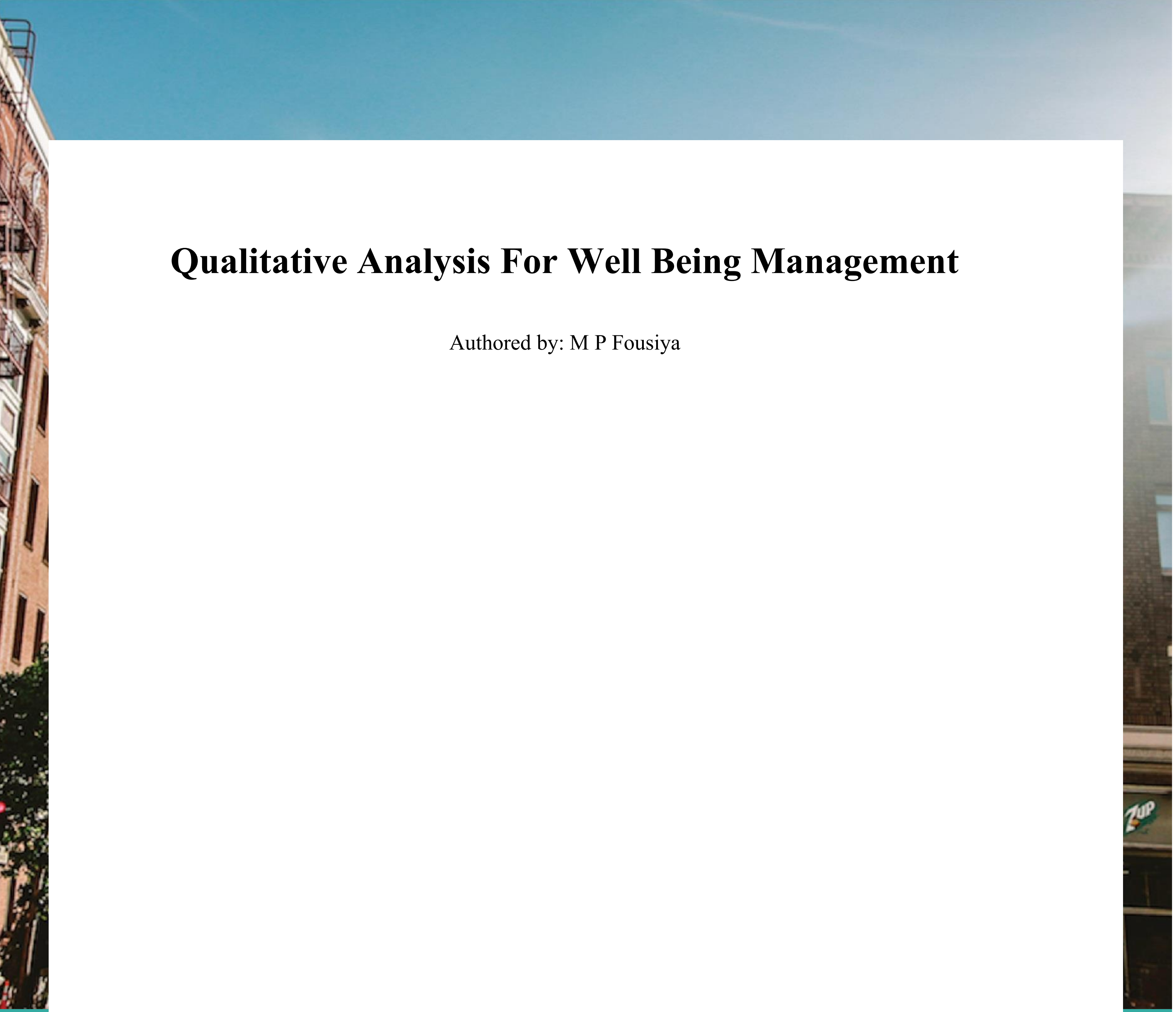


Qualitative Analysis For Well Being Management

Authored by: M P Fousiya

Qualitative Analysis For Well Being Management

ABSTRACT

Addressing the overall well-being of underserved communities has been a missing part of puzzle in Big Data Science. Every individual should have an equal opportunity to reach their full potential in every sphere of their life, but reality is far behind this. Since so many individuals lack the opportunity to improve their lifestyle, the expulsion of health disparity has been emerged as a major worldwide public health objective. This project focuses on the analysis of the reasons for the prevalence of these problems in a rural community in India and thus preventing those at low cost and high speed.

To unveil this, the content of the text data that has been obtained by conducting surveys and informal interviews, are analyzed using two different approaches: (i) Using Word2Vec which helps in understanding the relevant related keywords from the data and (ii) A Text Classification approach using a simple Artificial Neural Network (ANN), whose result is then visualized using a WordCloud. Textual data contains abundant qualitative information that are not easy to undergo a statistical analysis unlike quantitative data. The findings say that, for our data, the combination of Text Classification with Word2Vec provides more efficient results than using those modeling approaches individually, as it can find niche topics and associated vocabularies from the interview data. This project report provides an overview on the qualitative research, the techniques that are used to analyze our textual data for obtaining meaningful information, the limitations of those approaches and suggests some possible ways for further study.

INTRODUCTION

According to the recent report from the World Bank and WHO (WHO, 2017), about half of the world population lacks access to essential health services and 100 million are pushed into extreme poverty because of the health expenses. The report also mentions that even though, in 21st century, there is an increase in the number of people who have access to the health services, the progress is still uneven and the people who are being affected by these are mainly from the rural and underserved communities. Nowadays, the business management for the mass markets like healthcare, pharmaceutical and insurance, are mostly driven and influenced by big data, cloud and mobile technologies. Even though there had occurred many technological advancements in the healthcare sector, in the form of Internet of Things (IoT) and wearable devices, it still remains formidable because of its lack of reach to the masses, especially to the rural and underserved communities which are being characterized by low-income and low-literacy. Technologies would become feasible only when it is designed according to the needs and specifications of the end users within a sustainable social business framework.

The United Nations Sustainable Development Goals (SDG) are a call for action by all countries - poor, rich and middle income - to achieve a better and more sustainable future for all (UnitedNations, 2015). They address the challenges that are being faced by the people from all over the world, including those that are related to inequality, poverty, prosperity, environmental degradation, climate, peace and justice. Their target is to achieve all of their goals by 2030 without leaving anyone behind. One of their goal, ”**Good Health and Well Being**” , is to ensure healthy lives and promote well-being for all in order to achieve sustainable development. This project focuses on employing machine learning to address the problems, in terms of overall well-being, that is being faced by the rural and underserved communities in Ahmedabad, India, and to identify the reasons behind that. This project focuses on employing machine learning to address the problems, in terms of overall well-being, that is being faced by the rural and underserved communities in Ahmedabad, India, and to identify the reasons behind that.

AIMS AND OBJECTIVES

Using machine learning in qualitative research has its own challenges, like inaccurate or inadequate data, data preprocessing and identifying the right technique for dealing with the qualitative data. Considering the above challenges our requirement is to develop an algorithm that can efficiently identify the challenges and the reasons that are affecting their overall well-being, by analyzing the textual interview data. This project explores how different algorithms like Artificial Neural Networks (ANN) and Word2Vec performs in understanding the themes or extracting useful knowledge from an unstructured interview data.

This project compares different machine learning algorithms that takes the unstructured textual data as its input, performs thematic analysis and extracts useful information from those data. The objectives of this project are as follows:

- Translate and transcribe the data into English language, as the interview data is in Gujarati and Hindi language.
- Prepare the training data.
- Prepare the transcribed interview data to make it ready for analysis.
- Performing advanced simulations on the resulting data:
  - Using Word2Vec: Convert each and every word in the data into vector format which will then help to identify the semantic similarities between them.
  - Using a Neural Network for text classification: Classify the responses based on the questions which is already been categorized into different groups and identify the codewords from each group using a Word Cloud.
- Compare the developed models and identify the approach that would help to define our theme “Well Being Management” well.

QUALITATIVE RESEARCH

Research is an organized, systematic and disciplined approach to answer the question about our observations and experiences about the world. It is a structured approach to gather and interpret information that allows us to understand, theorize about or explain experience. Doing only the quantitative research might make the effort of understanding about the overall well-being of the population as void, since it emphasizes only on the objective measurements without having more in depth and systematic examination of the underlying cause. On the other hand, qualitative research focuses on generating meanings and understandings through rich description obtained by diving deeper into the problem (Rose Marie Nieswiadomy, 2012). Most researchers who rely on qualitative approach would agree with the observation that the numbers obtained from quantitative research are impressive, but unfortunately, it conceals more than they reveal (Suter, 2012). Typically, the qualitative approach can address several problems that arises from different philosophical view of the world, by using different methods and design. The design characteristics of this approach is flexible, evolving and emergent in nature. But at the same time, it should be emphasized that the qualitative approach, in no way suggests that it is less disciplined or easier to implement.

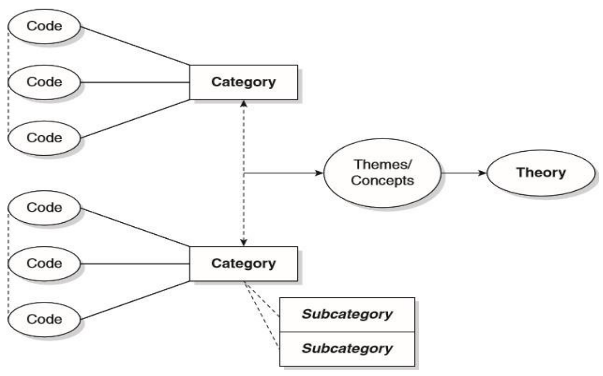


Fig 1: A streamlined codes-to-theory model for qualitative inquiry (Saldana,2015) .

