National College of Ireland

Speech utterance classification for triage of 911 emergency calls

MSc Research Project
Data Analytics

Viraj Pawar
x16112521

School of Computing
National College of Ireland

Supervisor: Lisa Murphy

National College of Ireland
Project Submission Sheet – 2015/2016
School of Computing

Student Name:	Viraj Pawar
Student ID:	x16112521
Programme:	Data Analytics
Year:	2016
Module:	MSc Research Project
Lecturer:	Lisa Murphy
Submission Due Date:	16/08/2017
Project Title:	Speech utterance classification for triage of 911 emergency calls
Word Count:	XXX

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL internet material must be referenced in the bibliography section. Students are encouraged to use the Harvard Referencing Standard supplied by the Library. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action. Students may be required to undergo a viva (oral examination) if there is suspicion about the validity of their submitted work.

Signature:	
Date:	16th September 2017

PLEASE READ THE FOLLOWING INSTRUCTIONS:

1. Please attach a completed copy of this sheet to each project (including multiple copies).
2. **You must ensure that you retain a HARD COPY of ALL projects**, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer. Please do not bind projects or place in covers unless specifically requested.
3. Assignments that are submitted to the Programme Coordinator office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Speech utterance classification for triage of 911 emergency calls

Viraj Pawar
x16112521
MSc Research Project in Data Analytics

16th September 2017

Abstract

The utilisation of speech utterance classification for triage of 911 emergency calls is described in this research paper. Using speech utterance of caller to analyze and prioritize 911 emergency calls can reduce call wait time and enable the call taker to make better and faster decisions. The objective of the research is to examine and compare the performance of four most widely used text classification algorithms, Logistic Regression, Support Vector Machine (SVM), AdaBoost and Naive Bayes for speech utterance classification and prioritization of 911 calls. The research utilised transcripts of 911 audio calls that were created using IBM, Google and CMUSphinx Automatic Speech Recogniser (ASR) as training datasets. The classifiers were trained on numerical feature vectors of bag-of-words from each training dataset and tested on manually transcribed first utterance of 911 calls and were evaluated on classification accuracy and F-score. Furthermore, these classifiers were trained on three different features vectors from Bag-of-words, tf-idf weighting and word2vec model of Google ASR generated dataset to verify if word2vec features vectors improve classifier's accuracy. The result showed there was no improvement in classification accuracy when word2vec features vectors were used. Support Vector Machines achieved the highest classification accuracy of 73% when feature vectors of bag-of-words from Google ASR generated dataset were used for training.

1 Introduction

In North America, 911 emergency telephone number is used in emergency circumstances. A call made to 911 is routed to a nearest public-safety answering point (PSAP) where it is received by a call taker. Calls are received as they come in and are prioritised based on the nature of urgency. Based on priority, a call is routed to an appropriate dispatcher for handling (LEITSC; 2006). A large volume of calls is received in a dispatch centre which leads to a long wait time when calls with low priority are received first over a call requiring immediate assistance. Moreover, a call taker receives calls reporting the same incident which increases the call wait time for other type of calls. An efficient way is to determine the priority of a call before it is received and cluster calls based on priority, which will enable a call taker to make better and faster decisions. Statistical machine learning classifier and Natural language processing can be used to triage calls based on features extracted from a callers' utterance. Based on the circumstances of the situation reported, calls will be priority labelled as high, medium or low and clustered on labels and displayed to a call taker adding more intel to calls. Calls reporting the same incident will be assigned a single ID and clustered to avoid multiple calls reporting the same incident which will reduce call wait time on other types of call. However, triage by call classification cannot completely take over human judgement for analyzing calls. The research proposes to use two statistical models, Automatic Speech Recognition (ASR) to transcribe the speech utterance and a classification model to map the transcript to an appropriate priority label for triage. There are several technical challenges in this approach: 1. Accurate transcript of callers' utterance will be needed to classify the call. 2. The system should unambiguously understand caller's utterance and map it to a right priority label. 3. A large amount of accurate transcripts will be needed to train the system.

1.1 Speech utterance classification

Speech utterance classification has many applications in spoken language understanding tasks. One of the real life application is call routing in call centres, which utilises the first utterance of a caller to determine a destination of the call within a call centre (Attwater et al.; 2000; Chu-Carroll and Carpenter; 1999). Two statistical models are used in speech utterance classification, Automatic Speech Recognition model, and classification model. The main component of this system is the classification model which needs to classify text of the caller's transcript accurately. Vector space model, discriminative and generative models have been used for speech utterance classification in call routing task (Chu-Carroll and Carpenter; 1999; Sarikaya et al.; 2005). Training the classifier requires manual transcripts and annotation of the classification destination (Wang et al.; 2006). Spontaneous speech, Language variation, ambient noise present a major challenge for ASR which affects the transcripts that need to classified (Schapire et al.; 2005). Due to high word error rate in the transcripts, the classification error increases. Different techniques have been used to improve classification accuracy and reduce classification error rate (Sarikaya et al.; 2005; Niyogi et al.; 2000; Faruquie et al.; 2007; Horvitz et al.; 2007; Zitouni et al.; 2001; Kuo and Lee; 2003; Garfield and Wermter; 2006). The research proposes to utilise speech utterance classification for triage of 911 calls. The focus of research approach is on successful call classification by the classifiers. In real time application for triage of calls using speech utterance will need accurately transcribed text from ASR.

