**CAPSTONE PROJECT PROPOSAL**

****

**Project**

**Guided by Professor**

Denis Vrdoljak

**Submitted By**

Mytreyi Reddy

Pooja Varadaraj

Kavitha Kallakere Indrakumar

Sourabh Wadhwa

DeivaSubhaRanjani Pandurangan Ramamurthy

*Table of Contents*

[Abstract concept extraction of large datasets using NLP 3](#_Toc42271462)

[Problem Statement 5](#_Toc42271463)

[Objectives/Scope 6](#_Toc42271464)

[Business Requirements: 6](#_Toc42271465)

[Functional Requirements: 6](#_Toc42271466)

[NLP standard practices: 7](#_Toc42271467)

[Tools and Techniques used in NLP project implementation: 8](#_Toc42271468)

[Technique 1: Gensim Library -Doc2vec 8](#_Toc42271469)

[Technique 2: Topic Modelling 10](#_Toc42271470)

[Technique 3: Word2Vec model implementation using TensorFlow 11](#_Toc42271471)

[Technique 4: Text Embeddings 12](#_Toc42271472)

[Scoring Metrics used in our project: 14](#_Toc42271473)

[Heatmap Analysis: 16](#_Toc42271474)

[DevOps and Cloud (AWS): 17](#_Toc42271475)

[Team Weekly Deliverables 19](#_Toc42271476)

[Team Weekly Meeting Schedule: 20](#_Toc42271477)

[Deliverables 20](#_Toc42271478)

[Deliverable for Fall 2020: 20](#_Toc42271479)

[Our Team: 21](#_Toc42271480)

# **Abstract concept extraction of large datasets using NLP**

NLP is the ability to extract and act upon information from natural languages is a way meaningful to applications and machine learning systems.[[1]](#footnote-1)

Concepts are sequences of words that represent real or imaginary entities or ideas that users are interested in. Abstract Concept extraction, a subdomain of natural language processing (NLP) with a focus on extracting concepts of interest, has been adopted to computationally extract information from text for a wide range of applications in the past few years.

The abstract concept extraction can be classified based on:

1. Macro concept extraction– provides a general understanding of the document as a whole.

* Typically performed with statistical techniques
* It is used for: clustering, categorization, similarity, topic analysis, word clouds, and summarization

1. Micro concept extraction– extracts understanding from individual phrases or sentences.

* Typically performed with NLP techniques
* It is used for: extracting facts, entities (see above), entity relationships, actions, and metadata fields

Building a web of concepts will form the backbone of the next generation of search technology and it will also help in analyzing large datasets easily. A web of concepts would not only allow us to identify user intent better, but also to rank content better, support more expressive queries and present the integrated information better. Our definition of a concept is based on its usefulness to people.That is, a string is a concept if a “significant” number of people say it represents an entity, event or topic known to them. For instance, “Flying Pigs Shoe Store” is not a concept if only one or two people know about this store, even though this store may have a web page where “Flying Pigs Shoe Store” appears.[[2]](#footnote-2)

Once we have built the pipeline, there are multiple approaches to automate this extraction process. They are:

1. **Rule-based Approach**: We define a set of rules for the syntax and other grammatical properties of a natural language and then use these rules to extract information from text
2. **Supervised:** Let’s say we have a sentence S. It has two entities E1 and E2. Now, the supervised machine learning model has to detect whether there is any relation (R) between E1 and E2. So, in a supervised approach, the task of relation extraction turns into the task of relation detection. The only drawback of this approach is that it needs a lot of labeled data to train a model
3. **Semi-supervised:** When we don’t have enough labeled data, we can use a set of seed examples (triples) to formulate high-precision patterns that can be used to extract more relations from the text

This capstone will build analytics models to characterize the COVID-19 outbreak.We will initially collect the COVID data and implement the NLP (Natural Language Processing), Machine learning algorithms to build a model that can input any dataset and gives the characteristics of the data by performing the analysis.

We would like to extract information from the corona data sources starting with standard techniques like bag of words concept, vectorisation techniques and document similarity scores then proceed with compressed vectors created by encoders to create the abstract of any dataset. This can be used in decreasing the defects and increasing the code quality in the data pipelines.

# **Problem Statement**

The recent corona outbreak gave the Data enthusiast the real time challenge. To come up with the analysis within the shortest time, to understand each and every dataset, choosing the right techniques for the selected dataset is always a challenge, given the large amount of data our industries used. In the world of Big data, effective concept extraction using Natural language processing (NLP) to extract information for knowledge graphs is identified as a problem. We would like to take this project to build a model that takes any dataset as an input to build a knowledge graph which is incorporated into improving the code quality and decreasing defects in the development pipelines. We would start with the corona datasets to identify the outbreak and analyze the different aspects and measures.

# **Objectives/Scope**

By gathering data from different sources to create the knowledge base about symptoms and treatments for the most pandemic illness. The data set used in the project has been collected from various systems and the objective is to predict whether or not a patient has symptoms of the illness based on certain diagnostic measurements included in the data set.