There are different approaches (Grove et al., 2012, Rose Marie Nieswiadomy, 2012) for conducting a qualitative research and the purpose of our project determines which approach do we need to use in our research design. Our survey (interviews, observations, and focus groups) is conducted using a grounded theory qualitative research approach, as it focuses on developing a new theory or expanding an existing theory. It uses constant comparative method to analyze the qualitative data. The survey was conducted with selected stakeholders who are from within or are associated with the target community and without having any prior knowledge about the investigations. The theory, which is grounded in the data, is developed once the data is collected and analyzed. When theory generation is more critical than theory testing, constructs and concepts are grounded in the data and the hypotheses are tested as they arise from the research. The theory which we would implement will be closely associated with a knowledge translation framework, that aims to bridge the significant gap between what is known (evidence) and what is implemented by the stakeholders (practice) (Manns, 2015). The Fig 1 shows the visual representation of how the codes and categories obtained from the provided data can lead to theory formation.

DATASET PREPARATION

Translation And Transcription

Our interview data was comprised of 81 videos in Gujarati and Hindi language whose altogether duration was about 18 hours, which has been then translated and transcribed into English text. The interview was taken from a rural community in Ahmedabad, India, inorder to deeply understand the problems prevailing there. While performing data translation, we need to take care of a variety of factors like presence of multiple languages, presence of alien words, multi-actor scenario, sentence framing, choosing the right word, etc. Also, the translation needs to be done without any additional interpretation and therefore the dialogues should be in its raw form itself. During transcription, we need to keep in mind that in future, to leverage it for analysis, this data is to be converted into a machine-readable format. So, maintaining a standard structure throughout the transcription process will help to ease the data preprocessing step. It should follow a standard structure which includes actor specification, connected words, punctuations and proper nouns in its proper format.

Data Conversion

Training Data: For preparing the training data the categorized questions, that has been prepared for the interview, are labelled with their respective categories prefixed with *"__label__"* and was stored in a tabular format which is then saved into a text file. Table1 shows the category distribution across the training dataset. The table shows that there is class imbalance and also there is only less training data. So, during model selection we need to deal with these issues.

Following shows a part of our prepared training data with labelled questions.

__label__ sm *How many meals would most of the men or women or children or old people would have in a day?*
__label__ sm *What do they generally eat in these meals?*
__label__ sm *Any differences in food for children or the ill persons?*
__label__ sm *Do you think that you eat healthy food?*
__label__ sm *Do you think that others might be considering you as a foody who care for the taste?*
__label__ sm *What is being fat, thin or proper measured body?*
__label__ sm *What activities do every family member do in a day?*
__label__sm *Which members of the family would do physical exercise at least 30 minutes in every day?*
__label__sm *Can you describe the kind of exercise that they would do? For how long and how often?*
__label__sm *How many people in the community are conscious about the need to do exercise?*
__label__ss *What is the average size of a family? Generations or Number of children or age groups?*
__label__ss *Who in a family would be most aware if one has problems?*
__label__ss *Who in a family would be most smart or wise? Who cares the most?*
__label__ss *Who in a family do get cared for the most?*
__label__ss *Do family members tell you to follow their advice? If yes, when and how often?*
__label__ss *How much do they pressurize or force you? And how?...........*

Labels : Number of questions	**Description**
Self Management (sm): 10	Dealt with understanding how the people in the community manage themselves individually.
Social Support (ss) : 11	Tried to understand how the members of a family, different families or the people in a community support and advises each other.
Chronic Diseases (cd) : 16	Investigated about different chronic diseases (mainly focused on the case of diabetes) prevailing in the community and understand the reasons behind that.
Communications (comm) : 5	Tried to understand what main sources are using for communicating with each other or with the outside world and how do they benefit from each. It also included questions about the language that they use in general, also about the social activities and celebrations.
Job and Income (job) : 9	Dealt with understanding about their common jobs, the way they try to get jobs (through employment agencies or NGOs), the services they use (banks, insurance, government schemes) and how they use technology for generating their income.
Poverty (pov) : 10	In general this category of questions tried to investigate about their education, condition of their house, fuel they use for cooking and also about the possessions that they have.
Tools and Design (tool) : 5	Tried to understand about how different technologies like glucose monitoring devices and strips, M-pesa, RFID card, mobile phones, etc, are used currently by the people in the community. Also investigated about what is the probability to use those by them in the future.

Table 1 : Category distribution of training data.

Interview Data: We need to aggregate and convert the whole transcribed data into a tabular format for proceeding with the text preprocessing and analysis. Doing this aggregation manually will consume a huge amount of time, so this is done by using the Python programming language with the help of libraries like pandas, docx2txt, re and os. Extra care need to be taken while converting these data files into a single tabular format, since each questions and responses should be aligned with each other as in the data we already have. Since we had followed a standard structure while performing the transcription itself, it was not that much tough to perform the required conversion. The columns or the features that we need to have in our tabular data are:

- ”AskedBy” : Represents the person who has asked the question. There are 3 interviewers (Interviewer, Interviewer1 and Interviewer2) and we need to specify each of them in their respective scenarios.
- ”Question” : Represents the question asked by the specified interviewer.

- ”AnsweredBy” : As the interview is conducted from different locations in Ahmedabad, also in cases like focus groups there are multiple respondents, there are many values which comes under this category.
- ”Answer” : Represents the answer or response provided by the respondents.

AskedBy	Question	AnsweredBy	Answer
Interviewer	That means there is one anganwadi for every 200 houses?	Lady1	Yes.
Interviewer	That means there are 26000 people here.	Lady1	Yes. That much is for sure, from one corner to another corner there are that much.
Interviewer	That means this is a huge community	Lady1	There is a lot in the interiors. How much ever population is there in the interior, that much is there in the exterior also.
Interviewer	When we come from Gandhi Ashram, we have seen some near the bridge and here.	Lady1	There is still a lot in the interiors. Till Ranip Crossing there is lot of them.

Table 2 : First few rows of the aggregated tabular data

Data Preprocessing

Natural Language refers to the language that is being written and spoken by humans and Natural Language Processing attempts to extract information from these human languages using various algorithms. Most of the computer processing that is being applied to the human languages is just shuffling and skating over these strings or symbols, without understanding what they signify. Therefore, we need to preprocess the data inorder to reduce the huge dimension of the text data and thus makes the process of analysis smoother. Following provides a brief discussion about the steps to be followed for processing our text data.

- **Standardization and Cleaning :** Standardization and cleaning of the input data is extremely important, as it is going to have a huge impact on the quality of the results we obtain. Following steps are performed while standardizing and cleaning our text data:
  - Removing Background Information : As our data contains the background information (like to whom they are speaking), since it may affect our analysis adversely, we need to remove it. During transcription, this background information has been included within brackets. So while cleaning, it can be removed easily by using only a single statement.
  - Dealing with Hyphenated Words : The hyphenated words like ”well-wisher” should be always kept together, even during analysis. So before dealing with other punctuations, we need to deal

with this hyphen at first. Those hyphens should be replaced with an empty string. So that the word ”well-wisher” would be converted as ”wellwisher”.

  – Removing Punctuation and Numbers : Since in text analytics, we are giving more importance to the content of the text than the numbers and punctuation, it is necessary to clean the textual data by removing those punctuation marks and numbers, and this helps to make the data content simpler and easier to read and analyse.

- **Unitize And Tokenize :** The next step of text preprocessing is the identification of the units in text upon which the analysis need to be done. This is also called as unitization. Tokenization is the process of breaking up a piece of text into many pieces such as words, group of words or sentences according to the units that is been identified during unitization (Anandarajan et al., 2018). This is done by separating the text units on the basis of occurrence of spaces or punctuations. As the following example, in our project the tokenization is done on the basis of uni-gram concept where each term or unit have only a single word.

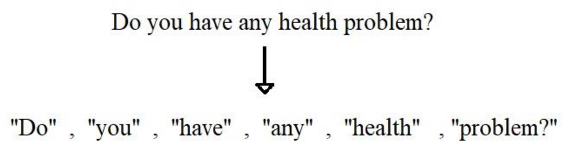

Do you have any health problem?
"Do" , "you" , "have" , "any" , "health" , "problem?"

Fig 2 : Example of Word Tokenization.

Here the sentence ”Do you have any health problem?” has been split into six tokens ”Do”, ”you”, ”have”, ”any”, ”health” and ”problem?”.

- **Parts-Of-Speech (POS) Tagging :** Parts-Of-Speech (POS) Tagging simply means tagging words with their respective Parts-Of-Speech, based on its definition and the context. In the world of NLP most of the models are based on the Bag-Of-Words concept which fails to identify syntactic relations among words (Neale et al., 2018). For example, the Bag-Of-Words model identifies the word ”like” in ”I like you” where ”like” is a verb representing a positive sentiment and ”I am like you” where ”like” is a preposition having a neutral sentiment, as same. POS tagging helps to build parse trees which are helpful in doing Named Entity Recognition and identifying relation between words (Kumavat and Jain, 2015). In our case, POS is assigned based on certain rules (for example, the words ending with ”ing” or ”ed” can be assigned as a verb), and it is called as Rule Based Tagging. Inorder to achieve high precision with this approach we will have to use an exhaustive set of hand coded rules. As these rules are already predefined in Python’s Natural Language Tool Kit (NLTK), POS tagging were done with the help of only a single statement.