The accuracy of the transcripts will decide the call label. Inaccurate transcripts will lead to the wrong classification of calls. The system needs to be trained and tested on accurately transcribed text. Manually transcribed text are considered most accurate transcripts compared to ASR generated transcripts. However, in real time application training classifiers on manually transcribed text is inefficient and time-consuming. The classification model needs to be trained on ASR generated transcripts since it is fast, efficient and new transcripts can be added any time for training to improve classification accuracy. However, these transcripts have less accurate text. The research utilises ASR generated transcripts for training the classifiers and tests on the manually transcribed first utterance of the caller from each call to examine and compares the performance of classifiers on three ASR generated training datasets. With the assumption that in real time application the classification model will classify correct utterance transcripts, the classification model is tested on the manually transcribed first utterance of the calls. Since classifiers are trained using supervised learning method, all datasets are manually labelled and fed to the classifiers for training. Using bag-of-words numerical feature vectors from the three datasets created using Google, IBM and CMUSphinx ASR, the classifiers accuracy and f-score is compared on these datasets. Bag-of-word count vectors, tf-idf vectors, and word2vec word embedding are three different techniques to extract features from a text by converting it into feature vectors. The classification accuracy depends on the features extracted from the text. To find which features extraction techniques produces best classification accuracy and to test the hypothesis, the classifiers performance is compared on these feature vectors.

1.2 Research objective

The objective of the research is to examine different techniques to build a good classification model using real 911 audio calls transcripts to train the classifiers. The research also investigates the potential of machine learning classifiers on how successfully it classifies all call classes based on the features extracted from the utterance of the call. The research uses technique like boosting to improve the accuracy of the classifiers. Bag-of-words, tf-idf and word2vec features vectors are used for training the classifiers. With these features as input to classifiers, a comparison of accuracy between the classifiers is made. Garfield and Wermter (2006) highlights the use of alternative approach for feature representation of words which preserve syntactical and semantic information of words in the speech utterance classification task. The research fills the gap by using word embeddings from word2vec model which considers the semantic similarity of words in vector form to obtain better classification performance.

1.3 Research Question

Can speech utterance classification be utilised for triage of 911 emergency calls?

1.3.1 Research hypothesis

Word embeddings from word2vec model capture semantic similarity of words in a document (Mikolov, Sutskever, Chen, Corrado and Dean; 2013). Word2vec model uses distributed representation of words to group similar words capturing syntactic and semantic word relationship, which helps the learning algorithm to achieve better performance (Mikolov, Sutskever, Chen, Corrado and Dean; 2013). Preserving semantic of words

in utterance provides a deeper meaning of the words in the utterance which is important in call classification task. Preserving semantic word relationship in feature vectors of the utterances of the calls will improve the accuracy of the classifiers. With this hypothesis, the accuracy of the classifiers using word2vec feature vectors is compared with accuracy on feature vector from bag-of-words and tf-idf to verify whether word2vec feature vectors improve the classifiers accuracy.

1.4 Dataset

Training dataset is created by transcribing 911 audio calls recordings using IBM, Google speech to text APIs and open source automatic speech recogniser CMUSphinx. These transcripts are manually labeled and collated in a structured format in a CSV file format creating three training datasets. The test dataset is created by manually transcribing and labelling the first utterance of the caller from each call and saved in a CSV file format.

1.5 Language model

N-grams are used in language modelling. It is used to represent text as a set of characters. N-grams are represented as a unigram, bigram, and trigram. A single word representation is a unigram while a pair of words is a bigram. Text features are extracted from n-grams in all text classification task.

1.6 Feature Extraction

Raw text cannot be fed to a classifier since it expects normalized numerical feature vectors. Feature extraction is the process of transforming text in numerical features that can be used by the classifiers (Pedregosa et al.; 2011). A common representation of words in numerical features are; local representation and distributed representation of words. In local representation, count by the occurrence of words in a document called bag-of-words, and weighting of words according to the importance of a word in a document called tf-idf weightings are used. In distributed representation of words, words are converted to factorial vectors which preserve the context of the word in a document.

1.6.1 Bag-of-words

Bag-of-words model is a simple unigram model which models each document by counting a number of times each word occurs in a document from the vocabulary of unique words created from all of the document (Pedregosa et al.; 2011). The order of the words in a sentence is not considered in bag-of-words model thus the meaning of a sentence is lost. Bag-of-words is a vector space model which is widely used in information retrieval to find similarity between documents. Count features are also represented as a binary occurrence marker which assigns 1 if a word is present in a test document which is also present in the vocabulary and 0 if the word is absent in the vocabulary. Binary features representation is more suitable for short documents like speech utterances (Pedregosa et al.; 2011).