Build a model using different Machine Learning algorithms to accurately predict the contingency of disease and outbreak. This helps to take the required measures and take important decisions.

To build a data science pipeline that takes corpus of texts as input, concept and automates the vectorization model. The scope of the project is to generalize the NLP model for abstract data, which predicts not only for Coronavirus Dataset, it predicts for any data set.

To analyze the free text engineers, spend a lot of time vectorizing the text and this consumes a lot of bandwidth and slows the process of execution. To overcome this model will act as a plugin for any NLP process which is automated to do vectorization. This model can be used predominantly in building Chatbots, based on the vectorized training data, it would have the ability to interpret the text.

## **Business Requirements:**

1. The System will provide a semi-automated system that can create the text vectorization.
2. The Model must be domain independent and should provide the initial machine learning model on text data.
3. The Model will also create a pipeline for building text vectorization.

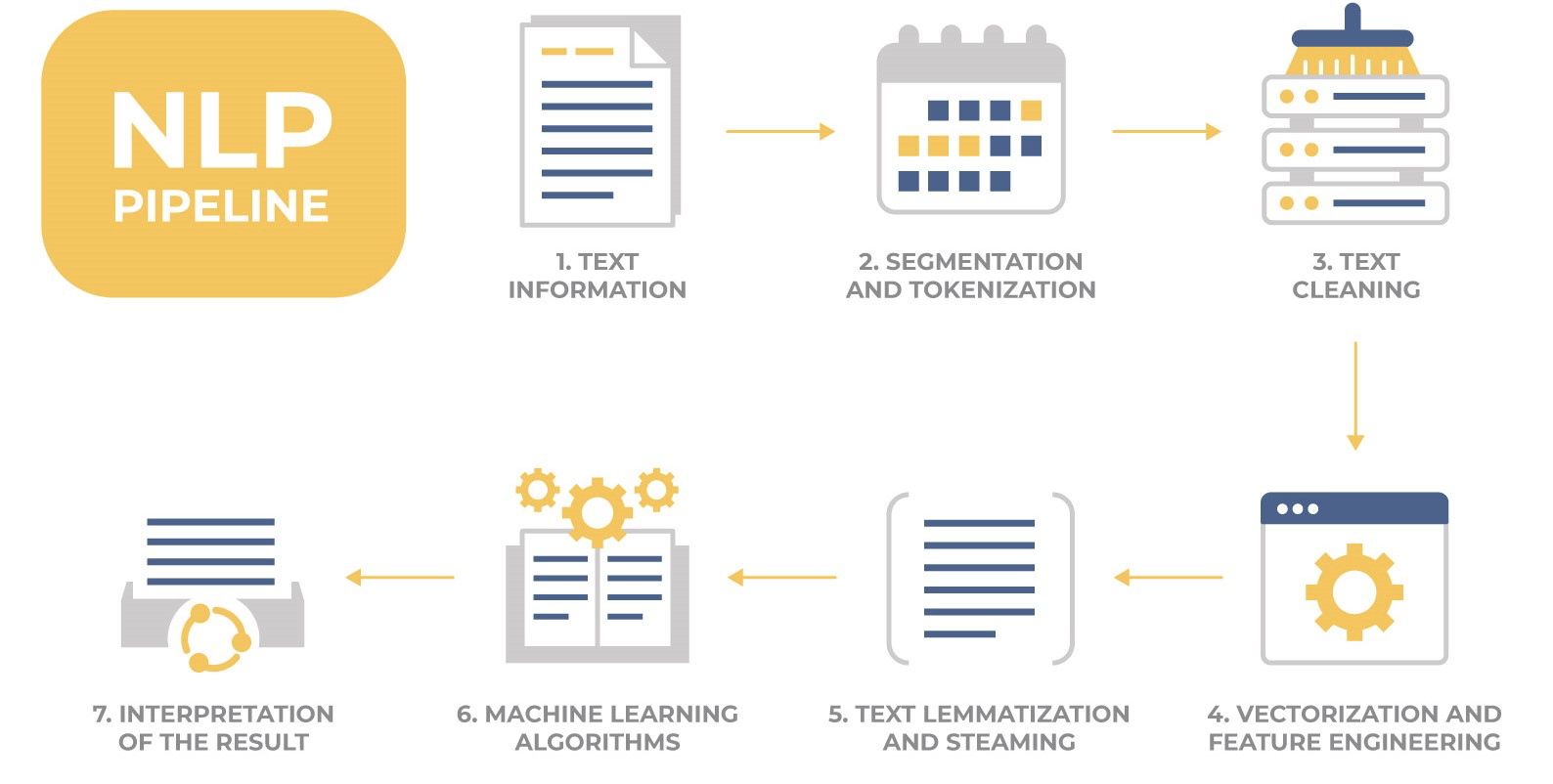
## **Functional Requirements:**

1. The model will take the ability to take any dataset as the input without the constraints
2. The Model will vectorize the input data.
3. The model will compile all the documents, feature them and compare their features.
4. The model will use compressed vectors created by the encoder
5. The model will be used to develop the knowledge graphs based on the data input.

