RESEARCH ARTICLE

Designing a Chat-bot for College Information using Information Retrieval and Automatic Text Summarization Techniques

Radha Guha1,*

1CSE Department, SRM University AP, Andhra Pradesh, India

Abstract: Background: In the era of information overload it is very difficult for a human reader to make sense of the vast information available on the internet quickly. Even for a specific domain like a college or university website, it may be difficult for a user to browse through all the links to quickly get the relevant answers.

Objective: In this scenario, the design of a chat-bot which can answer questions related to college information and compare between colleges will be very useful and novel.

Methods: In this paper, a novel conversational interface chat-bot application with information retrieval and text summarization skill is designed and implemented. Firstly, this chat-bot has a simple dialog skill; when it can understand the user query intent, it responds from the stored collection of answers. Secondly, for unknown queries, this chat-bot can search the internet, and then perform text summarization using advanced techniques of natural language processing (NLP) and text mining (TM).

Results: The advancement of NLP capability of information retrieval and text summarization using machine learning techniques of Latent Semantic Analysis (LSI), Latent Dirichlet Allocation (LDA), Word2Vec, Global Vector (GloVe) and TextRank is reviewed and compared in this paper first before implementing them for the chat-bot design. This chat-bot improves user experience tremendously by getting answers to specific queries concisely which takes less time than to read the entire document. Students, parents and faculty can get the answers for a variety of information like admission criteria, fees, course offerings, notice board, attendance, grades, placements, faculty profile, research papers, patents, etc. more efficiently.

Conclusion: The purpose of this paper was to follow the advancement in NLP technologies and implement them in a novel application.

Keywords: Chat-bot, natural language processing, text mining, information retrieval, text summarization, topic modeling, latent semantic analysis, latent dirichlet allocation, word2vec, GloVe, word embedding, textrank.

1. INTRODUCTION

This section resumes the need and feasibility study of a chat-bot. Since the advent of the internet in 1990s, it has evolved to several Zetta bytes (ZB) of data portal today and is growing everyday with exponential velocity. Businesses and academic institutes advertise their offerings online to the public for doing businesses more profitably and conveniently. Most of this data is unstructured, written in natural languages like English. Thus, automatic processing of unstructured data needs Natural Language Processing (NLP) and text mining techniques [1-6]. The field of information retrieval from structured databases is well established now. But automatic unstructured data processing field has become imminent only recently as the internet volume has grown rapidly. For a human reader, relevant information extraction and making sense of the large volume of online articles quickly is not possible. Relevant information extraction and summarization of text even from a focused domain of college websites are very difficult for a user. Sometimes students need to compare college information from different college websites before making a decision, like where to take admission.

This problem has motivated the design and implementation of a conversational chat-bot which can answer specific questions of a user more succinctly. Instead of the user clicking through several links to find the relevant data and then reading the whole page, the use of this chat-bot will save much time giving the college website a competitive edge. A bot is a short form of the word “robot” that automates some
activity of human. Like a human can find a relevant article on the internet and generate a summary of the article manually, our chat-bot will do the same automatically for the user. The input and output of our chat-bot are written text for the time being. It is a human-computer interaction software as conversational interface which needs advanced techniques of natural language processing and text mining.

In history, [7, 8] we have seen chat-bots designed for entertainment, for example Eliza, the psychotherapist introduced in 1966 by an MIT teacher. In a simplistic implementation a user query can be matched with the previously saved frequently asked questions (FAQ) and answer template. A user can pose the same question in many different ways and the chat-bot should be able to detect the intent of the query and find the closest match with the FAQ. Later, A.L.I.C.E chat-bot (2000) followed an advanced pattern matching script, based on XML based artificial intelligence markup language, AIMA [2], as shown below in Table 1. In this script, A.L.I.C.E recognized the user name in wild character (*) and used the name during greeting and goodbye to make it a more human-like response.

Over the years, chat-bots are becoming smarter using more complex pattern matching techniques and combining external sources of knowledge other than canned responses only.