- **Named Entity Recognition :** The main idea behind Named Entity Recognition (NER) is to search particularly for the named entities like person name, locations, dates or times in text and classify them to an appropriate predefined category for the purpose of information extraction about those entities (Ratinov and Roth, 2009). Named entities are meant to refer to discrete things in the world which can be referred using a defined name. Some examples of named entities in our data are ”Manav Sadhna”, ”Diwali”, ”Ranip Crossing” etc. If we are using Bag-Of-Words approach without POS tagging, the word ”Manav Sadhna” is taken as two seperate words ”Manav” and ”Sadhna”. For this reason itself POS tagging and NER are really very important in Natural Language Processing. As our project focuses on defining the lifestyle of a particular community, most of the named entities will not be having much importance in this context. Therefore, as part of the text preprocessing stage, the recognised named entities, which has occured less than three times, are removed from our data.

- **Lowercasing and Stop Word Removal:** For machines, the words ”Diabetes” and ”diabetes” are considered as different, since as mentioned before, the system is based on the concept of Bag-of-Words model, where the underlying meaning is not taken into account (Neale et al., 2018). But for humans, both means the same. So we need to standardise the words into a single format (like lowercase), so that the system would understand those words as same.

  Stopwords are the commonly used words (such as “the”, “a”, “an”, “of”, “in”, ”would” etc.) that does not contribute much to the context of the sentences. We do not want these words as part of our data and utilizing our valuable processing time. So as a part of text preprocessing, we need to remove them inorder to limit the number of tokens and thus to improve the efficiency of our analysis.

- **Lemmatization:** Lemmatization is the process of grouping together the inflected forms of a word identified by the word’s lemma (Anandarajan et al., 2018). This process removes inflectional endings of words by returning the dictionary form of the word using vocabulary analysis. Lemmatizing leverages more informed analysis by grouping the words with similar meaning, based on the context around the word, part of speech and other factors. For example, in our data the word ”addictions” and ”addiction”, are lemmatized into its root form ”addiction”. Because of considering the context of words, lemmatizing is typically more accurate than other methods like stemming, but the downside is that it may be computationally more expensive than stemming. Lemmatizing helps us to take the data preprocessing one step further by providing better data for analysis

WORD VECTORIZATION

The qualitative data that has been collected through a survey (interviews, focus groups and group discussion) will help to discover far more than what we intended to find. An interview data is comprised of the dialogues between the interviewer and the respondent, in which the respondent might respond with whatever has come into their mind, even it can be out of the scope. So, to develop a framework that drill deeply down into such a raw data and retrieve useful information is a big challenge. This requires systematic transformation of the unstructured data into a structured format using a method that is able to capture the underlying themes within the data.

When we deal with a text data, we cannot use the text as such as an input to a model, as the computational systems will not be able to understand their meaning or what it signifies. Instead, we need to convert and represent it in a numerical format, so that the algorithm within the model can do the necessary mathematical transformation and obtain the desired result. Nowadays, many researchers who deals with the textual data are using neural word embeddings, which helps to represent each and every word in the vocabulary as a set of numbers (Fernandez-Reyes et al., 2018, Kusner et al., 2015). In this project we are using the word embedding known as Word2Vec, inorder to extract the keywords that helps to understand about the well-being of people in the community.

Word2Vec

Word2Vec is a three layered neural network (LeCun et al., 2015) - one input layer, one hidden layer and one output layer - that helps to process the input text corpus and provides the vector representation of each word in the provided corpus as its output. Therefore, it is also termed as neural word embeddings. Even though Word2Vec is not trained and built using a deep learning neural network, the produced vector format is understandable by any deep learning model.

Each words in a provided text corpora represents a discrete status, and inorder to capture the syntactic and semantic relation between different words, we need to find the probability of co-occurrence of those statuses. This is where the Word2Vec algorithm become really useful, as it tries to understand the probability of co-occurrence of each and every word and this helps to group the words that were having similar meaning or co-occurred together (Mikolov et al., 2013). Based on the appearances of each word in the provided data, Word2Vec is able to find the words that are related with each other word and thus the algorithm generates its training data by itself from the neighboring words.

Training Data Generation: Even though Word2Vec is an unsupervised model, it still need to leverage into a supervised mode and thus need a training data to learn. The training data is produced from within the data without having any auxiliary information like labels. The training data needed to train our neural network is built from the provided text corpora itself, by considering the neighbors of each and every word present in it (Mikolov et al., 2013). Window size determines how many words should be considered as a word's neighbors. When window size equals 1, a word's neighbor would be its immediate adjacent words to its left and right.

While training the model, it captures the semantic feature of every word by considering its context of appearance in the data, thus each word in the vocabulary has been described by another word in the provided data. Word2Vec training is done in one of the two ways (Ganegedara, 2018):

- Continuous Bag of Words (CBOW) : In CBOW, the model learns to predict the target word when the surrounding context is provided. For example in our data if the word ”*sugar ”* and ”*pressure”* are given it would predict the target word ”*diabetes”.*

- Skip-gram : On the other hand, in skip-gram, the target context is predicted from a word. During training if the word is not able to predict its surrounding context properly, the weights at the hidden layer (that means the feature vector of a word) are continuously refined. Thus, the surrounding context acts as the teacher to help a word to adjust its vector values.

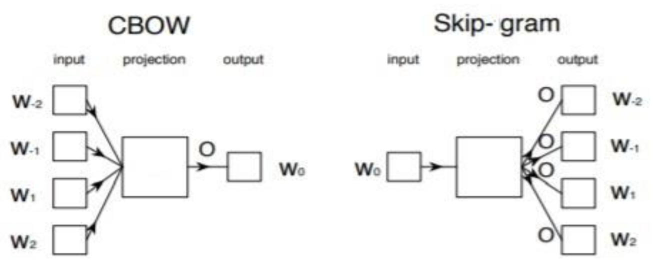


Fig 3: CBOW and Skip-gram model (Ling et al., 2015)

During each epoch of training the distance between more related words becomes closer and closer. Also the words which do not have any relation, move farther and farther. A well trained Word2Vec model places the words that conveys same meaning together, based on the context in which it has appeared in the data. The relation between words are converted into numerical distance, thus the quality is converted to quantity which can be processed by any computational model. But the words are only learned and related based on the vocabulary that has been presented to the model, and not beyond that.

Hyperparameters: Hyperparameters are the attributes that need to be decided by the designer inorder to run the Word2Vec algorithm. These attributes decide how the Word2Vec should perform to achieve the desired result. The important Word2Vec hyperparameters are as follows (Mikolov et al., 2013):

- Dimensions : The dimension of the produced word vector. The number of nodes in the hidden layer will be equal to this dimension.
- Window size: This defines how many words should be considered as a word's neighbor to capture the context. The smaller window size help to understand the syntactic structure. The larger window size helps to understand the semantic nature of the text and this takes more time to train the data.
- Minimum word count: The threshold that determines the minimum occurrence of a word to consider it to convert into a vector. Only the words occurring greater than or equal to the threshold are considered by Word2Vec.
- Downsampling: This determines how the more frequent words like "*of*" are trained.
- Epochs: The number of epochs the network should pass through to obtain the final vector.

Vector Representation: The vector format of each word is unique and it encodes the word's meaning in such a way that we can directly figure out the similarity or the relation between different words mathematically (Mikolov et al., 2013). All these are done without any additional human interference. The output of the algorithm would be a dictionary with all unique words with its vector representation, which can be considered as the meaning of that word understandable by any computational model. In one-hot-encoding (Ganegedara, 2018), as each word can be represented by using only 0s and 1s, the dimension of each word should be equal to the number of unique words in the provided text corpora and also it does not help to capture the semantic similarity between words. On the other hand, in Word2Vec, a word can be represented using a vector with any numbers, also we can decide the dimension of the vector as needed by initializing the number of nodes in the hidden layers. The more the dimension, the result would be more generalized (Ganegedara, 2018). This numerical representation can be fed into any computational model according to our requirement, instead of providing the text in its raw form.

Cosine Similarity: In the case of linear algebra, the similarity between two vectors can be calculated by using the cosine similarity measurement between them. If their cosine similarity is near to 1, it means that the angle between them is near to 0 degree and they are more similar to each other. If a pair of vectors are not at all similar with each other, then their vector representation would be at an angle of 90 degree (Kenter and De Rijke, 2015). If A and B are two vectors, then the formula for finding the cosine similarity between those vectors is as follows (Wei and Wei, 2018).

$$\cos(\theta) = \frac{\boldsymbol{A.B}}{|\boldsymbol{A}||\boldsymbol{B}|}$$

The list of vector representation of words in the vocabulary act as a lookup table to find the most related word. The multidimensional vectors of the words which are more similar to each other would be present nearer to each other in its vector space.

Related Works:

Initially most of the NLP tasks were performed on the bag-of-words concept, which does not help to preserve the semantics involved and only captured the number of occurrences of a word in the text (Lodhi et al., 2002). In 2003, Bengio et al. has proposed the first neural probabilistic model with the objective that inorder to build a proper language model, it needs to understand the sequence of the words in the provided text. This model was called as Feed Forward Neural Network Language Model (FFNNLM) and was capable to learn the distributed representation of words with its corresponding probability in a sequence and has performed well in longer context and obtained good perplexity measure. Even though the model dealt with high dimension, with larger data, the complexity was increasing linearly.

In FFNNLM, the model only captured the sequential words following a particular word inorder to calculate the probability. The model developed by Mikolov et al. [2010] was called as Recurrent Neural Network Language Model (RNNLM) and was able to capture the surrounding context of a word as well. Mikolov et al. [2010] criticized that the FFNNLM model need to specify the context beforehand and since for a word the context window would be different in different scenarios, FFNNLM will not be able to provide accurate results. RNNLM stores temporal information and this helped to improve the perplexity of the model. Even though the model captures the context dynamically, it was not performing well with longer text.