# **NLP standard practices:**

Typically, any NLP-based problem can be solved by a methodical workflow that has a sequence of steps.We usually start with a corpus of text documents and follow standard processes of text wrangling and pre-processing, parsing and basic exploratory data analysis. Based on the initial insights, we usually represent the text using relevant feature engineering techniques. Depending on the problem at hand, we either focus on building predictive supervised models or unsupervised models, which usually focus more on pattern mining and grouping. Finally, we evaluate the model and the overall success criteria with relevant stakeholders or customers, and deploy the final model for future usage.[[3]](#footnote-3)

1. Text Information
2. Segmentation and tokenization
3. Text Cleaning
4. Vectorization and Feature Engineering
5. Text Lemmatization and Stemming
6. NLP Libraries/ Machine Learning Algorithms
7. Interpretation of the results.

**[[4]](#footnote-4)**

# **Tools and Techniques used in NLP project implementation:**

## **Technique 1: Gensim Library -Doc2vec**

Gensim = “Generate Similar” is a popular open source natural language processing (NLP) library used for unsupervised topic modeling. It uses top academic models and modern statistical machine learning to perform various complex tasks such as −

* Building document or word vectors
* Corpora
* Performing topic identification
* Performing document comparison (retrieving semantically similar documents)
* Analyzing plain-text documents for semantic structure

Apart from performing the above complex tasks, Gensim, implemented in Python and Cython, is designed to handle large text collections using data streaming as well as incremental online algorithms. This makes it different from those machine learning software packages that target only in-memory processing.

**Doc2Vec Implementation:**

Doc2Vec model, is opposite to the Word2Vec model, is used to create a vector representation of a group of words taken collectively as a single unit. It doesn’t only give the simple average of the words in the sentence.

Words maintain logical (grammatical) structure but documents don’t have any logical structures. To solve this problem another vector (Paragraph ID) needs to be added with the word2vec model. Two variations of doc2vec are available:

* **Distributed Memory Model of Paragraph Vectors (PV-DM)** - Distributed Memory (DM) model is similar to Continuous-Bag-of-Words (CBOW) model in word2vec which attempts to guess the output (target word) from its neighboring words (context words) with the addition of a paragraph ID.
* **Distributed Bag of Words version of Paragraph Vector (PV-DBOW) -** Distributed Bag-Of-Words (DBOW) Model similar to skip-gram model of word2vec, which guesses the context words from a target word. The only difference between skip-gram and distributed bag of words (DBOW) is instead of using the target word as the input, Distributed Bag of Words (DBOW) takes the document ID (Paragraph ID) as the input and tries to predict randomly sampled words from the document.

**For our project we have chosen the PV-DM model because it preserves the word order in a document whereas PV-DBOW just uses the bag of words approach, which doesn’t preserve any word order.**

**Steps followed in our Project for Doc2Vec Implementation:**

1. We tagged the using the lemmitized column.
2. Trained the model using the tagged data.
3. Loaded the model.
4. Saved the model.
5. Using the infer vector on the saved model we convert any string input into a vector.
6. Using cosine similarity we find the similarity between the different strings passed based on our trained model.

## **Technique 2: Topic Modelling[[5]](#footnote-5)**

In machine learning and natural language processing, a topic model is a type of statistical model for discovering the abstract "topics" that occur in a collection of documents. Topic modeling is a frequently used text-mining tool for discovery of hidden semantic structures in a text body. It is an unsupervised approach used for finding and observing the bunch of words (called “topics”) in large clusters of texts. Topics can be defined as “a repeating pattern of co-occurring terms in a corpus”. Topic Models are very useful for the purpose for document clustering, organizing large blocks of textual data, information retrieval from unstructured text and feature selection.[[6]](#footnote-6)

We are using Latent Dirichlet Allocation which is the most popular topic modeling technique.

LDA assumes documents are produced from a mixture of topics. Those topics then generate words based on their probability distribution. Given a dataset of documents, LDA backtracks and tries to figure out what topics would create those documents in the first place.

Parameters of LDA

* Alpha and Beta Hyperparameters – alpha represents document-topic density and Beta represents topic-word density. Higher the value of alpha, documents are composed of more topics and lower the value of alpha, documents contain fewer topics. On the other hand, higher the beta, topics are composed of a large number of words in the corpus, and with the lower value of beta, they are composed of few words.
* Number of Topics – Number of topics to be extracted from the corpus.
* Number of Topic Terms – Number of terms composed in a single topic. It is generally decided according to the requirement. If the problem statement talks about extracting themes or concepts, it is recommended to choose a higher number, if the problem statement talks about extracting features or terms, a low number is recommended.
* Number of Iterations / passes – Maximum number of iterations allowed to LDA algorithm for convergence.[[7]](#footnote-7)

## **Technique 3: Word2Vec model implementation using TensorFlow**

TensorFlow is an end-to-end open source platform for machine learning. It has a comprehensive, flexible ecosystem of tools, libraries and community resources that lets researchers push the state-of-the-art in ML and developers easily build and deploy ML powered applications.