Table 1. A.L.I.C.E Designed Using AIML.

```xml
<?xml version = "1.0" encoding = "UTF-8"?>
<aiml version = "1.0.1" encoding = "UTF-8">  
  <category>  
    <pattern>Good Night</pattern>  
    <template>  
      Hi <get name = "username"/>! Thanks for the conversation!  
    </template>  
  </category>  
  <category>  
    <pattern>I am *</pattern>  
    <template>  
      Hello <set name = "username"> <star/>! </set>  
    </template>  
  </category>  
</aiml>
```

Nowadays, we see more use of chat-bots. As early as 1990s, Canadian government is using forecast generator (FOG) chat-bot for summarizing numeric meteorological data in textual template. From the beginning of 2010, virtual assistants like Amazon’s Alexa, Apple’s Siri, Microsoft’s Cortana and Google Assistant are part of our daily lives. Virtual assistants of these tech giants are providing voice enabled chat services. Tech giant IBM’s Watson competed in American TV quiz show Jeopardy in 2011 and emerged as a champion beating all human competitors. The use of chat-bots in the industry and businesses has become a recent trend. Any business selling products can integrate a chat-bot with their main transaction application for answering user queries about their products and services. Travel companies can use chat-bots to collect passenger information and provide them tour, flight and hotel information. The interface can be transactional also where actual purchase of tickets and booking takes place. For example, a uber-bot can take a ride reservation. Similarly, a weather-bot will answer weather related questions. A medical-bot will output diagnosis to symptoms related input. In essence, these days chat-bots can replace human customer service representatives.

The performance of a chat-bot is measured by its smartness of behaving like a human and for how much time it can engage a human in conversation. For determining the intelligence of a computer, Turing test was proposed by British mathematician A.M. Turing in 1950 [7] in his famous paper “Computer Machinery and Intelligence”. A computer passes a Turing test if it can fool a panel of human users as if they are conversing with another human. A program or computer passing a Turing test is declared as artificially intelligent. To encourage research in artificial intelligence field, Loebner prize- annual competition was announced in 1990 for programs that will pass Turing Test [8].

Intelligence is not only retrieving answers by matching user query with a stock of frequently asked question-answer templates. Humans are intelligent at many other levels, for example they can compute arithmetic and logical operations, they have memory, they have perceptions, they can infer and they can continuously learn from new experiences. In order for machines to achieve human level of intelligence, it will take much longer. First of all, computers need the most challenging skill, which is of Natural Language Processing (NLP). In this paper, we first explore the technological advancement in NLP and then apply that in our chat-bot implementation. We have used machine learning algorithm for classification to understand the user’s input intent more efficiently. Our chat-bot is designed to generate answers from scratch, i.e. first retrieving relevant documents from the internet and then generating a gist of the document. For implementing this chat-bot we had to explore machine learning algorithms for classification and current development of underlying technologies for information retrieval and automatic text summarization.

The organization of this paper is as follows. Section II describes implementation details of dialog skills of the chat-bot. Section III describes implementation details of search skill of the chat-bot by exploring advancement in information retrieval technologies like LSA, LDA, Word2Vec and GloVe. Section IV describes implementation details of text summarization skill of the chat-bot by exploring Text Rank algorithm. Section V concludes the paper with future scope of research.

2. CHAT-BOT DESIGN STEPS

A chat-bot has to be smart and user friendly. An intelligent chat-bot should have many different skills. In this chat-bot, dialog skills, search skills and text summarization skills are added. For designing the college information chat-bot, the IBM Watson Assistant [2] general design principle is followed for the dialog skill. The actual conversation varies from application to application. Let’s look at the comparatively easy dialog skill first for the college information chat-bot. Dialog skill comprises of series of nodes of user intents and corresponding response options. As a user can ask the same question in many different ways, they all are grouped together to create an intent. So intent is a collection of user queries which mean the same thing. For example, user can use the
words “hello”, “hi”, “good morning”, or “kia ora” etc. and the chat-bot should recognize it as #greetings intent and respond appropriately.

Even the chat-bot’s response should vary from time to time to sound less robotic and more human-like. So for each intent a number of responses are prepared and one is selected randomly or sequentially. Even if user input does not match exactly to the “intent” training data, artificial intelligence analytical skill in the chat-bot will use classification algorithm to classify the user input to the closest intent and respond accordingly. So if the user says “aloha” the chat-bot should recognize it as #greetings intent as well.

For a chat-bot to answer many questions, a number of intent nodes such as #greetings, #about_the_college, #placement, #fees, #courses, #thank_you, #goodbye, #anything_else etc. are created, which are perceived to be the most frequently asked questions for the college information chat-bot. For any unrecognized query an external search is triggered. So that information can be retrieved from the internet as per the query requirement and then it is summarized by triggering the summary skill of the chat-bot.