In 2013, Mikolov et al. has built a model that were able to produce more generalized results than the previous models that are based on neural network (Bengio et al., 2003, Mikolov et al., 2010) and criticized that those models consider each word as separate units without taking the similarity between words into consideration. The paper has proposed two variations called as Continuous Bag of Words (CBOW) and Skip-gram model. It is been derived from their earlier work which was based on a simple neural network (Mikolov, 2007, Mikolov et al., 2009).The developed model is called as Word2Vec and it considered the fact that a particular word has multiple degrees of similarity and the representation of a word depends on this. The Word2Vec model was able to capture the similarity between words depending on the context similarity of different words. For example, the words like "*big* " and "*bigger* " were closer in the vector space. Also it shown similarity in words like "*France* " and "*Italy* " , as both represents country names. Word2Vec was working fine even with larger text corpora. But the default model was not capable enough to capture the syntactic details. For example if there is a named entity as "*New Delhi* " , the model considers those as two separate words, "*New* " and "*Delhi* " . And this might cause the model to improper results.

In 2015, Ling et al. has explained some additional methods to be done, before implementing Word2Vec, so that it can capture the syntactic details of the provided text, unlike default Word2Vec. The paper criticized that the default Word2Vec alone is not enough to provide accurate results where covering the syntax details is important. Semantics refers to "*what words occurs together* ", but the syntax refers to "*to where each word goes* " in a sentence.

In 2016, Chiu et al. has used Word2Vec, as it was the method which provided efficient result so far. The paper tried to capture the relation between biomedical terms in a text corpora from PMC and PubMed data and has started with the assumption that larger text corpora will give more generalized result. Chiu et al. [2016] experimented with text corpora of various sizes and hyperparameter settings. The paper also concluded that the larger text corpora does not help to improve the model. More than size of the data, the content (even though the text corpora is smaller) helps to find the relation between important keywords with more accuracy.

In 2016, Ghosh et al. has used the same unstructured text corpora of PMC and PubMed data used by Chiu et al. [2016] to model diseases, agents and symptoms from it. The paper started with the assumption that when Word2Vec is trained with the whole dataset, it would be able to model the diseases and symptoms accurately. The paper has found that even though the vocabulary driven Word2Vec is capable to find the relation between different words in the corpora, for domain specific problems, like healthcare, the general vocabulary is not enough to model the solution. Ghosh et al. at first tried to model diseases with the raw vocabulary of data from PMC and PubMed and found that even though it finds the relation between words, the result is not more domain specific. So, it tried to build a vocabulary by including all the words related with disease, symptoms and agents, so that the algorithm can understand the important keywords to model. This vocabulary can then be used in future in conjunction with the current news data to understand the diseases and symptoms in a current outbreak. Instead of Word2Vec, the developed model was called as Dis2Vec.

As Chiu et al. [2016], Zhu et al. [2017] also tried to build a model that helps to find relation between biomedical terms by using publication data of different sizes. Zhu et al. [2017] found that the result varies as the content and size of the data varies. The model has given poor performance when used only recent available data, as the content is not saturated for producing a complete model. With historical data, even though the model was able to find relation between words, some mostly highly correlated terms were missing. On the other hand, when the model was built using only the abstracts of the documents, the model provided keywords that are highly correlated with a particular word. But still more keywords were missed out compared to when used large historical data.

In 2019, Khatua et al. has used Word2Vec to do a theme wise classification of the unstructured tweets related with Ebola and Zika during outbreaks. The paper has started by saying that the Glove embedding are very rarely used in literatures (Zhu et al., 2017). Motivated by what Chiu et al. [2016] and Ghosh et al. [2016] said about domain specific Word2Vec, the author has tried to classify the tweets with generic pretrained Word2Vec and domain specific Word2Vec. The paper has tried to answer what could be used during initial outbreak when there is scarcity of data and whether domain specific Word2Vec outperforms pretrained Word2Vec and Glove. During scarcity of data the scholarly abstracts in PubMed related with Ebola and Zika has been found useful for doing the text classification and has also found that the domain specific Word2Vec has provided accurate result than Glove and pretrained Word2Vec.

TEXT CLASSIFICATION

Text Classification is a supervised learning problem in which the texts are tagged with some predefined categories on the basis of its textual content (Aggarwal and Zhai, 2012). Extracting useful information from an unstructured data could be a very challenging task, unless it is being organized in a more efficient way. Organizing the whole data manually can be a cumbersome task, if we are having a large amount of unstructured textual data. This chapter provides an overview about the importance of text classification in Natural Language Processing and describes how it is useful for our project.

FastText:

The supervised version of FastText is a text classification model developed by Facebook AI Researchers (Joulin et al., 2016). Nowadays deep learning has been used for building most of the machine learning models, as it is capable of providing more accurate results. On the other hand, these models might show problems when we are having a very large dataset. Also text classification depicts a nonlinear relation between the features of the text and the labels, which can be captured by a nonlinear classifier like a neural network. So instead of using deep learning, FastText employed a shallow neural network that imitates a hierarchical classifier. In this project we are using the FastText model, which is based on CBOW (Mikolov et al., 2013), for classifying the interview data into respective categories.

Training Data And Input : The training data to be passed into FastText should be a text file with the text and its corresponding labels. The label should follow a particular format, so that the model can differentiate it easily. By default, the model expects the labels to be prefixed with *"__label__"*. So the labels are presented to the network as a normal word of the text appended with *"__label__"*. The whole input data for which the label is to be predicted should be provided as a list of text, so that each list item can be assigned with a label depending on its cosine similarity with the context.

Average of NGrams : In FastText a text is considered as combination of NGrams instead of combination of words. For example if NGram=2, then considering a word ”*gujarati*”, its corresponding NGram representation would be as

”gu”,”uj”,”ja”,”ar”,”ra”,”at”,”ti”

The vector representation of a single word would be the average of the vector of the constituting ngrams. Similarly the vector representation of a sentence is also taken as the average of the vector of the constituting ngrams (Joulin et al., 2016)

Architecture : The supervised FastText classifier has the following architecture:

- Input Layer: The vector representation of the ngrams constituting the text are given as input. So, the number of input nodes will be equal to the number of ngrams in a data record.
- Hidden Layer: The embedded input features are averaged to form the hidden layer. So that the number of nodes in the hidden layer would be equal to the embedding dimension of an ngram.
- Output layer: The output layer implements a hierarchical softmax architecture, which is a binary tree having leaf node for every word in the vocabulary.

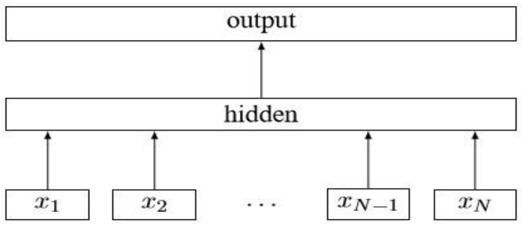


Fig 4 : FastText Model Architecture (Mikolov et al., 2013)

Hierarchical Softmax : In case of FastText classification, we need to know only about the probability of the words prefixed with *”__label__”* in a given context. So searching the entire list of words for the labels would be cumbersome if we are having a very larger vocabulary of words. Inorder to fix this, FastText uses a hierarchical softmax architecture at the output layer.

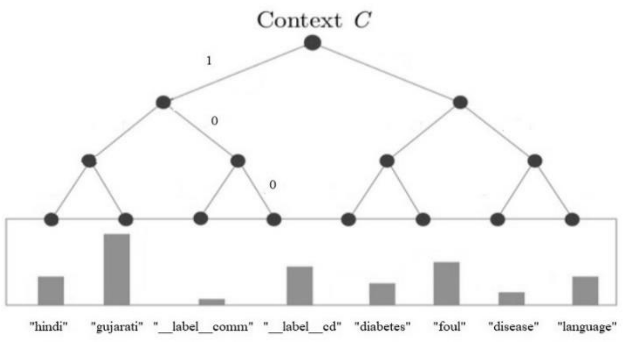


Fig 5: Example of Hierarchical Softmax Architecture

Inspired by the binary tree created by Morin and Bengio [2005], FastText used a hierarchical softmax architecture at the output layer, in which each words in the vocabulary is associated with a leaf node of a binary tree. Let us deeply understand this architecture with the help of an example. If the unique words in the data are *[”hindi”, ”gujarati”, ”language”, ”foul”, ”__label__comm”, ”diabetes”, ”__label__cd”, ”disease”].* As there are eight words in total, it can be represented using a binary tree having eight (ie 2^3) leaf nodes with each leaf node having a unique path. Therefore the depth of the balanced binary tree would be 3 (ie $\log_2 8$). So if there are N unique words in the vocabulary, then we need a binary tree of depth $\log_2(N)$. The structure of the tree is always stable, thus the path from root to a particular word is fixed.

But how each words are assigned to the leaf nodes?

As there are only eight words, each word can be represented using eight different representation using 1s and 0s. So for forming eight (ie 2^3) different representation, we need only 3 bits like 000, 001, 010, 011, 100, etc. Arrangement of the words in the binary tree depends on this binary representation. 1 denotes movement from a node to its left node and 0 denotes movement from a node to its right node. So on the example tree, the word ”__label__cd” is at 100 and the word ”__label__comm” is at 101. Thus by looking at the binary representation of the labels, we can find its corresponding probability without doing an exhaustive search of the entire list. Thus the complexity of finding the probability of a particular label has been reduced from O(n)
to O($\log_2 n$)

How probability of labels are calculated in hierarchical softmax?

- In hierarchical softmax, each nodes other than the leaf nodes are represented as vector v'_n

.