1.6.2 Tf-idf term weighting

Tf-idf transform is used to re-weight the count features of words into floating point values which are fed to a classifier (Pedregosa et al.; 2011). Term frequency is the number of

times a word occurs in a document, while inverse document frequency is the log of a total number of documents N divided by the number of documents containing the term t. Tf-idf is computed as:

$$tfidf(t,d) = tf(t,d) \times idf(t)$$

By tf-idf weighting, words which are rare in a document has high weights while words that often occur in a document get low weight. This weighting technique is extremely useful in document classification (Pedregosa et al.; 2011). The length of the document is normalized in tf-idf vectorization. The normalisation of the transcripts is useful in speech utterance classification since the length of the utterance varies in length.

1.6.3 Word2vec word embeddings

Bag-of-words (Count vectorizer) and tf-idf vectorizer rely on the word count in a sentence to vectorization of words. These methods do not save any syntactical information of words. Word embedding is another method of extracting features out of text which is used as input to the classifiers. However, word embedding preserves syntactical information of a sentence (Mikolov, Sutskever, Chen, Corrado and Dean; 2013). Distributed representation of words by word2vec uses one hot vector window in the continuous bag-of-words model and skip-gram model to obtain feature vectors with semantic word relationship. A corpus of documents is trained on these model which in turn return vectors which contain syntactic and semantic word relationship in vector form. Mathematical operations can be performed on these word vectors to obtain word vectors close to those vectors. For example, addition of $vec("Germany") + vec("capital")$ gives a vector which is close to $vec("Berlin")$ (Mikolov, Sutskever, Chen, Corrado and Dean; 2013). Distributed representation of words of speech utterance can produce better classification results since it considers the semantic similarity of words.

1.7 Classifiers

The speech utterance classification task is a multi-class classification problem. Given a set of training examples $(x_1, y_1), ..., (x_i, y_i)$ belonging to space X and Y of all possible instances x_i and class labels y_i. Each instance x_i is assigned class label y_i. With the given set of training data, the goal of a learning algorithm is to use the training data to learn $p(x|y)$ and accurately predict the call type or class label for a new utterance x. Such type of learning algorithm is called classifier (Schapire et al.; 2005).

Due to small size of training dataset Artificial Neural Networks algorithms (ANN) are not implemented as it needs large training dataset. For this reason, traditional learning algorithms are used for classification task which has been successfully implemented in speech utterance classification for call routing.

The classification models used are the generative and discriminative model for training and testing. Generative model is a probabilistic classifier which estimates joint probability $p(x, y)$, for input x and class label y and makes prediction using Bayes rule to calculate $p(y|x)$ (Ng and Jordan; 2002). Whereas discriminative model uses conditional probability distribution $p(y|x)$ by learning boundaries between classes and map document to the class. The discriminative model performs better in classification task where there is no need of joint probability (Ng and Jordan; 2002). The generative model has higher asymptotic error compared to discriminative classifiers when training example increases

(Ng and Jordan; 2002). The research uses five widely used machine learning text classifiers, Support Vector Machine (SVM), Logistic regression, Bernoulli Naive Bayes, and AdaBoost. SVM and logistic regression are binary classifiers which are used in the one-vs-all strategy. Common multi-class classification techniques used are 1. Hierarchical classification, 2. Extension from binary, 3. Decomposing into binary classification (Aly; 2005). In decomposing to binary classification, the multi-class classification problem is solved by decomposing the multiple class into several binary classification tasks and uses a binary classifier for classification (Aly; 2005). One-versus-all (OVA), all-versus-all (AVA), Error-correcting output-coding(ECOC) and Generalised coding are some of the strategy used in decomposing to binary classification. The research uses One-versus-all (OVA) strategy with SVM and logistic regression for classification since these are most widely used binary classifiers (Aly; 2005).

1.7.1 One-vs-all classification

One-vs-all classification strategy is used in multi-class classification. In binary classification, a linear classifier like SVM separates two classes as positive or negative using a hyperplane. For multi-class classification, the one-vs-all strategy is used. Given a training set of three classes y_1, y_2, y_3, the one-vs-all turn multi-class classification into three separate binary class classification problem. In this strategy, a classifier learns to fit a single class as a positive class while the rest two classes as a negative class and treats it as a binary classification problem. This is repeated with other two classes. A label y of the test set is decided by one of the three classifiers confidence. SVM and logistics regression is used in one-vs-all strategy in the research.

1.7.2 Naive Bayes

Naive Bayes is generative probabilistic classifier used in many Natural Language Processing task. It naively assumes independence of features to assign the probability of class to a test document. Naive Bayes uses Bayes rule to find the likelihood of a class given the document

$$P(c|d) \propto P(c) \prod_{1<k<n_d} P(w_k|C)$$

$P(w_k|c)$ is the conditional probability of k words occurring in a document of a class c (Christopher D. Manning; 2009). The prior probability of a document $P(c)$ and $P(w_k|c)$ is estimated from labeled training data. n_d is the number of words in the vocabulary. The goal of the Naive Bayes classifier is to find the probability of a class for a document. Class with highest posterior probability for a document d_i is selected (Sarikaya et al.; 2005). Naive Bayes is simple and fast to implement. Bernoulli Naive Bayes is a variant of Naive Bayes classifier. Bernoulli Naive Bayes assumes features as binary valued variables (Pedregosa et al.; 2011). Binary features are set by count vectorizer (bag-of-words). Since Bernoulli Naive Bayes performs better on short text, it is chosen over multinomial Naive Bayes which is another variant of Naive Bayes (Pedregosa et al.; 2011).