For moving from generic response to more specific response some entity variables are captured from user input if they are there. Suppose a user has a question about #placement intent of a particular program, then the program name is saved as @ program entity. This program entity can have multiple values like "CSE", "IT", or "Civil" etc. The entity value helps giving more specific response by the chat-bot. If an entity value is not already present in user input the chat-bot will ask for it from the user to give specific answer.

The entity values can also be stored as $context variable to last for the entire conversation. Another user friendly concept of “digression” is used in the dialog flow design which prevents the chat-bot to ask the same question from the user again and again if the user cannot answer it. This makes the behavior of the chat-bot more human-like. Fig. (1) shows how the chat-bot converses with the users (left side) internally, and how it processes user input and triggers external search skill (right side).

After describing simple dialog skill in this section, more complex search and summarization skills of the chat-bot are explored in the next sections.

3. INFORMATION RETRIEVAL

Search skill of a chat-bot means automatic information retrieval from the internet. Automatic information retrieval field involves Natural Language Processing (NLP) with Machine Learning (ML) algorithms. This is a challenging field as text data is unstructured or semi-structured and needs to be converted to structured data before they can be processed by computer. To impart structure a corpus of documents is first represented as a Term-Document Matrix (TDM). For a typical corpus of documents on the internet, a TDM will have 100,000 rows for each unique term and 1000,000 columns for each document. Some dimensionality reduction of the TDM is achieved by eliminating the stop words, and by stemming and lemmatization of words. Each cell in TDM keeps the number of times a word or term appears in a particular document. This Term Frequency (TF) of a TDM is also weighted with another number called Inverse Document

![Fig. (1). Dialog Skill (LHS) and Search Skill (RHS) of College Information Chat-Bot. (A higher resolution / colour version of this figure is available in the electronic copy of the article).](image-url)
Frequency (IDF). As occurrences of a term in many documents reduce its discriminative power, the TF is weighted with IDF. If \( d \) is the number of documents in the corpus and \( d(t) \) is the number of documents having the term \( t \), then IDF = \( \log(d/d(t)) \). Low IDF value means a term has less distinctive power and vice versa. So each cell in TDM keeps TF*IDF score.

This way, an unstructured text corpus is converted into a structured data model. This is also called a vector space model as every row is a fixed length word vector indicating a word appearing in which document, and with how much importance. Every column is a fixed length document vector indicating which words represent it. This is also a bag of words concept, where word ordering is not important. In Vector space model, document ordering is also not important.

After an unstructured corpus is transformed into a structured corpus, other analyses like document to document similarity can be computed by cosine distance measure. The similarity between the documents are measured by cosine of angle \( \theta \) between two documents vectors \( A \) and \( B \) where cosine \( (\theta) = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{A^2} \sqrt{B^2}} \).

But natural language like English has the problem of “synonymy” and “polysemy”. For example “ocean” and “sea” are synonymous as they both mean the same thing. The word “bank” is polysemous as it may mean river bank or financial institution or heap. Therefore, for information retrieval if the query words are tried to match exactly with the documents in the TDM, the retrieval may not be that relevant. The problem of synonymy retrieves fewer number of relevant documents and the problem of polysemy retrieves more documents even though some of them are irrelevant. A good IR system should understand the semantic intent of the query. In this section, emergence of better algorithms for NLP, which applies two different concepts such as topic modeling and word embedding are explored. LSA-1990 and LDA-2003 use topic modeling concept [9-15], whereas, Word2Vec-2013 [16, 17] and GloVe-2014 [18] use word embedding technique which preserve the local and global context of words respectively.

### 3.1. Topic Modeling: LSI and LDA

Taking cosine distances from the raw term document matrix (TDM) perform poorly for document similarity calculation because of the “synonymy” and “polysemy” problem of English language. So a better method is required. We know that in a library books are grouped either subject or topic wise, so it becomes easy for a user to find a particular book. However, on the internet there is no such physical organization of information. So better algorithms, like Latent semantic indexing or analysis (LSI/LSA) and Latent Dirichlet allocation (LDA) try to find the topics in a document so that the documents can be grouped topic wise. A topic is expressed as a group or cluster of words that occur together in a document or multiple documents. These clusters are soft as they overlap with each other i.e. a word can belong to more than one topic. LSA and LDA try to find topics and topic proportion in each document so that document similarities can be measured more effectively as required for information retrieval.