How word2vec works:

1. The idea behind word2vec is that:
2. Take a 3-layer neural network. (1 input layer + 1 hidden layer + 1 output layer)
3. Feed it a word and train it to predict its neighboring word.
4. Remove the last (output layer) and keep the input and hidden layer.

Now, input a word from within the vocabulary. The output given at the hidden layer is the ‘word embedding’ of the input word.

High-level steps followed in implementing tensor flow:

1. We need to convert the input into an output pair such that if we input a word, it should predict the neighboring words: the n words before and after it, where n is the parameter window size.
2. We created a dictionary which translates words to integers and integers to words.
3. Listed our sentences as a list of words.
4. Generate our training data. This basically gives a list of word, word pairs. (we are considering a window size of 2)
5. We convert our training data into one hot vectors.
6. We created our tensor flow model to convert the training data into embedded representations.
7. We have the embedded dimension and make a prediction about the neighbor. To make the prediction we used SoftMax.
8. Now we train our model.
9. We tried to modify our W1 and B1 values to get the least loss value.

We implemented the tensor flow in AWS Sagemaker since the computation time required on our local systems was too long.

## **Technique 4: Text Embeddings**

Text Embeddings are real valued vector representations of strings. We build a dense vector for each word, chosen so that it’s similar to vectors of words that appear in similar contexts. Word embeddings are considered a great starting point for most deep NLP tasks. We are using Skip gram and CBoW for Word2vec:

* [[8]](#footnote-8)Skip-Gram: Skip-gram learns to predict the context words from a given word, in case where two words (one appearing infrequently and the other more frequently).

SoftMax Objective: Skip-gram’s objective is to predict the contexts of a given target-word. The contexts are immediate neighbors of the target and are retrieved using a window of an arbitrary size *n* — by capturing *n* words to the left of the target and *n* words to its right.

Architecture: Skip-gram is a simple neural network with only one hidden layer. The input to the network is a one-hot encoded vector representation of a target-word — all of its dimensions are set to zero, apart from the dimension corresponding to the target-word. The output is the probability distribution over all words in the vocabulary which defines the likelihood of a word being selected as the input word’s context

Steps:

1. NLP process - cleaning and tokenize
2. Word2Vec constructor takes a broad range of parameters, but we will only concentrate on a few that are most relevant:
   1. **sentences** — The iterable over the tokenized sentences we will train on (the Brown sentences).
   2. **size** — The dimensionality of our embeddings.
   3. **window** — This determines which words are considered contexts of the target. For the window of size *n* the contexts are defined by capturing n words to the left of the target and n words to its right.
   4. **min\_count** — We can use this parameter to tell the model to ignore some infrequent words — don’t create an embedding for them and don’t include them as contexts.
   5. **negative** — Defines the number of negative samples (incorrect training pair instances) that are drawn for each good sample (see the Skip-gram section).
   6. **iter** — How many epochs do we want to train for — how many times we want to pass through our training data.
   7. **workers** — Determines how many worker threads will be used to train the model.
3. Define and Train the Model
4. Evaluating the Model
5. Building a Context Dependent Model

**Advantages of Skip-Gram Model**

1. Skip-gram model can capture two semantics for a single word. i.e it will have two vector representations of Apple. One for the company and other for the fruit.
2. Skip-gram with negative sub-sampling outperforms every other method generally.

[[9]](#footnote-9)Continuous Bag of Words: The way CBOW works is that it tends to predict the probability of a word given a context. A context may be a single word or a group of words. But for simplicity, I will take a single context word and try to predict a single target word.

Suppose, we have a corpus C = “Hey, this is a sample corpus using only one context word.” and we have defined a context window of 1.

The flow is as follows:

1. The input layer and the target, both are one- hot encoded of size [1 X V]. Here V=10 in the above example.
2. There are two sets of weights. one is between the input and the hidden layer and second between the hidden and output layer.  
   Input-Hidden layer matrix size =[V X N] , hidden-Output layer matrix size =[N X V] : Where N is the number of dimensions we choose to represent our word in. It is arbitrary and a hyper-parameter for a Neural Network. Also, N is the number of neurons in the hidden layer. Here, N=4.
3. There is a no activation function between any layers.( More specifically, I am referring to linear activation)
4. The input is multiplied by the input-hidden weights and called hidden activation. It is simply the corresponding row in the input-hidden matrix copied.
5. The hidden input gets multiplied by hidden- output weights and output is calculated.
6. Error between output and target is calculated and propagated back to re-adjust the weights.
7. The weight between the hidden layer and the output layer is taken as the word vector representation of the word.

