2014). Convolutional neural networks for sentence classification. arXiv preprint arXiv:1408.5882.
Lang, K. (1995) Newsweeder: Learning to filter netnews. Proceedings of the Twelfth International Conference on Machine learning, 331–339.
Lenc, L., & Král, P. (2017). Two-level neural network for multi-label document classification. In A. Lintas, S. Rovetta, P. Verschure, & A. Villa (Eds.), Artificial Neural networks and Machine learning –
ICANN 2017. ICANN 2017. Lecture Notes in Computer Science (Vol. 10614, pp. 368–375). Cham: Springer.

Li, J., Monroe, W., Ritter, A., Galley, M., Gao, J., & Jurafsky, D. (2017). Deep reinforcement learning for dialogue generation. [online] Arxiv.org. Retrieved from https://arxiv.org/abs/1606.01541

Maas, A. L., Hannun, A. Y., & Ng, A. Y. (2013). Rectifier nonlinearities improve neural network acoustic models. In International Conference on Machine learning (ICML).

Mayfield, J., & McNamee, P. (2003). Single n-gram stemming. Proceedings of the 26th annual international ACM SIGIR conference on research and development in information retrieval - SIGIR ’03.

Single N Gram Stemming.

Metsys Bot. (2017). Metsys Bot. [online] Retrieved from http://metsys.io/chat/

Morph.ai. (2017). Morph.ai - Chatbots for conversational marketing. [online] Retrieved from https://morph.ai/

Motion.ai. (2017). Motion AI - Chatbots made easy. [online] Motion.ai. Retrieved from http://www.motion.ai/

Özgür, A., Özgür, L., & Güngör, T. (2005). Text categorization with class-based and corpus-based keyword selection. Computer and Information Sciences-ISCIS, 2005, 606–615.

Pang, G., Jin, H., & Jiang, S. (2015). CenKNN: A scalable and effective text classifier. Data Mining and Knowledge Discovery, 29(3), 593–625.

Pavlić, M., Dovedan Han, Z., & Jakupović, A. (2015). Question answering with a conceptual framework for knowledge-based system development “node of knowledge”. Expert Systems with Applications, 42(12), 5264–5286.

Powers, D. M. (2011). Evaluation: from precision, recall and F-measure to ROC, informedness, markedness and correlation.

Rosasco, L., De Vito, E., Caponnetto, A., Piana, M., & Verri, A. (2004). Are loss functions all the same? Neural Computation, 16(5), 1063–1076.

Salton, G., Wong, A., & Yang, C. (1975). A vector space model for automatic indexing. Communications of the ACM, 18(11), 613–620.

Setiaji, B., & Wibowo, F. W. (2016). Chatbot using a knowledge in database. 7th International Conference on intelligent systems, Modelling, and Simulation.

Shang, W., Sohn, K., Almeida, D., & Lee, H. (2016, June). Understanding and improving convolutional neural networks via concatenated rectified linear units. In International Conference on Machine learning (pp. 2217–2225).

Wu, H. (2017). Global stability analysis of a general class of discontinuous neural networks with linear growth activation functions. Information Sciences, 179(19), 3432–3441.

Yue, C., Cao, H., Xiong, K., Cui, A., Qin, H., & Li, M. (2017). Enhanced question understanding with dynamic memory networks for textual question answering. Expert Systems with Applications, 80, 39–45.

Zhang, Y., Jin, R., & Zhou, Z. (2010). Understanding bag-of-words model: A statistical framework. International Journal of Machine Learning and Cybernetics, 1(1-4), 43–52.