### 3.2. Latent Semantic Analysis (LSA)

LSA [9, 14] addresses the synonymy problems of vector space model. LSA was developed by T. Landauer, P.Foltz and S.Dumais in 1990’s at the University of Colorado, Department of Psychology [Deerwester et al., 1990]. LSA mimics the human way of word grouping and topic categorization. A human will judge that there are fewer topics in a document or corpus of documents than the number of unique words in the corpus. She may think that words like (ocean, sea, water, boat, ship) all belong to one hidden topic named “aquatic vehicles” and words like (ocean, sea, water, fish, whale, shark) all belong to a second hidden topic named “aquatic animals”.

In LSA, co-occurrence of words is statistically measured for latent semantic or topic determination, and so is the name of the algorithm. LSA uses singular value decomposition (SVD) algorithm to factorize the raw TDM into three matrices (Fig. 2). In SVD any \( m \times n \) matrix \( A \) is decomposed as \( U_m \sum V_n \). Here \( r \) is the rank of the matrix \( A \) and \( r < \min(m, n) \). In SVD decomposition \( U \) and \( V \) are orthonormal i.e. \( UV^T = 0 \) and \( UV = 1 \) and \( VV^T = 1 \).

For a TDM \( A \), \( U \) is a term by topic matrix and an element in \( U \) depicts how strongly a term belongs to a topic. A row in \( U \) is a lower dimension, dense vector representation of a word. \( U \) is an orthogonal matrix with left singular vectors of \( A. A^T \). Similarly, \( V \) is a topic by document matrix and an element in \( V \) indicates how strongly a document relates to a topic. A column in \( V \) is a lower dimension dense vector representation of a document. \( V \) is an orthogonal matrix with right singular vectors of \( A^T A \). \( \sum \) is a diagonal matrix of singular values for \( A \), ordered in decreasing order. Each singular value represents the importance of the corresponding topic in the corpus. Now this SVD decomposition can be reduced to lower rank by eliminating the smaller singular values in \( \Sigma \) matrix and keeping only \( k \) larger singular values. This dimensionality reduction works as noise removal by eliminating detail, and less prominent topics of the corpus. In reality, when the original TDM dimension is 100,000 x 10,000, \( k \) value usually ranges from 100 to 300 to represent a good approximation of the corpus topics. The approximated matrix is computed as \( U_m \sum_{k} V_n \). Thus a huge dimensionality reduction is possible in LSA where documents relate better to fewer prominent topics in the corpus.

### 3.3. Information Retrieval

In LSA, the pseudo query doc is mapped to the reduced vector space as \( q_k = \sum_{k} U_k q_k \). Similarity of \( q_k \) can be computed with all other documents in the reduced matrix. In LSA information retrieval, similar documents are ranked as per their cosine similarity measures.

LSA is an algebraic model and has several shortcomings. It is not a probabilistic model of term occurrences. Also it is not a generative model. Here the topic embeddings are not interpretable and may have arbitrarily positive and negative
values in $U$ and $V$ matrix. Another disadvantage is, LSA needs a very large set of documents and vocabulary to create a model to get accurate results. So LSA model needs to be regenerated after regular intervals so that query documents can relate better with the growing corpus in the internet. Also deciding on the best $k$ value (number of topics) needs some domain knowledge and adjustment. LSA takes care of synonymy problem of English vocabulary but it cannot account for polysemy problem of the English language.

Latent Dirichlet Allocation (LDA) is a more well defined generative and Bayesian probabilistic topic model, which is explored in detail in the next section. LDA is also better interpretable by human.

### 3.4. Latent Dirichlet Allocation (LDA)

LDA [10-14] was developed by David Blei, Andrew Ng, and Michel Jordon in 2003 [Blei et al., 2003]. LDA is a probabilistic, generative model in which documents can be generated by following probability distribution over topics and then probability distribution over words for each topic. LDA generates soft clusters of topics which overlap as a participating word can belong to more than one topic because of polysemy of English language. LDA (Fig. 3) takes original TDM matrix as input and decomposes it into two matrices, one term by topic ($\phi$ matrix) and another topic by document ($\theta$ matrix). The diagonal matrix $\Sigma$ of LSA, is not present in LDA model. Thus this model is more human interpretable. Another difference with LSA model is that here the topic probabilities in matrix are all non-negative numbers and sum up to one [14]. The author of this paper has shown the difference between LSA and LDA matrix decomposition for an example TDM [14].