**Advantages of CBOW:**

1. Being probabilistic is nature, it is supposed to perform superior to deterministic methods(generally).
2. It is low on memory. It does not need to have huge RAM requirements like that of a co-occurrence matrix where it needs to store three huge matrices.

**Disadvantages of CBOW:**

1. CBOW takes the average of the context of a word (as seen above in calculation of hidden activation). For example, Apple can be both a fruit and a company but CBOW takes an average of both the contexts and places it in between a cluster for fruits and companies.
2. Training a CBOW from scratch can take forever if not properly optimized.

# **Scoring Metrics used in our project:**

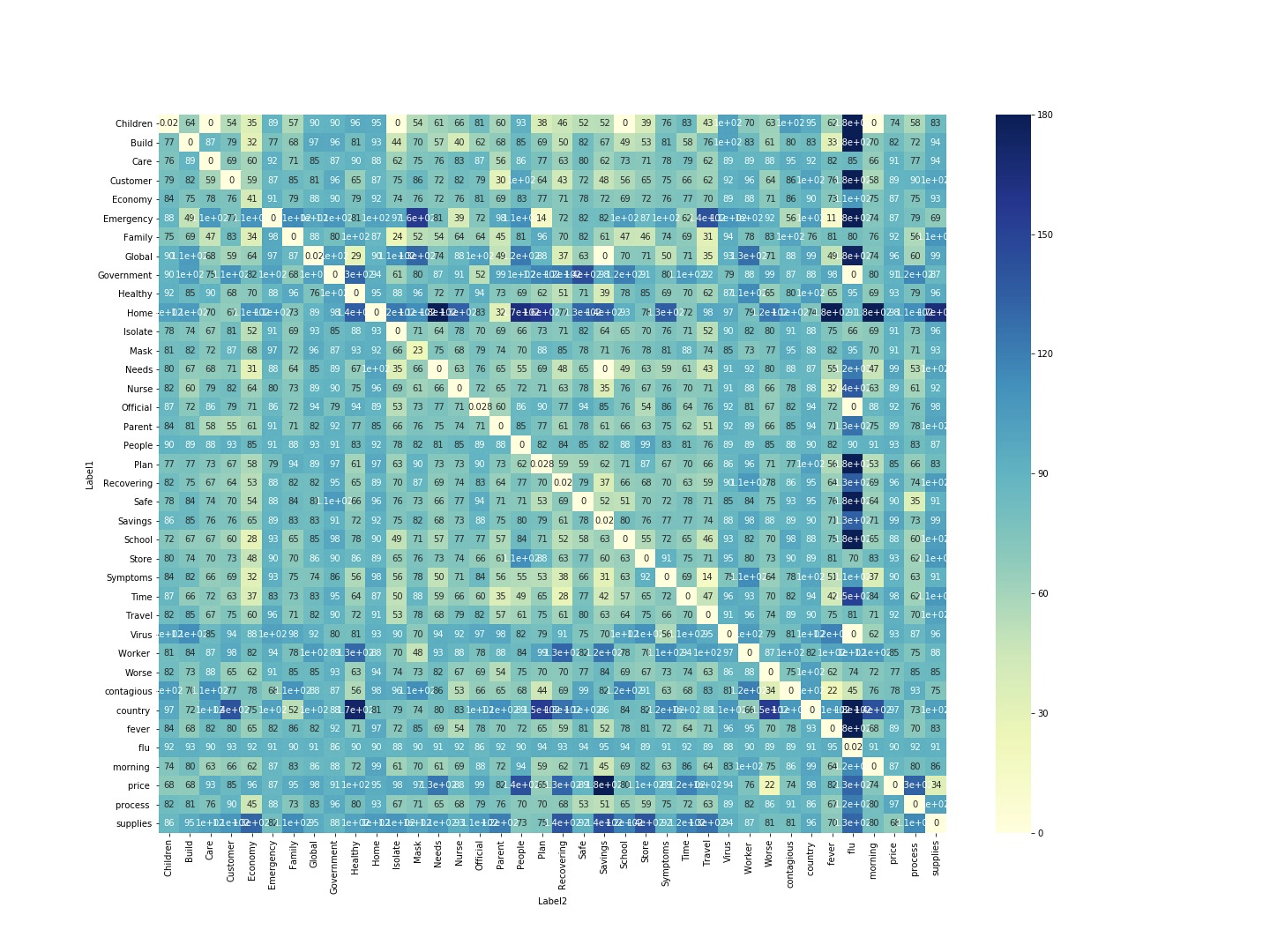
1. **Cosine Similarity:** Cosine similarity is a metric used to measure how similar the documents are irrespective of their size. Mathematically, it measures the cosine of the angle between two vectors projected in a multi-dimensional space. The cosine similarity is advantageous because even if the two similar documents are far apart by the Euclidean distance (due to the size of the document), chances are they may still be oriented closer together. The smaller the angle, higher the cosine similarity and larger the angle the lesser is the similarity between the strings taken as input.
2. **Cost Function:** It is a function that measures the performance of a Machine Learning model for given data. Cost Function quantifies the error between predicted values and expected values and presents it in the form of a single real number. Depending on the problem Cost Function can be formed in many different ways. The purpose of Cost Function is to be either:

* Minimized - then returned value is usually called cost, loss or error. The goal is to find the values of model parameters for which Cost Function returns as small a number as possible.
* Maximized - then the value it yields is named a reward. The goal is to find values of model parameters for which the returned number is as large as possible.