1.7.3 Support Vector Machines

Support Vector Machines is large-margin discriminative classifier. The goal of SVM is to find decision boundary $(\vec{w} \cdot \vec{x}) + b$ between two classes with maximum margin to the

closest point in the sample with decision function

$$f(\vec{x}) = sgn((\vec{w} \cdot \vec{x}) + b))$$

(Sarikaya et al.; 2005). Given sample set X and label set Y for sample $(x_1, y_1), ..., (x_i, y_i)$ where $x_i \in X$ and $y_i \in Y$, a hyper-plane $(\vec{w} \cdot \vec{x}) + b$ separates the classes as positive or negative class for $(\vec{w} \cdot \vec{x}) + b > 0$ if $y_i = 1$ and $(\vec{w} \cdot \vec{x}) + b < 0$ if $y_i = -1$. Minimising the weight vector $\vec{w}$ maximises the margin that split the two classes. The weight vectors is minimised using Langrange multipliers λ_i. There are different choice of kernels in SVM which decides the classification power of SVM (Sarikaya et al.; 2005). SVM is an effective classifier for high dimensional space and has shown better performance compared to other classifiers in text classification (Pedregosa et al.; 2011; Joachims; 1998). However, the performance of SVM is poor when the number of the features is much higher than number of sample (Pedregosa et al.; 2011). Linear, polynomial and radial basis functions are some of the commonly used kernels for classification task (Sarikaya et al.; 2005). Linear kernel is implemented in one-vs-rest strategy in the research.

1.7.4 Logistic Regression

Logistic regression is another discriminative classifier. It is one of the most widely used learning algorithms in text classification. A hypothesis function is representated as $h_\theta(x) = g(\theta_x^T)$ and $g(z)$ is a sigmoid function. The hypothesis predicts a class $y = 1$ if $\theta_x^T \geq 0$ and $y = 0$ if $\theta_x^T < 0$. The logistic regression is used in one-vs-all strategy where the multi-class problem for three classes is treated as three binary classification problem.

1.7.5 AdaBoost

Boosting algorithms has been successfully used for text categorization problem (Schapire et al.; 2005). Boosting technique is used to improve the accuracy of a learning model (Schapire; 1999). It produces new accurate prediction rules by combining weak rules (Sarikaya et al.; 2005). Freund and Schapire (1995) AdaBoost algorithm is widely used boosting algorithm since it is fast, easy to program, requires no parameters tuning and does not suffer overfitting (Freund et al.; 1999). AdaBoost produces a highly accurate model by combining weak rules set by a weak learning algorithm. The algorithm minimizes training error rate by training sequentially on weak rules (Schapire et al.; 2005; Sarikaya et al.; 2005).

1.8 Automatic Speech recognition

Automatic Speech Recognition (ASR) is one of the main components for speech utterance classification. The task of the ASR is to map acoustic signals to a string of words (Jurafsky; 2000). ASR has two major component acoustic model and language model (Patil et al.; 2016). Language model predicts the probability of a given sentence while acoustic model maps the pronunciation of spoken word (Patil et al.; 2016). These models are trained on large vocabulary to obtain low word error rate of transcripts. Commercial speech recognition systems are the best speech recognition system as they give lowest word error rate. However, the word error rate increases if the audio quality is poor and there are multiple speakers in the audio. The research uses commercial ASR from Google, IBM and open source CMUSphinx [1].

[1] `https://cmusphinx.github.io/`

The further section gives details of the previous work on speech utterance classification followed by the methodology used to answer the research question. The classifiers are evaluated on accuracy and f-score on three training datasets created using IBM, Google, and CMUSphinx ASR. The classifiers are also evaluated for accuracy, and f-score on three feature extraction model on training dataset created using Google ASR generated.

2 Related Work

2.1 Speech utterance classification

Speech utterance classification has many practical applications. One such application is call routing, where the callers' utterance is used for routing the call within a call centre to desired destination (Chu-Carroll and Carpenter; 1999). Call routing is similar to document routing and topic identification task (Garfield and Wermter; 2006). Main components of utterance classification are manually transcribed and labelled training data, statistical machine learning classifier and accurate recognition of utterance by the Automatic Speech Recognizer (ASR) (Wang et al.; 2006). Vector based classification and probabilistic based classifiers had been used by researchers for call classification task (Zitouni et al.; 2001; Chu-Carroll and Carpenter; 1999). Chu-Carroll and Carpenter (1999) used vector-based routing module and disambiguation module for routing the calls to the desired department. Vector-based routing module used cosine distance between the query of the caller and the trained corpus to determine the destination of a call (Kuo and Lee; 2003). The destination and the callers' utterance were represented as vectors in n-dimensional space (Chu-Carroll and Carpenter; 1999). The destination of the callers' utterance (query vector) was selected by computing cosine distance between the callers' query vector and each destination vector. Query vector with high cosine similarity with a destination vector was chosen as callers' destination. However, when the queries were closer to two destinations, disambiguation module was invoked. Disambiguation module solicits more information from the caller to refine the request vector. The new vectors added were used to determine the destination of the call. Using n-gram and bag of words model for information retrieval, the researchers had achieved 94% accuracy on error free transcripts; the accuracy dropped by 9% when the output of the ASR was used with 23% word error rate (Chu-Carroll and Carpenter; 1999). The researchers used only vector-based information retrieval techniques for routing the calls, and no discriminative or generative classifiers were used for routing the calls. The advantage of the vector-based information retrieval classification is that it is a single pass system where the speech recognition and classification is concurrent which responds quickly to users' utterance whereas classification using classifiers is a two-pass system where the calls are classified after speech recognition (Chelba et al.; 2003).