- We know that the sum of probability of a branching decision from a particular node in a binary tree is 1. That is if the probability for going right is p , then for going left is 1-p. Thus the sum of probability to reach the leaf nodes will be 1.
- When using the traditional softmax, the probability of a label or word is obtained by finding the dot product of hidden layer vector h with the respective word vector v'_w and then applying softmax transformation on it (Mahmoud, 2018). Instead in hierarchical softmax, the probability of a word is obtained by multiplying the probabilities at each internal node on the path to the respective word.
- In the above example, the path to reach the word ”__label__cd” is 100. If the nodes traversed to reach ”__label__cd” are v_1, v_2, v_3 and corresponding vector representation are v'_1, v'_2, v'_3, then the probability of the word ”__label__cd” is as following:

$$p(\text{"\_\_label\_\_cd"} \mid context) = (1-\sigma(b_1+v'_1.h(x)))*\sigma(b_2+v'_2.h(x))*\sigma(b_3+v'_3.h(x))$$

Therefore, in general, the formula for finding the probability of a word can be written as follows. If the leaf node is at depth d+1 with parents $n_1, n_2, ..., n_d$, then its probability is (Joulin et al., 2016):

$$p(n_{d+1}) = \Pi^{d}_{i=1} p(n_i)$$

Hyperparameters: The important hyperparameters of FastText algorithm are as follows:

- wordNgrams: This denotes the number of ngrams to be considered that constitutes a text.
- Learning rate: Learning rate denotes the speed that the algorithm takes to learn the relation between the context and the labels.
- Dimension: The word embedding dimension.
- Epochs: The number of epochs to train the network.
- Loss function: The activation function to be used while training the algorithm. It can be negative sampling, hierarchical softmax or softmax.

Related Works:

As SVM (Fan et al., 2008, Joachims, 1998) and ANN is capable to capture the non-linear relation between the data and their label, Moraes et al. [2013] has compared the performance of both using text classification. From the results, it has been found that ANN outperformed SVM. Even though ANN has provided good results, it was not stable all the time as SVM, also the training time for ANN was very much higher than that of SVM. In the presence of noisy data, ANN performance was also affected, unlike SVM.

In 2015, Zhang et al. has used Convolutional Neural Network (CNN) for text classification with ConvNets at character level and has compared different approaches of word representation like Bag-Of-Words with TF-IDF, Bag-Of-NGrams with TFIDF and average of word embeddings. With CNN, Bag-Of-NGrams with TF-IDF has given good accuracy results. Usage of character level ConvNets has helped to give good result even with less curated user generated data. Even though TF-IDF with Bag-Of-NGrams has given good performance, it was not suitable for larger text corpora that ranges to millions of records. Also, when the text was represented as the average of word embeddings, CNN has given the worst performance.

In 2016, Joulin et al. has started by saying that even though deep neural network gives good performance, it is very slower compared to linear models when dealing with larger data. The paper attempted to build a text classification model by using a simple neural network with a hierarchical softmax layer at the output layer. This proposed model by Facebook AI Researchers has outperformed the character CNN (Zhang et al., 2015). This model is called as FastText as it performs the text classification with high accuracy and within lesser time. As the model represents a word as average of embedding of ngrams, it is capable to handle out of vocabulary words as well, unlike the default Word2Vec. During text classification, the hierarchical softmax architecture helps to train data which is more than one billion words in less than 10 minutes.

Armand et al. [2016] has compressed the developed FastText model (Joulin et al., 2016), and has suggested the way to optimize it in terms of the memory consumed while running the model without sacrificing its efficiency and speed. Methods like product quantization (Jegou et al., 2010), feature reduction by pruning the vocabulary, using hashing trick and bloom filter (Agarwal et al., 2014) are implemented to optimize the model. The refined model has been compared with the default FastText and CNN results of text classification and has found that the proposed model was performing better.

In 2017, Bojanowski et al. has tried to showcase the importance of considering the sub word information to form a word or a sentence. The model was developed using a Recurrent Neural Network (RNN) with 650 Long Short Term Memory (LSTM) units with drop out and weight decay. The model which were using the sub word information has performed well than the model which were using the words as such. The proposed model which used ngrams were providing good results for ngrams between 3 to 6 and is able to represent the out of vocabulary words as well. Also, it has been found that less ngrams (like ngrams=2) is not sufficient enough to capture the sub word information.

In 2017, Mikolov et al. has proved that even though the FastText model is performing well in text classification, the model might underperform in presence of data redundancy and this could be dealt with deduplicating the data. Mikolov et al. [2017] has found that data deduplication has helped to capture the relation between different keywords more efficiently and thus produced more accurate results.

METHODOLOGY

Word2Vec

All the tasks like dataset preparation and preprocessing, Word2Vec implementation and visualizing the results are done with the help of Python programming language. Word2Vec is implemented using Gensim's Word2Vec model, which can be imported into the Python environment.

Data preprocessing: After preparing and preprocessing the raw transcribed interview data, by passing through each of the steps mentioned in the Dataset Preparation section, now we have the data with only relevant features. The data does not have any stopwords, numbers and punctuations, the named entities are recognized and attached together, the background information are removed, and the words are lemmatized. Following steps are done to further prepare the data to pass it through Word2Vec:

- Each of the preprocessed question and response are combined. We need both the question and the response, as both contains relevant information.
- The combined text has been converted into a list of words. The resulting data format to be passed into the model is a list of lists of texts as follows:
  [['anganwadis', 'one', 'corner', 'another', 'corner'], ['mean', 'one', 'anganwadi', 'every', 'house'], ['mean', 'people', 'sure', 'one', 'corner', 'another', 'corner'], ['mean', 'huge', 'community', 'lot', 'interior', 'ever', 'population', 'interior', 'exterior'], ...]

Building Word2Vec: A Word2Vec model which could take the one-hot-encoding of a unique word in the vocabulary as input and give its vector format as its output is designed with the following hyperparameters:

- Dimensions : 300
- Downsampling: 1e-3
- Minimum word count : 3
- Window size : 7
- Epochs : 20

At first, the model tries to figure out the unique words in the vocabulary and find one-hot-encoding (Ganegedara, 2018) of each, which is then passed as input to find the respective embedded vector. Each one-hot-encoded representation is trained against the one-hot-encoded representation of its 7 neighboring words, as the window size is 7. During each epoch, the most related words comes nearer to each other in the 300-dimensional vector space. Following python statement helps to do the model training where response2vec is an instance of Word2Vec class.

response2vec.train(dataToVectorize,total examples=response2vec.corpus count,epochs=20)

After training, each word will have its 300 dimensional vector format, which can be then saved into a file. We can retrieve and use it later as a pretrained model without the need of training it again on the same data. Following statement help to save the trained model in a file response2vec300D20.w2v in trained300D20 folder.

response2vec.save(os.path.join(”trained300D20”, ”response2vec300D20.w2v”))

Already saved pretrained model can be loaded using the following statement:

w2v.Word2Vec.load(os.path.join(”trained300D20”,”response2vec300D20.w2v”))

Visualizing Vectors using t-SNE: T-distributed Stochastic Neighbor Embedding (t-SNE) (Maaten and Hinton, 2008), is a well known technique that helps to visualize the high dimensional dataset by converting it into low dimension.

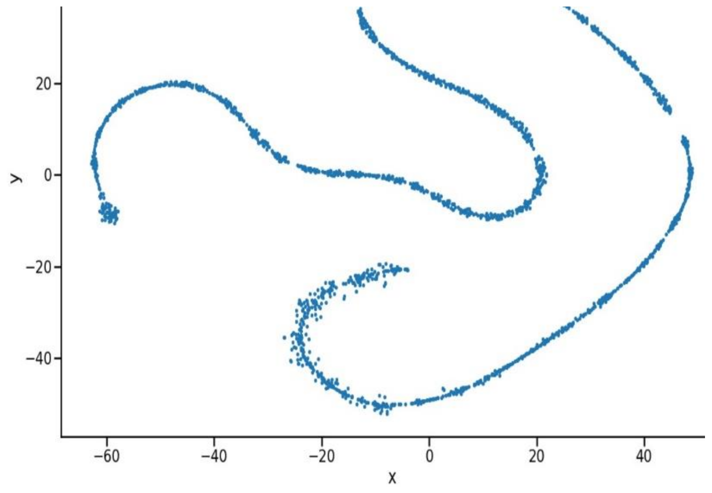


Fig 6: Compressed 300-D vector in 2-D vector space after 1 epoch of training

Our vector representation of each word is of 300 dimension, which is difficult to visualize on a two-dimensional screen. Inorder to visualize it, at first we need to compress our 300D data into 2D format. t-SNE helps to perform this reduction task by preserving the original clustering in the high dimensional space, which is 300D in our case. Fig 6. is the t-SNE plot obtained by training the Word2Vec for only 1 epoch. We can see that all points are clustered near to each other, without any distinction. These points move apart as the number of epochs increases and finally the most related words would be nearer to each other and will have high cosine similarity with each other. Also the words which have no relation would be very far from each other. Following is the t-SNE plot of the vectors obtained after training model for about 20 epochs.

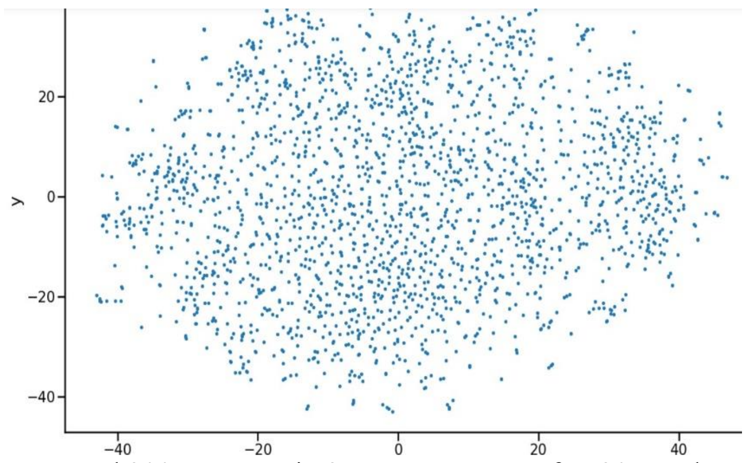


Fig 7: Compressed 300-D vector in 2-D vector space after 20 epochs of training

For more clarity, let us now try to zoom into each part of the plot. From the following plot we can see that the words like laziness, suryanamaskar, yoga, paranayam, etc, which are related with the exercises has been clustered together.