The most common cost function is a quadratic function. It is usually an average value since we just need a single cost value for our model.

**We have the basic cost function model and Euclidean distance model ready for implementation in the fall quarter.**

## **Heatmap Analysis:**



The words or phrases that are not correlated mean they are completely different topics. But the words or phrases that show a cosine similarity of 0, means they are highly similar.

**Highly correlated words:**

Words such as “virus” and “flu”, “contagious” and “fever”, “emergency” and “fever” are showing high similarity so that they can be correlated to each other. These words are related and can be meaningful in a given context.

**Less correlated words:**

Words such as “travel” and “people”, “contagious” and “home”, “isolate” and “virus” show very less similarity and they cannot be correlated to each other. These words do not go together or they can mean different things.

**Zero Degree Vectors or Vectors that have smaller Angles - Similar Vectors**

In general context, supply has a positive correlation with price. Price and Supplies in heat map have a value of 34 degrees which means that these vectors are closer to similarity angles. When two different phrases or labels such as “price” and “worse” are analyzed in this model, it gives a result of 22 degrees corresponding to a similarity angle.

**90 Degree Vectors or Vectors that have higher Angles - Orthogonal Vectors**

Words such as “Nurse” and “Care” are related in general context but when tested on the model resulted in 79 degrees which means that they are not related and not similar. Other words such as Store and government or Store and Emergency which are unrelated in general context resulted in 90-degree value representing the orthogonality of these phrases.

**180 Degree Vectors or Vectors that have above 90-degree Angles - Opposite Vectors**

“Country” and “Flu”, “Fever” and “flu” have an angle of 180 degrees, which states that they are completely opposite to each other. Also, “Country” and “fever” have an angle of more than 150 degrees. This result can be compared to a mathematical formula such as

“If “a” and “b” are similar,

“b” and “c” are similar then

“a” and “c” should be equal.”

Therefore, based on the COVID articles data set that is trained and the test data (phrases/synonyms list) that is given, the model generates similar vectors, orthogonal vectors and the vectors that are completely opposite and not related to each other.

# **[[10]](#footnote-10)DevOps and Cloud (AWS):**

Employing Amazon SageMaker Service[[11]](#footnote-11) for building a DevOps Pipeline for deploying Machine learning models.

Amazon SageMaker is an AWS service used to build, train and deploy machine learning (ML) models. Automating the build and deployment of machine learning models is an important step in creating production machine learning services. Models need to be retrained and deployed when code and/or data are updated. It provides a CI/CD(continuous Integration/Continuous Delivery) workflow[[12]](#footnote-12)

SageMaker supports frameworks and libraries[[13]](#footnote-13) such as TensorFlow and Apache MXNet out-of-the-box. It comes with some built-in algorithms, for instance, PCA, K-Means and XGBoost.

Apache Spark can be used to pre-process the data. Package any machine learning algorithm into a Docker container and plug it into SageMaker’s training-serving pipeline.

Amazon SageMaker offers[[14]](#footnote-14):

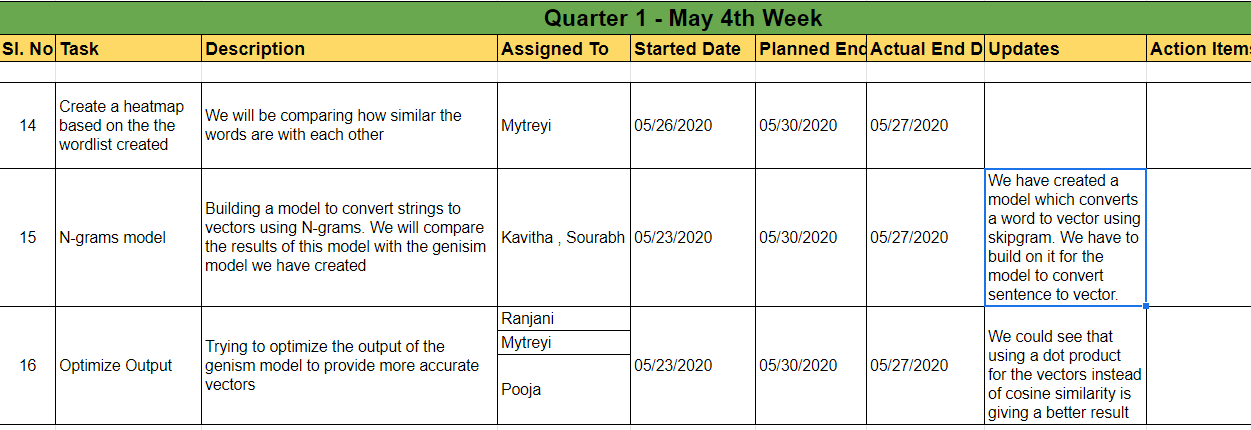
* Habitual environment (Jupyter Notebook, python);
* popular ML frameworks (Apache MXNet, TensorFlow);
* option to bring any other frameworks and libraries;
* “zero-configuration”workflow for training;
* out-of-the-box support for multi-node training;
* straightforward deployment of trained models to production.