LDA is named after eighteen century mathematician Gaustav Dirichlet. A Dirichlet process is a probability distribution function over a range which itself is a set of probability distribution. Thus Dirichlet process is a distribution over distribution. LDA assumes a document is a mixture of a few unobserved (latent) topics. In LDA a sparse Dirichlet prior alpha ($\alpha$) is assumed for topic distribution per document. This hyper-parameter controls the smoothness of topic mixture in the document. When $\alpha$ is less than one, it favors topic distribution over a few topics for a document out of all the topics in the corpus. Griffith and Steyvers introduced [Griffiths et al., 2004] another symmetric Dirichlet prior Beta ($\beta$) to control the tightness of a topic. Tightness of a topic means a topic can be described with a few number of words. Experiment shows good choice of $\beta$ value is .01 and it controls that each topic uses few words more frequently out of all the words in the vocabulary of the corpus. In LDA a term can belong to more than one topic with different probability and with different co-occurring terms. LDA like LSA, also relies on bag of word concept where words ordering is ignored.

![LDA Topic Model](image)

**Fig. (3).** LDA topic model.

### 3.5. Information Retrieval

Document similarity in LDA model is measured according to topic distribution similarity between two documents $d1$ and $d2$. In LSA, cosine similarity measure was used for this purpose. However, here in LDA as we are dealing with probability distribution values another standard measure of similarity called Kullback Liebler (KL) Divergence distance is used. If $p$ and $q$ are two probability distribution of topics between two documents $d1$ and $d2$ then KL Divergence $D =: \Sigma_{j=1}^{m} p(j) log_2 \frac{p(j)}{q(j)}$ KL divergence is zero when two probability distributions are exactly same for all topics $j$. As KL divergence is asymmetric, it is also expressed as: $KL(p, q) = \frac{1}{2}[D(p, q) + D(q, p)]$. Similarity of a query to other documents in the corpus can be measured with this KL distance. A second approach is to maximize the probability of the query given the document, i.e. $p(q|d_i)$, where $p(q|d_i) = \Pi_{w_k\epsilon q} p(w_k|d_i) = \Pi_{w_k\epsilon q} \Sigma_{j=1}^{n} p(w_k|z=j)p(z=j|d_i)$. Both LSA and LDA, which factorize a matrix, are computationally expensive when the corpus size is large.
3.6. Word Embedding: Word2Vec (Mikolov et al. 2013)

The above mentioned LSA and LDA algorithms use bag of words model where word ordering in sentences was not considered for semantic preservation. Such models have the weakness of not preserving the context of a word, and perform poorly on similarity analysis. Words that co-occur in the context of each other are similar. A context of a word is surrounding n-gram known as the window size. Smallest token of a text is a word and when a word is represented efficiency considering its context, similarity analysis between words will be more accurate. All other subtasks of NLP like information retrieval, document classification, question answering, named entity recognition, and text summarization etc. will perform better. Advancing state of the art in IR, Thomas Mikolov at Google, proposed a Word2Vec neural network, an unsupervised learning model in 2013 [16, 17] that predicts context words for a target word by scanning the corpus in a fixed window size. He proposed a shallow neural network with a single hidden layer for word embedding that preserves word context, where each word is mapped to a much lower dimensional dense real valued vector space (usually less than 300). When a word is represented as a dense, small size vector (<300) it is called word embedding. Input and output layers of the NN are still sparse one-hot encoding word vector, where every word vector is unique and is of the size of the vocabulary V.

Mikolov’s word embedding has two architectures: i) Skip Gram and ii) CBOW (continuous bag of words) as shown in Fig. (4). The NN is trained by setting up a prediction task. CBOW model predicts a target word when input is a set of context words within a small window size (<10) of the target word. Skip Gram model is the opposite of the CBOW model where surrounding words are predicted from the given middle word.