Chelba et al. (2003) compared one-pass system using n-gram classifier with a two-pass system using Naive Bayes (NB) and maximum entropy (ME) classifiers. The comparison showed an improvement in the accuracy of the classifier when 2-gram features were used over 1-gram features on NB and ME classifier. The discriminative technique used for training classifier had a modest improvement in the performance of the classifiers when more features sets were used. The discriminative training proved more robust to speech recognition errors Chelba et al. (2003). The results showed an improvement in the accuracy by 55% when the classifiers were trained and tested on error free transcripts. It also surpassed the performance of the one-pass system. However, in real time applica-

tion, accurate transcriptions are not always available for classification since the calls are transcribed by the speech recogniser. Erroneous transcripts from the speech recogniser can lead to the wrong classification of calls which may result in a routing of calls to a different destination other than the desired destination. Tyson and Matula (2004) proposed re-weighting of query vectors from the confidence score of ASR model to improve the accuracy of Latent Semantic Indexing (LSI) Classifier. Their method involved training of commercial ASR engine language model and LSI classifier on same utterance transcripts. Taking the geometric mean of the confidence score of the words from the ASR model and improvement of 1% was achieved over the standard LSI classifier and achieved a significant reduction in classification error rate (Tyson and Matula; 2004).
Kuo and Lee (2003) achieved 10-30% reduction in classification error rate when discriminative training was used on classifiers. In addition to discriminative training Zitouni et al. (2001) used boosting and relevance-feedback to improve the accuracy of the classifiers. Using the above three techniques on vector-based and probabilistic classifiers, an improvement of 15-45% in the accuracy of the classifier was achieved (Zitouni et al.; 2001). An improvement in the accuracy of 10% was obtained when a combination of vector-based probabilistic classifiers was used. A 4% improvement in accuracy was achieved by linear interpolation of two classifiers while there was no improvement in the accuracy when linear interpolation of the classifier was used on constrained minimisation technique, a 7% improvement in the accuracy was observed on using boosting method on linear interpolation of classifiers (Zitouni et al.; 2001). All classifiers used by the researchers above used supervised learning method which needs a large amount of labelled dataset to train the classifier. Sarikaya et al. (2005) used an unsupervised method with a large amount of unlabelled data and limited labelled data to improve classifiers performance. This dataset was used for training SVM, NB and MaxEnt classifiers. A combination of SVM and MaxEnt classifier improved the accuracy of the classifier by 2% (Sarikaya et al.; 2005). Niyogi et al. (2000) used a combination of three classifiers which resulted in improvement in accuracy compared to the utilisation of a single classifier.
A comparison with vector-based model and other traditional classifiers was made with simple recurrent neural networks (SRN) and Finite State Transducer (FST). SRN achieved better accuracy compared to conventional classifiers (Garfield and Wermter; 2006). FST produced better performance on call classes which were problematic to SRN (Garfield and Wermter; 2006). However, it required manual symbolic tagging of each utterance of the call to create a regular expression which is time-consuming. Sarikaya et al. (2011) compared Deep Belief Nets (DBN) initialised feed-forward neural network with traditional classifiers. Using DBN with multiple features to feed forward neural networks optimized the performance of neural networks and made it less prone to overfitting. DBN achieved better accuracy over MaxEnt and boosting classifiers and an equivalent accuracy to SVM (Sarikaya et al.; 2011). All of the above the classifiers used produced the best performance when they were trained and tested on correct transcripts which were manually created and annotated. During the deployment of the system using manual transcriptions is less efficient as the system needs to be regularly updated to add features from the users' utterance. Moreover, correct call classification is often affected by the incorrect recognition of users utterance by ASR output due to poor audio quality. Faruquie et al. (2007) used statistical machine translation (SMT) to reduce word error rate (WER) of ASR transcribed the text. IBM translation model 2 was trained on a parallel corpus of N-best sentences of ASR text and manually transcribed text. The machine translation correct sentence output showed 8% improvement in the classification accuracy compared to N-