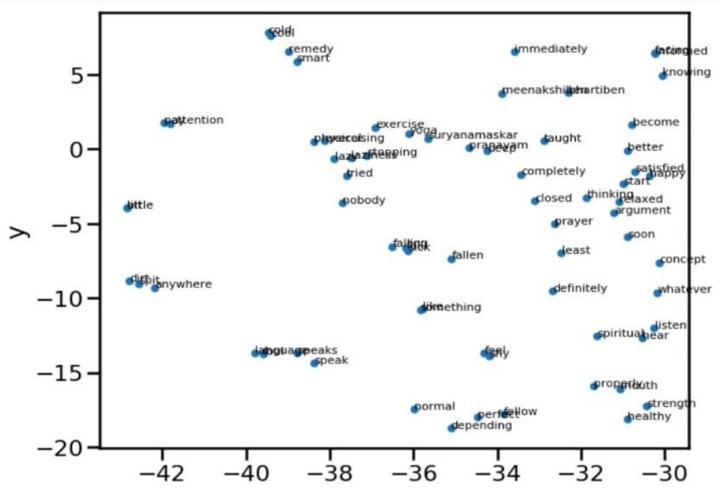


Fig 8: Zoomed visualization of t-SNE plot

Finding Related Words: The vector form obtained can be used for different purposes like distance calculation, similarity measure between different words or for ranking words in a document. Our project uses Word2Vec for distance calculation using cosine similarity measure to find the most related keywords. Word2Vec class is capable to provide the most related 10 words to a particular word with a single statement. For example the following statement tries to obtain the related words with the word ”*language*” in the provided data.

response2vec.most similar(’language’)

The trained model will have the 300-dimensional vector representation of all the unique words in the vocabulary. In a skip gram model, when we want to find the most related words or context with a particular word, the vector representation of that particular word is given as the input to the model and the list of vector format of all other words acts as a look up table, from which the cosine similarity between each vector and vector of the respective word can be found. The words having higher similarity will be provided as the output context of that particular word.

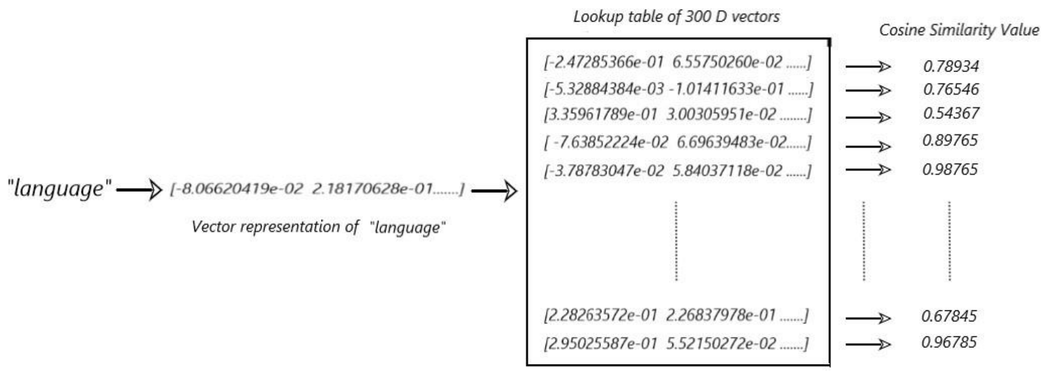
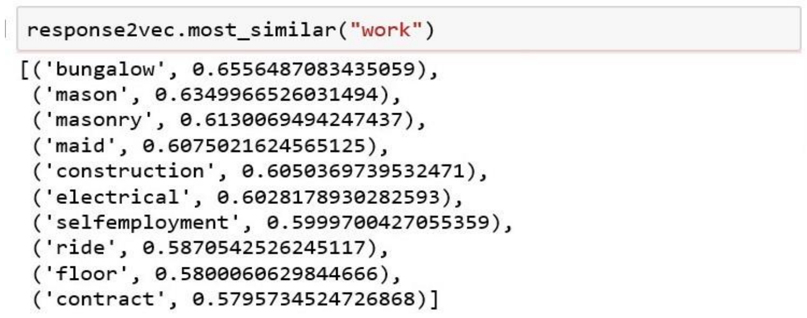

Lookup table of 300 D vectors
Cosine Similarity Value
[-2.47285366e-01 6.55750260e-02 ......]
[-5.32884384e-03 -1.01411633e-01 ......]
[3.35961789e-01 3.00305951e-02 ....]
[-7.63852224e-02 6.69639483e-02......]
[-3.78783047e-02 5.84037118e-02 ......]
[2.28263572e-01 2.26837978e-01 ......]
[2.95025587e-01 5.52150272e-02 ......]
0.78934
0.76546
0.54367
0.89765
0.98765
0.67845
0.96785
"language"
[-8.06620419e-02 2.18170628e-01.......]
Vector representation of "language"

Fig 9: Representation of finding most related word with ”*language*” .

Result: Our project needs to find the codes and subcodes related with **Well Being Management** of the people in rural communities in Ahmedabad. As a human, we know certain keywords like *language, work, disease, food, fuel, devices at home, etc,* that can define the lifestyle of the people in a community. Let us try to find whether Word2Vec is capable to define about these keywords. Following are the results obtained when Word2Vec tried to drill deep into the keyword “*work*”:

response2vec.most similar(”work”)



```
response2vec.most_similar("work")
```

```
[('bungalow', 0.6556487083435059),
 ('mason', 0.6349966526031494),
 ('masonry', 0.6130069494247437),
 ('maid', 0.6075021624565125),
 ('construction', 0.6050369739532471),
 ('electrical', 0.6028178930282593),
 ('selfemployment', 0.5999700427055359),
 ('ride', 0.5870542526245117),
 ('floor', 0.5800060629844666),
 ('contract', 0.5795734524726868)]
```

Evaluation: Following are some dialogues about their work that tells whether the result provided above is conveying the right information or not.

*”She goes for work in **bungalow**.”,”Mostly ladies go as **house maids** and they also go to pick the papers from the garbage.”,”**Masonry** work, they go for collecting recyclable materials from garbage, pedaling rickshaws.”,”Power house, from where the **electricity** and light comes. In that also they have work.”*

Even though the information captured by Word2Vec about their work is right, some of the information like ”*collecting recyclable materials from garbage”* has been missed out.

By using Word2Vec, we were able to get a lot more by representing a word by means of its context or neighbors. Deeply observe for a moment, that the Word2vec algorithm has never been taught a single rule of English syntax and semantics. Still it learned more in a flexible manner without any human intervention. Word2Vec has started its job on the interview data with a blank slate, as it knows nothing about the world. And as soon as the training completes, it was able to find relations between different keywords.

As concluded by Chiu et al. [2016], Ghosh et al. [2016] and Khatua et al. [2019], in our data, this method can be used if we have the domain specific keywords or vocabulary that helps to define the well-being of a community. Also, these keywords need to be present in our preprocessed data. Otherwise, either we would end up in a blank state without having the desired result or the model might show the words that are not important to the context as the most related keyword with high probability.

FastText

Exploratory Data Analysis: It is important to do an Exploratory Data Analysis (EDA), inorder to deeply understand the training data. The main issue that could be faced by a text classification problem is imbalanced classes in the data. For example, if the training data contains more data (say 95%) belonging to a particular class, then the model would produce a biased result towards that class. And it will not produce a generalized output for the provided new data. Undersampling of majority class or oversampling of minority class needed to be done as required (Han et al., 2005).

During EDA, it has been found that in our training data there is imbalanced classes. As FastText hierarchical softmax architecture deals with this class imbalance by itself, by having more deeper branch for more frequent classes, there is no need to oversample or undersample to make the classes balanced**.**

Data Preprocessing: Both the list of questions and interview data are preprocessed and randomized inorder to avoid bias. The preprocessed training data is stored as a text file dataForClassifn.txt. The resulting preprocessed training data with the label will be as follows:

__label__pov *many household member year old younger*

__label__ss *wellwishers find*

***__label__job** service important bank account accident insurance health insurance skill development training access government scheme*
***__label__job** describe look job opportunity get help peer friend relative try employment agency ngo trust agency*
***__label__cd** think beneficial physical exercise reason*
***__label__job** long using phone popular phone*
***__label__ comm** people communicate family friend worry kind worry talk.........*

Building FastText: The FastText model which takes the preprocessed data as its input and provides the probability measure of each of the labels to be assigned as output is designed with the following hyperparameters. After optimizing the model with different hyperparameters, following had given a more convincing categorization during FastText classification.