The NN is trained by scanning through the documents of the corpus one by one and adjusting the NN weights for right prediction. The neural network is trained by back propagating prediction errors [19] and adjusting neural network weights [Rumelhart et al., 1986] to minimize the error. As shown in Fig. (5), W_in and W_out are the learnable parameters of the NN where W_in corresponds to input term embeddings and W_out corresponds to output terms embeddings. After training the model, generally only W_in matrix is used for simple vector algebra for word analogy and similarity computation. A famous example is vector [King] - vector[Man] + vector [Woman] = vector[Queen]. Vector algebra in this lower dimension embeddings is much more efficient and is used to solve many sub tasks of NLP such as finding similar words, word relation, language translation, named entity extraction, sentiment analysis, etc.

Word2Vec is a computationally effective neural network learning model for distributed word embedding that is able to capture more meaningful syntactic and semantic word relationships. The loss function of SkipGram and CBOW model is given in Equation 1 and Equation 2 below, which is to maximize the average log probability of context words given the middle word and vice versa. S is the set of all windows in the corpus and ε is the window size before and after the target term. To some extent, larger window size ε can lead to higher accuracy but takes more training time. The Softmax function used for calculating is costly, so two efficient training algorithms namely hierarchical Softmax and negative sampling are used for the same.

\[
l_{\text{SKIP GRAM}} = \frac{1}{|S|} \sum_{i=1}^{S} \sum_{j<k \in c} \log \left( P(t_{ij} | t_i) \right) \]

\[
l_{\text{CBOW}} = \frac{1}{|S|} \sum_{i=1}^{S} \log \left( P(t_i | \sum_{j<k \in c} t_{ij}) \right) \]

This neural network word embedding outperforms LSA and LDA on various tasks. LSA/LDA model is computationally expensive and cannot be factorized on a very large dataset. For a very large dataset word2vec NN can be trained in parallel or incrementally in less time. Python’s Gensim library is there for word2vec training. The code is as follows:

```python
import Gensim
model = Gensim.models.word2vec(sentences, size=100, window=5, min_count=5, workers=4)
```

For the above code, each word will be represented as 100 dimension vector. Window=5 says the maximum distance between current and predicted word is 5. This model will be parallelly processed by 4 workers and min_count = 5 will ignore the words whose frequency is less than 5 in the corpus.

With this word vector, the sentence vector can be computed by averaging the word vectors that constitute a sentence. After computing the sentence vectors the sentences can be clustered in different groups according to their similarity. For overall summarizing a text, some sentences from each cluster can be picked. For query based answer formulation, the sentences that have more similarity with query sentence will be picked.

3.7. Word Embedding: GloVe Model from Stanford

The drawback of Word2Vec model is that it trains on local statistics of the context window and does not consider the global statistics of word co-occurrence count of the entire corpus. So there is a better algorithm named GloVe which stands for global vector for word representation, as proposed in 2014 by the NLP researchers of Stanford University, USA [18]. Glove is a global log bilinear regression model that combines both local statistics of Word2Vec and global statistics of word co-occurrence matrix. GloVe starts with a term context matrix (TCM), which keeps the count of how frequently each word is seen in a fixed window size context of other terms for the entire corpus. Xij is the count of frequency of context word j for target word i. These counts can be very large number as the whole corpus is considered at once. So the TCM matrix is normalized first. Let Xi = Σk Xki, where k is all neighboring words for word i and then Pij = p(j|i) = Xij / Xi is the probability of j in the context of i. Then it computes ratio of word-word co-occurrence probabilities with respect to probe words as shown in Table 2 to cancel out noise and to establish stronger relationship between relevant words. For learning the word vectors, a function relates the three
word vectors \((i, j, k)\) to co-occurrence ratio of Table 2 as 
\[ F(W_i, W_j, W_k) = \frac{P(k|i)}{P(k|j)}. \]
After simplifying this, Equation 3 is obtained.

Table 2. Co-occurrence Probability Ratio of Words [17].