best sentence classification output Faruquie et al. (2007). Manual transcription remains a major bottleneck for training and tuning of classifiers (Wang et al.; 2006). Many techniques were proposed to eliminate the need for manual transcription for model training. Huang and Cox (2004) used phone[2] sequence generated by phone recogniser as feature for the classifiers. This model achieved accuracy equivalent to word-based model. However, this model required multiple n-gram models and acoustic models to generate phone sequences of an utterance which increased operational cost (Wang et al.; 2006). Another method implemented by Wang et al. (2006) used unsupervised language model (LM) adaptation technique to reduce training set recognition error rate by 30% (Wang et al.; 2006). The classification model was trained by using audio files of speech utterances and their classes. The language model was trained on speech utterances of audio files and interpolated with the original language model of the ASR to perform recognition of audio again. However, the recognition error from the new model was fed to the old model which made less improvement in the recognition error. The classification error in the test set was equivalent to classification error observed by manual transcripts trained model and a significant improvement in the classification accuracy.
Horvitz et al. (2007) invented prioritization system for prioritizing voice mail and real-time telephone based level of urgency of calls from speech pattern, prosodic features, syllabic rate, pause structure and metadata such as callers id and call duration. The prioritizing system analyses real-time calls and based on the features determined from the calls, a call was either forwarded to the user or directed to voicemail. A voicemail was analyzed by the classifier, and a graphical display shows the urgency of the call Horvitz et al. (2007). The classifier used were a Gaussian process and Bayesian network for classification of calls. Many natural language call routing used classification based on vector space model to determine a calls cosine length from the destination. Based on the cosine distance the calls destination is selected. Vector space model, discriminative and generative models were used to improve the accuracy of the classifiers Chu-Carroll and Carpenter (1999); Sarikaya et al. (2005). An ensemble of classifiers showed better accuracy than a single classifier. Discriminative training and boosting improved accuracy of the classifiers Zitouni et al. (2001); Kuo and Lee (2003). The research used traditional classifiers in call routing task since they performed well in speech utterance classification. The features used from the utterances in call classification task were vectors generated by bag of words and tf-idf model. Garfield and Wermter (2006) highlights the use of feature representation of words in utterance which provides the meaning of the word. Latent Sematic Analysis (LSA) and word2vec word embedding consider the semantic similarity of words (Landauer et al.; 1998; Mikolov, Sutskever, Chen, Corrado and Dean; 2013). The research uses word2vec word embeddings to capture semantic similarity of words in the utterance of the call.

[2]phone - classes of sound

3 Methodology

Following steps were followed to benchmark the classifiers : 1. Data Collection. 2. Pre-processing. 3. Feature Extraction 4. Modelling. Fig. 1 shows the step that were used to answer the research question.

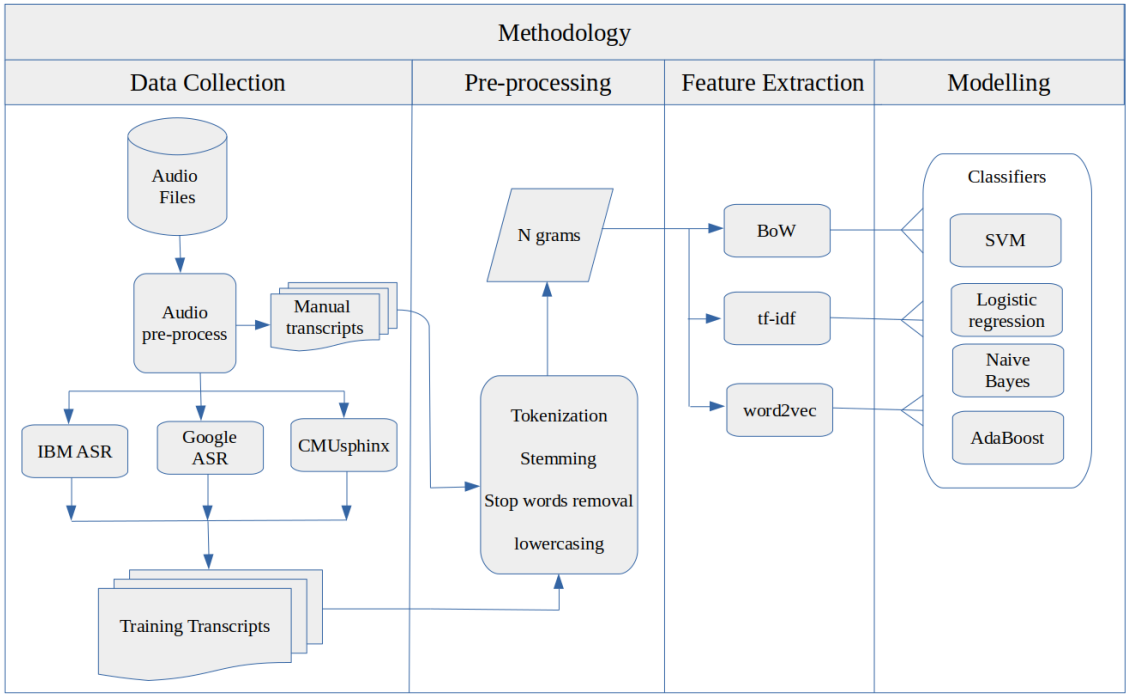

Figure 1: Methodology used for research consist of four phases: Data collection, Pre-processing, Feature Extraction and Modelling

1. **Data Collection** The transcripts used for the training of classification model were extracted from audio recordings of 911 calls. Audio calls were chosen as a data source to obtain meaningful features in the transcripts used for training the classifiers. However, there were many problems with the sound quality of the recordings.