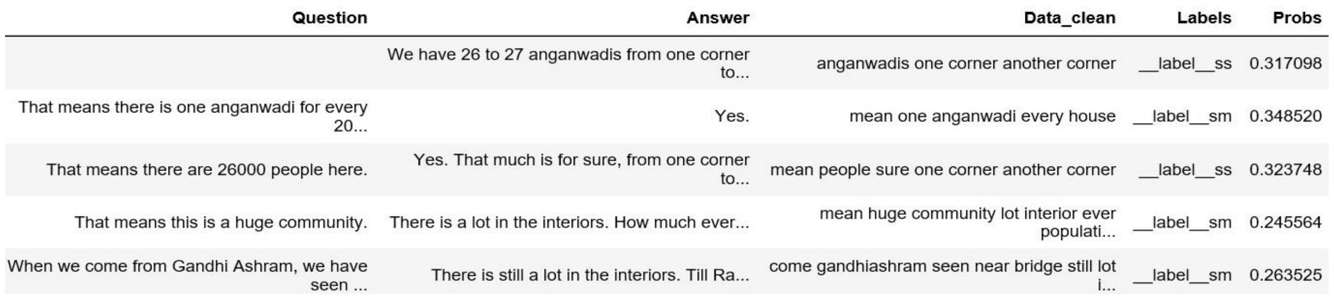
- Dimension: 20 :- As our training data have only 66 questions, after preprocessing there were only 312 unique words. As we have less number of unique words, the dimension of a vector is initialised as 20.
- Epochs: 500
- wordNgrams: 2 :- The experiment done by Bojanowski et al. [2017] has concluded that the ngrams between 3 to 6 were able to capture the semantics and give more generalised result. But with our interview data, as we have very less training data, the FastText model with wordNgrams=3 were not giving good classification result and therefore took wordNgrams=2.
- Learning rate: 0.01 • Loss function: hs

The model is trained on each unique words in the vocabulary, including the label. A 10 dimensional word embedding for each of the unique words are formed based on their context. Following python statement is executed to train the FastText model.

model = fasttext.train supervised(”dataForClassiffn.txt”, lr=0.01, dim=20, epoch=500, word ngrams=2, loss=’hs’)

After training, the labels are predicted for the preprocessed interview data with the following python statement, where testdata[’Data clean].tolist() is the list format of the preprocessed data.

labels = model.predict(testdata[’Data clean’].tolist(),k=1)]

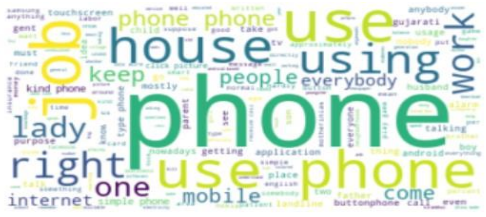


Question	Answer	Data_clean	Labels	Probs
	We have 26 to 27 anganwadis from one corner to...	anganwadis one corner another corner	__label__ss	0.317098
That means there is one anganwadi for every 20...	Yes.	mean one anganwadi every house	__label__sm	0.348520
That means there are 26000 people here.	Yes. That much is for sure, from one corner to...	mean people sure one corner another corner	__label__ss	0.323748
That means this is a huge community.	There is a lot in the interiors. How much ever...	mean huge community lot interior ever populati...	__label__sm	0.245564
When we come from Gandhi Ashram, we have seen ...	There is still a lot in the interiors. Till Ra...	come gandhiashram seen near bridge still lot i...	__label__sm	0.263525

Fig 10 : Classified data using FastText

Now the labels variable will have the label and its corresponding probability to be assigned as the respective text's label. As there are so many responses in the data, that does not belong to any of the seven categories, their assigned label probability is very less compared to the one that surely belongs to one of the category. The probability of the predicted labels has ranged from 0.1770 to 0.9327.

Result Visualization using WordCloud : WordCloud is a visualization technique that can be used to visualize the most prominent keywords in a piece of text corpora. The text or the keywords that occurred the most will be shown in a larger size. Like that each and every word in the corpora will be shown in different sizes depending on its number of occurrences. For example, following is the WordCloud visualization of the codes from the category "*Job and Income*" classified by FastText. The WordCloud visualization shows that job and income data talks more about their work, type of phones that is being used (touch screen or button phones), etc.



Evaluation : FastText with WordCloud visualization helped to find the most prominent keywords associated with different categories. But still it is not complete in the sense that it does not help to understand the details in deeper. For example, in the WordCloud visualization of data categorised as Job and Income, the word "*phone*" is shown as the most prominent one. Also we are not able to see much details about what kind of jobs are they doing in the respective WordCloud visualization.

Building an Analysis Engine by Combining FastText and Word2Vec

As specified, neither Word2Vec nor FastText alone were able to capture the underlying theme of our interview data properly. Both the results were not capable to understand about the lifestyle of the people in the community properly. Therefore, introduces an analysis engine that tries to drill deeper into the data by feeding the results of FastText into Word2Vec and provide the results in the form of graph visualization.

Extracting keywords from FastText: Perform text classification as before using FastText. This helps to obtain the prominent keywords from each category. Most of the obtained keywords contains many non-prominent words as well which have number of occurrences as 1, 2 or 3. Therefore from the keywords that were visualized using WordCloud, only those keywords that had occurred more than 5 times from the respective categories are extracted and converted into a list format. Following is an example list of codes from the category of "*Job and Income*".

['people', 'use', 'phone', 'house', 'kind', 'simple', 'touchscreen', 'boy', 'right', 'even', 'keep', 'type', 'must', 'mobile', 'landline', 'box', 'lady', 'using', 'usage', 'game', 'picture', 'getting', 'job', 'know', 'one', 'child', 'talk', 'work', 'son', 'husband', 'like', 'nobody', 'anybody', 'call', 'tv', 'everybody', 'android', 'father', 'get', 'internet', 'two', 'take']

Retrieve trained Word2Vec: The Word2Vec model that has been trained on our preprocessed interview data were saved before, in a file "*response2vec300D20.w2v*". While building the analysis engine, there is no need to retrain our data to the Word2Vec algorithm. Instead, we can retrieve the already saved model for our task. Following is the Python statement used to load an already saved Word2Vec model *response2vec300D20.w2v*.

response2vec = w2v.Word2Vec.load("response2vec300D20.w2v")

Deep Drilling: For each prominent code obtained from each of the categories, let us now try to obtain the relevant subcodes by drilling one level deeper into the data. As the data used in FastText and Word2Vec has been passed through the same preprocessing steps, the obtained codes from FastText classification will have its corresponding vector representation in Word2Vec. Now by giving these codes as an input to the trained Word2Vec model helps to find the related subcodes. Thus, we can obtain the result, with code and subcode relation, that will help for theory formation from the unstructured data (Refer Fig 1).

Visualization Using Graphviz : Graphviz is a library package that can be used in Python, to create a graph object with nodes and edges for visualizing the relation between different nodes. In our project, the graph has been built with the root node as one of the seven categories and the subsequent nodes representing the codes and the subcodes. For example with the list of codes that has been obtained from the category "*Job And Income*", if we try to drill deeper into "phone" or "*work*" we will obtain the following graphs.

Graph Example 1: Job and Income − > phone

Assumption from the Fig 11: "*The people in the community uses phones with touchscreen and also button phones. It includes Nokia, Samsung and china made phones. They use Facebook from phones.*"

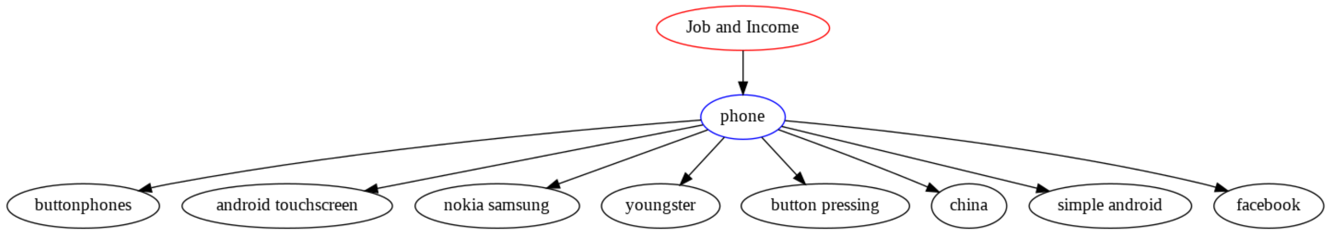


Fig 11: Graph Visualization of Job and Income − > Phone.

Actual Responses from the Data: “*Mostly all* ***youngsters*** *have phone with them and there will be one more phone with one of the other family members.”,”Company hardly any, but* ***Chinese made phones****.”, ”Yes,* ***Nokia and Samsung****, but they are very less. We see mostly china phones.”,”Yes.* ***Button pressing phones****. Some people have those* ***china made touch-screen*** *phones.”,”They use phones for downloading movies and songs or like you said they use* ***Facebook****, WhatsApp also”*

Graph Example 2: Job and Income − > work

Assumption from the Fig 12: “*The people in the community do bungalow floor cleaning, masonry work, maid job, construction work, ride pedal rickshaw and contract work for their earning.”*

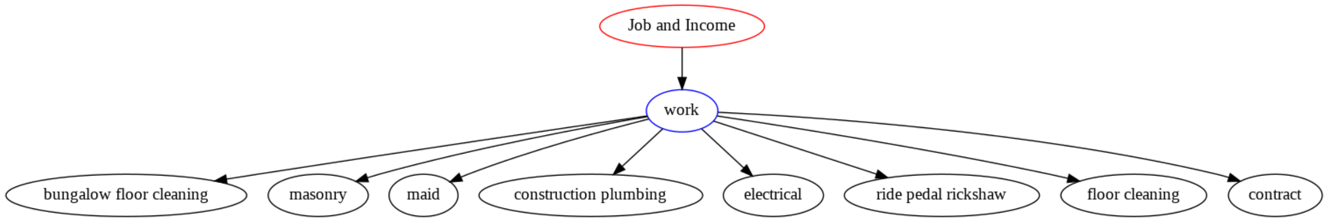


Fig 12: Graph Visualization of Job and Income − > Work.