<table>
<thead>
<tr>
<th>Probability and Ratio</th>
<th>(k = \text{solid})</th>
<th>(k = \text{gas})</th>
<th>(k = \text{water})</th>
<th>(k = \text{fashion})</th>
</tr>
</thead>
<tbody>
<tr>
<td>(P(k</td>
<td>\text{ice}))</td>
<td>(1.9 \times 10^{-4})</td>
<td>(6.6 \times 10^{-5})</td>
<td>(3.0 \times 10^{-3})</td>
</tr>
<tr>
<td>(P(k</td>
<td>\text{steam}))</td>
<td>(2.2 \times 10^{-5})</td>
<td>(7.8 \times 10^{-4})</td>
<td>(2.2 \times 10^{-3})</td>
</tr>
<tr>
<td>(P(k</td>
<td>\text{ice})/P(k</td>
<td>\text{steam}))</td>
<td>(8.9)</td>
<td>(8.5 \times 10^{-2})</td>
</tr>
</tbody>
</table>

Now GloVe constructs an objective loss function \(L\), as least square regression model after adding a weight function \(f(X_{ij})\) as shown below. Here \(V\) is the size of the vocabulary.

\[
L_{GLOVE} = \sum_{i=1}^{V} \sum_{j=1}^{V} f(X_{ij})(W_i^TW_j + b_i + b_j - \log X_{ij})^2 \quad \text{where}
\]

\[
f(X_{ij}) = \begin{cases} \frac{X_{ij}}{x_{\text{max}}} & \text{if } X_{ij} < x_{\text{max}} \\ 1 & \text{otherwise} \end{cases} \quad \text{Equation 4}
\]

The weighting term \(f(X_{ij})\) of objective function makes the computation easy as it only considers the non-zero co-occurring words. In Equation 4, learnable parameters \(W_i\) and \(W_j\) are obtained by Stochastic Gradient Descent (SGD) regression algorithm. It is shown that GloVe outperforms Word2Vec in word analogy, word similarity and named entity recognition tasks even with smaller vector sizes and smaller corpora. GloVe model also trains faster for very large corpora than Word2Vec.

4. AUTOMATIC TEXT SUMMARIZER (ATS)

Automatic Text Summarization (ATS) is the third skill of college information chat-bot. Text summarization is the pro-
cess of shortening text keeping the important content intact. A summary is usually much less than half of the original size. Depending on input text, summarizations are of two types, single document summary and multiple document summaries. Depending on output text, summarizations are of two types, abstractive summary and extractive summary. If important sentences of a text are included in the summary then it is called extractive summary. But human don’t summarize just by picking important sentences only. They rephrase the main idea of the text keeping the intent of the original text the same. This is called abstractive summary. Thus automatic text summarization field is very challenging. State of the art of ATS allows extractive summary only using machine learning techniques. One issue with extractive summary is that it may lack cohesion between the sentences and it will not be easy to make good sense of the summary. Even though abstractive summary is better it needs developing deep learning techniques and yet not very successful in producing good abstractive summary. In this chat-bot, extractive summary is generated. Summarization can also be of two types, generic summary of the whole text or producing answer text to a specific question.

Now let’s look at the unsupervised automatic summarization process that evolved over time [19-26]. Summarization processes first choose the important sentences from a document, then order them as per their relative ordering in the original document. Important sentences of a text can be found in many different ways. Early in 1958, from observation, it was found that the first and last sentences of a paragraph are usually important. This is called positional method (P) in picking important sentences from a text. Also it was found that lengthy sentences are more important. Later on, another method, called Luhn’s method, was used. Here sentence importance was computed by summing the word importance in that sentence. Word importance can be computed by the previously described $tfidf$ score in the document i.e. $\text{weight}(w_i) = \text{tf}_{ij} \times \text{idf}_i$. A smaller set of distinguished words can be chosen by a log likelihood ratio (LLR) or if the word appears in the query below.

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>if $-2\log(\lambda(w_i)) &gt; 10$</th>
</tr>
</thead>
<tbody>
<tr>
<td>weight(wi):</td>
<td>1</td>
<td>if $w_i \epsilon$ query</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>Otherwise</td>
</tr>
</tbody>
</table>

In 1969, Edmundson proposed another simple heuristic method for sentence importance calculation, which not only considered sentence position and word frequency but also considered domain specific cue words such as bonus words and stigma words. Python’s sumy library provides Edmundson-summarizer model which is shown in the following code snippet to produce 5 sentences long summary.

```python
from sumy.parser.plaintext import PlaintextParser from sumy.summarizers.edmundson import EdmundsonSummarizer parser=PlaintextParser.from_string(doc, Tokenizer ("english")) summarizer1=EdmundsonSummarizer(cue_weight=1, key_weight=1, title_weight=0, location_weight=0) summarizer1.bonus_words=('placement', 'salary', 'best', 'internship') summarizer1.stigma_words=('difficult', 'less', 'bad')

Summary = summarizer1(parser.document,5)
```