   (a) **Audio files** The audio data used for the research publicly available archived recordings of 911 calls of the emergency incident reported to a dispatcher. These recordings are from the year 2006 to 2014 and are in public domain with no copyright limitations [3]. The call recordings vary in length from 2-25 minutes and contain a conversation between a caller and a call taker. All calls are annotated giving details of the emergency reported including the location of the caller. The call recordings are from the USA containing both critical and noncritical emergencies. The recorded audio calls had many issues which make it a less ideal source for linguistics analysis (Burns and Moffitt; 2014). The problems with the audio calls were 1. The recorded sound quality is poor due to noise in the background of the caller. 2. There is dead air time in the calls when a caller is put on hold by the dispatcher to coordinate with emergency personnel. 3. Multiple speakers talk at the same time. 4 Certain

[3] `http://bit.ly/2oQLBqR`

sounds made by the caller due distress add noise in the audio. Due to the above problems, the sound quality of calls hugely degrades which it makes difficult to obtain correct transcripts.

(b) **Audio pre-processing** Based on the audio quality, only good quality audio calls were used for transcription to get maximum correct words from ASR transcripts. Even the good quality audio contained ambient noise. A total of 152 good quality audio calls were used for transcription. The calls were then manually analyzed and labelled as high, medium and low based on the annotation of the audio recordings. For example, calls reporting a life-threatening incidence, serious crime, reports of fire were labelled high. Calls reporting misdemeanour in progress, non-life threatening injuries, lost person was labelled medium, while non-serious calls like a person reporting a wrong order of food from McDonald's and other non-serious calls were categorized as a low priority call. After selection and classification of audio calls based on quality and categories, a total of 152 calls were utilised for transcription and creation of labelled dataset. The categorized calls contained 82 calls as high, 48 calls as medium and 22 calls as low priority. These audios were all trimmed to 1 minute since the initial utterance of the callers had the maximum number of content words that were used as features for training a model. The trimmed audio files were then converted into Free Lossless Audio Codec (FLAC) [4] format which was used for the speech to text APIs for transcription.

(c) **Train and Test dataset** The processed audio calls were transcribed using IBM and Google Speech to text APIs and open source CMUSphinx. IBM and Google ASR uses deep neural networks for speech recognition (Saon et al.; 2015; Kim et al.; 2017). These speech recognition models produce accurate transcript compared to other speech recognition models. CMUSphinx is an open source Automatic Speech Recognizer which can be trained on different acoustic and language models. It was pre-trained on acoustic and language models. Randomly selected 80 audio calls were transcribed manually to create test dataset. These transcriptions were the first utterance of the caller in the calls. The transcripts of the audio were collated in a CSV formatted file. Three train datasets and one test dataset were created with attributes incident ID, text, and class. A numerical representation for classes was given for incident ID attribute as '$0'$ for High, '$1'$ for Medium and '$2'$ for Low. The training set contained unbalanced class labels with 82 transcripts labeled high, 48 labeled Medium and 22 labeled Low. Test dataset contained unbalanced classes with 26 utterances labeled High, 36 labeled Medium and 17 labeled low. The train and test dataset were pre-processed in the next step. Class distribution for train and test dataset is shown in Fig. 2.

[4] `https://xiph.org/flac/`

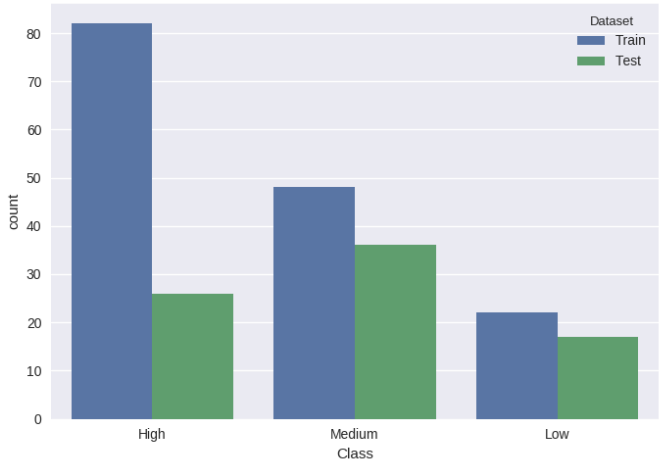

Figure 2: Train and test dataset class distribution

2. **Pre-processing** Pre-preprocessing is one of the most important components of text classification (Uysal and Gunal; 2014). The transcript contained words which did not contribute as features for classification. These words were removed in preprocessing step. Following are the pre-processing steps used for cleaning the transcripts.

   (a) Regular expression: Transcripts from the ASR output contained alphanumeric characters. Numbers in the text do not contribute to the features of the text, so these were removed using regular expression function in python. Text with only words was extracted using the regular expression.

   (b) Lower case: The extracted words were normalized by lower casing all words so that the difference between an uppercase word and lower case word was ignored (Bird et al.; 2009).

   (c) Stemming: The normalized text was stemmed to their root word. For example, 'calling', 'called' are all stemmed to base word 'call'.