Actual Responses from the Data: “*They drive those* ***pedal rickshaw****”,”In municipality,* ***they clean the floor*** *or sit in office”,”In the morning ladies finish their house hold work and go for jobs like* ***cleaning floor at bungalows*** *or for collecting recyclable materials from garbage”,”The majority of people in that area do the* ***plumbing and construction*** *work only”*

Evaluation: The model built by pipelining the results of FastText into trained Word2Vec is able to give more meaningful output than when used those models individually. When the codes obtained from FastText (by including the questions and the answers) has been given as input to the Word2Vec, the problem of using the wrong word as input has been solved by this model. Also, the model has provided the subcodes based on the respective categorized data as the result.

By going deeper into the codes obtained from different predicted classes, an important point to note is that, even though most of the results confirms with the actual data, some of the results are not meaningful enough to reach into a conclusion. This is because of the reason that the interview data is not curated and even after preprocessing it contains many words that are not domain specific. The model has vectorized all the words - either it can be domain specific that helps to define the lifestyle of the community or not. Creating a domain specific vocabulary as suggested by Ghosh et al. [2016] can be a solution to this problem.

CONCLUSION

The project has tried to address the overall well-being of underserved communities in Ahmedabad, using a Data Science approach on the unstructured interview data. This is the first time that a machine learning approach has been used to find themes from an interview data. The process involved data translation with transcription and then processing them using Word2Vec and FastText.

When the model was built using the Word2Vec or FastText alone, it was not capable enough to understand the data well and thus provided results which were not meaningful. Therefore, tried to build an analysis engine by pipelining the results of the FastText and Word2Vec models. This has helped to define the lifestyle of the community more deeper than before. We had successfully managed to design a generalized framework of the model which takes the categorized questions and the interview data in tabular format as input and provides the codes and subcodes as graph visualization images as output. These image files are automatically stored in an organized format in a folder VisualGraphs. The produced results has been compared with the actual data from the interview.

The results obtained from the model helped to identify most of the themes from the interview data within minutes, which would take hours when manually tried to investigate through the whole interview data. These results can be used to understand the problems in the community and the causes for it, and thus take necessary actions according to that.

Future Work: As we saw, even though most of the important keywords related with the lifestyle of the underserved community in Ahmedabad has been found using the model, still most of the questions has not been answered properly by it. This is due to the fact that the human responses are not curated, and it results in the introduction of unnecessary words into the model.

Building a domain specific vocabulary would help to deal with this problem to a great extent. But creating a vocabulary for the domain "***Understanding the Well-being***" could be a very time-consuming task, as people in different area will have different kind of lifestyle. With the domain specific vocabulary, the model would only have to vectorize those specific words and thus the model would be free from the intruding unnecessary words.

REFERENCES

Agarwal, A., Chapelle, O., Dud´ık, M., and Langford, J. (2014). A reliable effective terascale linear learning system. The Journal of Machine Learning Research, 15(1):1111– 1133

Aggarwal, C. C. and Zhai, C. (2012). A survey of text classification algorithms. In Mining text data , pages 163–222. Springer.

Anandarajan, M., Hill, C., and Nolan, T. (2018). Practical Text Analytics: Maximizing the Value of Text Data, volume 2. Springer

Armand, Grave, E., Bojanowski, P., Douze, M., J´egou, H., Mikolov, T., and Joulin (2016). Fasttext. zip: Compressing text classification models. arXiv preprint arXiv:1612.03651.

Bengio, Y., Ducharme, R., Vincent, P., and Jauvin, C. (2003). A neural probabilistic language model. Journal of machine learning research , 3(Feb):1137–1155

Bojanowski, P., Grave, E., Joulin, A., and Mikolov, T. (2017). Enriching word vectors with subword information. Transactions of the Association for Computational Linguistics, 5:135–146

Chiu, B., Crichton, G., Korhonen, A., and Pyysalo, S. (2016). How to train good word embeddings for biomedical nlp. In Proceedings of the 15th workshop on biomedical natural language processing, pages 166–174

Fan, R.-E., Chang, K.-W., Hsieh, C.-J., Wang, X.-R., and Lin, C.-J. (2008). Liblinear: A library for large linear classification. Journal of machine learning research, 9(Aug):1871–1874.

Fernandez-Reyes, F. C., Hermosillo-Valadez, J., and Montes-y Gomez, M. (2018). A prospect-guided global query expansion strategy using word embeddings. Information Processing & Management, 54(1):1–13

Ganegedara, T. (2018). Natural Language Processing with TensorFlow: Teach language to machines using Python's deep learning library . Packt Publishing Ltd.

Goodman, J. T. (2001). A bit of progress in language modeling. Computer Speech & Language , 15(4):403–434

Ghosh, S., Chakraborty, P., Cohn, E., Brownstein, J. S., and Ramakrishnan, N. (2016). Characterizing diseases from unstructured text: A vocabulary driven word2vec approach. In Proceedings of the 25th ACM International on Conference on Information and Knowledge Management, pages 1129–1138. ACM.

Grove, S. K., Burns, N., and Gray, J. (2012). The practice of nursing research: Appraisal, synthesis, and generation of evidence . Elsevier Health Sciences.

Jegou, H., Douze, M., and Schmid, C. (2010). Product quantization for nearest neighbor search. IEEE transactions on pattern analysis and machine intelligence, 33(1):117– 128.

Joachims, T. (1998). Text categorization with support vector machines: Learning with many relevant features. In European conference on machine learning, pages 137–142. Springer.

Joulin, A., Grave, E., Bojanowski, P., and Mikolov, T. (2016). Bag of tricks for efficient text classification. arXiv preprint arXiv:1607.01759.

Kenter, T. and De Rijke, M. (2015). Short text similarity with word embeddings. In Proceedings of the 24th ACM international on conference on information and knowledge management, pages 1411–1420. ACM

Khatua, A., Khatua, A., and Cambria, E. (2019). A tale of two epidemics: Contextual word2vec for classifying twitter streams during outbreaks. Information Processing & Management, 56(1):247–257.

Kumavat, D. and Jain, V. (2015). Pos tagging approaches: A comparison. International Journal of Computer Applications, 118. 32-38.

Kusner, M., Sun, Y., Kolkin, N., and Weinberger, K. (2015). From word embeddings to document distances. In International conference on machine learning , pages 957–966

LeCun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. nature, 521(7553):436

Ling, W., Dyer, C., Black, A. W., and Trancoso, I. (2015). Two/too simple adaptations of word2vec for syntax problems. In Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies , pages 1299–1304

Lodhi, H., Saunders, C., Shawe-Taylor, J., Cristianini, N., and Watkins, C. (2002). Text classification using string kernels. Journal of Machine Learning Research , 2(Feb):419–444

Maaten, L. v. d. and Hinton, G. (2008). Visualizing data using t-sne. Journal of machine learning research, 9(Nov):2579–2605

Manns, B. J. (2015). Evidence-based decision-making 7: knowledge translation. In Clinical Epidemiology . Springer. 485-500

Mikolov, T. (2007). Language modeling for speech recognition in czech. Ph. D. dissertation, Masters thesis

Mikolov, T., Kopecky, J., Burget, L., Glembek, O., et al. (2009). Neural network based language models for highly inflective languages. In 2009 IEEE International Conference on Acoustics, Speech and Signal Processing , pages 4725–4728. IEEE.

Mikolov, T., Karafi ́at, M., Burget, L., Cernock`y, J., and Khudanpur, S. (2010). Recurrent neural network based language model. In Eleventh annual conference of the international speech communication association

Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013). Efficient estimation of word representations in vector space. arXiv preprint arXiv:1301.3781

Moraes, R., Valiati, J. F., and Neto, W. P. G. (2013). Document-level sentiment classification: An empirical comparison between svm and ann. Expert Systems with Applications, 40(2):621–633

Morin, F. and Bengio, Y. (2005). Hierarchical probabilistic neural network language model. In Aistats , volume 5, pages 246–252. Citeseer.

Neale, S., Donnelly, K., Watkins, G., and Knigh, D. (2018). Leveraging lexical resources and constraint grammar for rule-based part-of-speech tagging in welsh. Proceedings of the LREC 2018 Conference. 3946-3954.

Ratinov, L. and Roth, D. (2009). Design challenges and misconceptions in named entity recognition. CoNLL '09 Proceedings of the Thirteenth Conference on Computational Natural Language Learning. 147-155.

Rose Marie Nieswiadomy, P. (2012). An overview of qualitative research. Foundations of Nursing Research, 6th ed.:46,33

Saldana, J. (2015). The coding manual for qualitative researchers. Sage

Suter, W. N. (2012). Qualitative data, analysis, and design. Introduction To Educational Research , 2nd ed.:344

UnitedNations (2015). 17 goals to transform our world. Sustainable Development Goals. Available at https://www.un.org/sustainabledevelopment/health/ [Accessed: 06 February 2019]

Wei, G. and Wei, Y. (2018). Similarity measures of pythagorean fuzzy sets based on the cosine function and their applications. International Journal of Intelligent Systems, 33(3):634–652

WHO (2017). World bank and who. Available at https://www.who.int/news-room/detail/ [Accessed: 18 July 2019].

Yoshida, Zhang, Wen, T., and Tang, X. (2011). A comparative study of tf* idf, lsi and multi-words for text classification. Expert Systems with Applications, 38(3):2758–2765.

Zhang, X., Zhao, J., and LeCun, Y. (2015). Character-level convolutional networks for text classification. In Advances in neural information processing systems, pages 649–657

Zhu, Y., Yan, E., and Wang, F. (2017). Semantic relatedness and similarity of biomedical terms: examining the eﬀects of recency, size, and section of biomedical publications on the performance of word2vec. BMC medical informatics and decision making, 17(1):95.