Now a days, we can use more advanced techniques of LDA, for finding the main topics and topic words of a document. Sentences containing the topic words are more important. Then a clustering algorithm can group the sentences in different clusters as per their similarity. A summarizer will pick up sentences from more important topic cluster in such a way that its importance score is maximum but also at the same time its similarity score is minimum from already picked sentences to avoid redundancy. Word2Vec or Glove method of word representation considers the local and global context of the word and is better in finding sentence similarity and perform better in text summarization.
In Fig. (6a) below, a faculty profile from a college website is first represented as a word cloud representation which shows the important words as per their frequency only. In Fig. (6b), GloVe representation of the words of the same faculty profile document shows the contextual correlation between words. It can be seen that “career”, “academic” and “industry” are close to each other in the vector space as they have been used in the same context.

The automatic text summarization steps are as follows:

Input text → Word representation using GloVe → Sentence similarity Matrix calculation → Sentence rank using Text Rank algorithm → Pick top N sentences for summary

A Text Rank algorithm [27] is a graph based ranking algorithm similar to Google’s Page Rank algorithm used for website importance determination. In TextRank, each vertex of the graph is a sentence and its score depends on how many links through word co-occurrences it has with other vertex and how important the other vertex is. After the ranking of the sentences with TextRank algorithm, top N highest rank sentences are picked for automatic summary generation. Below three sentences, long summary text generated from a faculty profile are shown.

4.1. Evaluating Summaries

Evaluation standard of a summary [28] is called Recall Oriented Understudy for Gisting Evaluation or ROUGE-N. It compares an automatic summary with several other reference summaries generated by human on the basis of n-gram matching between them. ROUGE-N is used for any length of N. ROUGE-2 means n = 2 and it matches bigram between the documents. ROUGE-N formula is given below. As the number S of reference summaries increases, ROUGE-N scores changes.

\[
\text{ROUGE-N} = \frac{\sum_{i=1}^{S} \sum_{x \in \text{baseline sentences}} \min(\text{count}(i,x), \text{count}(i,s))}{\sum_{i=1}^{S} \sum_{x \in \text{reference sentences}} \text{count}(i,x)}
\]

CONCLUSION

The purpose of this paper was to follow the advancement in natural language processing techniques and implement them in a novel application of college information chat-bot design and implementation. The aim of this paper was to demonstrate the effectiveness of the chat-bot which behaves like a human in answering questions more succinctly in a conversational way. It is a great experience to explore so many new technologies of NLP that are required to design a smart chat-bot. This kind of chat-bot can be implemented for various other problem domains like to mimic doctor-patient conversation, student tutoring, online shopping, etc. As relevant information retrieval and automatic text summarization field is a pertinent research area many new algorithms and methodologies are coming up very rapidly. I am going to track this technology development and try to improve our text summarization result with deep learning neural network in future. Also more effective human like abstractive summarizer using deep learning method can be used for this college info chat-bot. In future analytics skill and voice recognized input query skill will be added to this chat-bot.

CONSENT FOR PUBLICATION

Not applicable.

AVAILABILITY OF DATA AND MATERIALS

Not applicable.

FUNDING

None.

CONFLICT OF INTEREST

The authors declare no conflict of interest financial or otherwise.

ACKNOWLEDGEMENTS

Declared none.

REFERENCES

Designing a Chat-Bot for College Information using Information Retrieval


http://dx.doi.org/10.1002/(SICI)1097-4650(199009)41:6<391::AID-ASI1>3.0.CO;2-9


http://dx.doi.org/10.1073/pnas.0307752101 PMID: 14872004


http://dx.doi.org/10.1145/321510.321519


http://dx.doi.org/10.1147/rd.22.0159


http://dx.doi.org/10.1147/rd.22.0159


http://dx.doi.org/10.1038/323533a0


http://dx.doi.org/10.1109/TKDE.2006.130


http://dx.doi.org/10.3115/v1/D14-1162


http://dx.doi.org/10.1109/TKDE.2006.130


http://dx.doi.org/10.3115/v1/D14-1162


http://dx.doi.org/10.1038/323533a0


http://dx.doi.org/10.1147/rd.22.0159


http://dx.doi.org/10.1147/rd.22.0159


http://dx.doi.org/10.1147/rd.22.0159


http://dx.doi.org/10.1147/rd.22.0159