**Success metrics** :

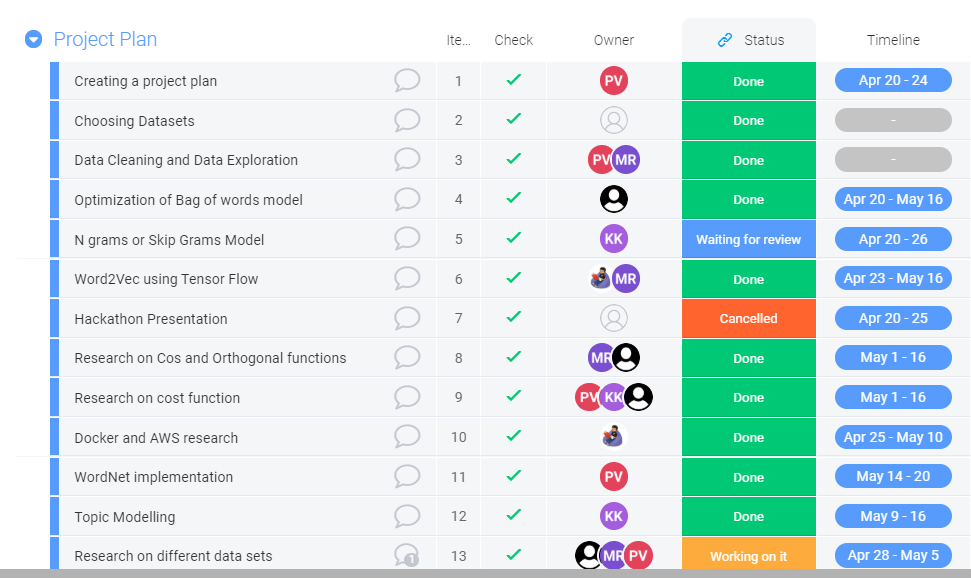
1. Vectorize an arbitrary X input.
2. Performs better than the baseline bag of words model.
3. Using scoring metrics to analyse and check the accuracy of our model.
4. Any data set can be used for this model to analyse factors and take measures or decisions.

# **Team Weekly Deliverables**

We are maintaining an excel sheet in which we have all the task details assigned to each of us on a weekly basis after our Saturday call with the Professor. We also update our action items, blockers and status for each task as well.

****

We have also created a project plan using monday.com where we can track our task details.

****

# **Team Weekly Meeting Schedule:**

|  |  |  |
| --- | --- | --- |
| **Days of Meeting** | **Hours of Meeting** | **Platform of Meeting** |
| Wednesday | 1:30 PM to 3:30 PM | Zoom Call |
| Friday | 1:30 PM to 3:00 PM | Zoom Call |
| Saturday | 11:00 AM to 12:00 PM | Zoom Call |

# **Deliverables**

As a team, our goal is to provide the following deliverables by the end of the implementation phase as per the timelines mentioned in the project plan:

A final report that:

* Deliver a working NLP model that converts arbitrary input strings into vectors.
* The system architecture design.
* Codebase for the system.
* Documents the process we underwent from idea to implementation.
* A final presentation and demo of our working model to our advisor and faculty members

# **Deliverable for Fall 2020:**

Deliver the codebase for a model that can be trained on generalized data and to demonstrate the model on any corpus of data.

# **Our Team:**

**Sourabh Wadhwa**

Sourabh Wadhwa is a MSIS student at Santa Clara University. He has 5 years and 5 months of work experience before joining the Master’s program. Working on various projects as part of his work have given him proficiency in various programming languages like Java, SQL and multiple Operating Systems including Red Hat Linux, CentOS and Windows Server. One of his projects got him interested in Machine Learning and displaying the power of data based predictions and how they are being implemented in the finance domain. The program has so far given him to explore machine learning in multiple other domains and how various businesses now have specialized data analytics for their domain. The various projects done has helped broaden his domain knowledge and understand how data science is implemented in different scenarios.

**DeivaSubhaRanjani Pandurangan Ramamurthy**

DeivaSubhaRanjani is currently pursuing Masters of Science in Information Systems. She has 3 years of work experience in the software industry. She is always enthusiastic to play with data, analyze, derive insights and make valuable business decisions. She has worked as a Product Analyst and played a pivotal role in developing Multi-dimensional Analytics and Reporting Solution (MARS), a self-service BI agile solution enabling business users to create, publish and collaborate on analytics and operational reports. She also has worked with large complex sales datasets to analyze the customer purchase trends and performed competitive market analysis for our client. She has hands-on experience in Java, Python, SQL, VBA and Unix shell scripting. She works well with tools like Tableau, Pentaho, MS excel and MySQL. During her Master’s journey, courses like Data Science with Python, Business Intelligence with Data Warehousing have helped her to take the data, cleanse them and turn them into valuable insights. She is very eager to work with this team and analyze CoronaVirus data and help us understand more about the situation using various Machine Learning Techniques

**Kavitha Kallakere Indrakumar**

Kavitha Kallakere Indrakumar is pursuing Masters in Information Systems.She has worked on different technologies and has 10 years of work experience. She has taken courses on System analysis and design,Database Management system ,Data Science with Python,Big Data and Modelling and Software Project Management.

She wants to work with our team who have experience in different sectors and learn more on Data analyses and build analytics and visualization. She is looking forward to learning more on Machine Learning Techniques and Tools.

**Pooja Varadaraj**

Pooja Varadaraj is a data-driven professional, currently pursuing a Master’s of Science in Information Systems with a focus on Data Analytics at Santa Clara University. Prior to her Masters, she was working as a SAP BI Consultant at Tech Mahindra,India for 3 years. She has solved Business Intelligence issues for the global food giant Nestle by working on live production data. She has hands-on experience in programming languages like python, java and SQL. She has also worked on ETL(pentaho) and data visualizations tools (Tableau and Tableau prep) for her academic course work.

She had taken up courses like data science with python in her master’s which created an interest and passion for her to work on a data science capstone. She is eagerly looking forward to ameliorating her data analytics skills and at the same time contributing to understanding the current pandemic situation arising in the world.

**Mytreyi Reddy**

Mytreyi is a MSIS student from Santa Clara University. She started her career working as a Systems Engineer for 2 and a half years at Tata Consultancy Services. She worked on server-side web applications using C# and Dot Net framework. She has hands-on experience in Java, SQL and she worked on Python, Tableau, MySQL, Pentaho in my academic projects. Currently, she is working more on analytics, accessing data from multiple sources, analyzing and coming up with descriptive insights to make decisions on the business and predictive analytics on the data which would result in better decision making. She is in the phase of learning machine learning models and applying them on the data which would derive accurate results. This learning process has driven her to gain an understanding, prioritizing analytic features, and knowledge on the functionality on the end-end of a product.

1. Quoted by Professor Denis Vrdoljak- <https://www.youtube.com/watch?v=it34KxYMQRQ> [↑](#footnote-ref-1)
2. <http://ilpubs.stanford.edu:8090/917/1/conceptMining-Techrep.pdf> [↑](#footnote-ref-2)
3. <https://towardsdatascience.com/a-practitioners-guide-to-natural-language-processing-part-i-processing-understanding-text-9f4abfd13e72> [↑](#footnote-ref-3)
4. <https://www.datasciencecentral.com/profiles/blogs/top-nlp-algorithms-amp-concepts> [↑](#footnote-ref-4)
5. <https://www.analyticsvidhya.com/blog/2016/08/beginners-guide-to-topic-modeling-in-python/> [↑](#footnote-ref-5)
6. [https://en.wikipedia.org/wiki/Topic\_model#:~:text=In%20machine%20learning%20and%20natural,structures%20in%20a%20text%20body](https://en.wikipedia.org/wiki/Topic_model#:~:text=In%20machine%20learning%20and%20natural,structures%20in%20a%20text%20body.) [↑](#footnote-ref-6)
7. [↑](#footnote-ref-7)
8. <https://blog.cambridgespark.com/tutorial-build-your-own-embedding-and-use-it-in-a-neural-network-e9cde4a81296> [↑](#footnote-ref-8)
9. <https://www.analyticsvidhya.com/blog/2017/06/word-embeddings-count-word2veec/> [↑](#footnote-ref-9)
10. <https://aws.amazon.com/devops/> [↑](#footnote-ref-10)
11. <https://aws.amazon.com/getting-started/hands-on/build-train-deploy-machine-learning-model-sagemaker/> [↑](#footnote-ref-11)
12. <https://hackernoon.com/should-i-use-amazon-sagemaker-for-deep-learning-dc4ae6b98fab> [↑](#footnote-ref-12)
13. <https://github.com/aws-samples/aws-sagemaker-build> [↑](#footnote-ref-13)
14. <https://towardsdatascience.com/a-practitioners-guide-to-natural-language-processing-part-i-processing-understanding-text-9f4abfd13e72> [↑](#footnote-ref-14)