   (d) Stop words removal: Stop words like 'and', 'the' etc. are frequently used in common vocabulary. These words do not contribute to features of the text and are not important in the classification task. Hence they were removed from the text. This also improves the performance of classifier by reducing the text data (Kannan and Gurusamy; 2014).

   (e) Tokenization: The processed text was then converted into individual tokens which are nothing but a sequence of characters. White space and punctuation were used as the token separator. A list of token also called as unigrams were formed. These unigrams were used as features by a learning algorithm. The unigrams were converted into numerical features in feature extraction process.

3. **Feature Extraction** Machine learning algorithm expects numerical features of fixed length as input rather than raw text data (Pedregosa et al.; 2011). Vectorization is a common process of converting words into the numerical feature vectors. Bag-of-words count features vectors, tf-idf term weighting, and word2vec word embeddings were used for converting text into numerical feature vectors features.

(a) Bag-of-words features: Bag-of-words features are most commonly used features in text classification. A list of vocabulary from the dataset was created. The vocabulary is nothing but a list of unique words from the dataset. The bag-of-words model counts the number of times each word from a transcript occurred in the vocabulary. A two-dimensional matrix with one column per vocabulary word and one row per transcript was created. These numerical count features were fed into the learning algorithm. Bag-of-words is a simple count based model which performs well on classification task. However, this model does not consider the structure of the sentence when converting text into numerical features. Binary occurrence representation as features as feature vector was used. In binary occurrence, if a word in a transcript occurs in vocabulary it was set as 1 while the absence was set as 0. Binary occurrence maker often gives better features for classification task and works well for short documents classification (Pedregosa et al.; 2011).

(b) tf-idf: Term frequency inverse document frequency weighting was used to re-weight the count features of words. The count features were converted into floating point values which were fed into classifiers (Pedregosa et al.; 2011). Common words occurring in the transcript were weighted low while rare words were weighted high so that rare words were considered as an important feature in a transcript. Short transcripts may contain noisy tf-idf values, hence count features from bag-of-words vectors are preferred over tf-idf values (Pedregosa et al.; 2011).

(c) Word2vec: Bag-of-words and tf-idf vectorization do not consider the structure of the sentence to create features vectors. Moreover, it does not preserve the syntax of the word in feature vectors. Word embeddings from word2vec preserve semantic word relationship in numerical features (Mikolov, Sutskever, Chen, Corrado and Dean; 2013). The semantic similarity of words helps in improving classification accuracy (Mikolov, Sutskever, Chen, Corrado and Dean; 2013). Preserving semantic word relationship is important in the task of speech utterance classification which will provide the meaning of the speech. Average feature vectors of the transcript were created from pre-trained word embeddings. For comparison of the accuracy of classifiers on three training datasets, numerical count features from each ASR generated transcripts were fed to each classifier to compare the accuracy of the classifiers on ASR generated transcripts. Feature vectors from bag-of-words, tf-idf, word2vec of Google ASR generated dataset were fed into each classifier to test whether word2vec features vectors improve the accuracy of the classifiers or not.

4. **Modelling** The classification model is an important part of speech utterance classification. The task of the classifier is to find the joint probability or conditional probability distribution to determine the class of the utterance. The classification models used were the generative models, discriminative models and boosting algorithm. Support Vector Machines, Bernoulli Naive Bayes, Logistic Regression and AdaBoost were used as learning algorithms. These are the most commonly used and most successful classification algorithm for speech utterance classification (Sarikaya et al.; 2005). Support vector machines and logistic regression were chosen as classifiers since they are widely used in text classification. SVM requires less parameter tuning and performs well in the classification task. Bernoulli Naive Bayes was util-

ized since it performs better on short text classification. AdaBoost was used since it had achieved better performance in call classification task (Schapire et al.; 2005).

4 Implementation

4.1 Python Modules

The pre-processing, feature extraction, training, and testing of classifiers werer implemented in Python using different Python modules.

- The module used for transcribing audio in python was 'SpeechRecognition' developed by (Zhang; n.d.). IBM and Google speech-to-text APIs used credentials which need to communicate with the servers, while the CMUSphinx is an offline speech recognition system pre-trained on acoustic and language model.
- For importing the datasets into a data-frame 'Pandas' module was used. 'Pandas' is easy to use open source library which handles structured data efficiently.
- Python module used for pre-processing transcripts were 'nltk' and 'regular expression' (Bird et al.; 2009). 'nltk' is an open source library specifically built for text analysis.
- The Python modules used for text feature extraction were 'CountVectorizer' and 'TfidfVectorizer' (Pedregosa et al.; 2011). For word embeddings, 'gensim' module was used.
- For classifiers, 'sklearn' ensemble for 'AdaBoost classifier', sklearn 'linear_model' for logistic regression, sklearn 'LinearSVC' for support vector machines and sklearn 'naive_bayes' for Bernoulli Naive Bayes were used (Buitinck et al.; 2013). Matplotlib and seaborn modules were used to plot graphs.

4.2 Pre-prossessing

Pre-processing is an important step in speech utterance classification. Using 'Pandas', dataset was imported into a dataframe and the text was pre-processed. Alphanumeric characters were removed using regular expression function in Python. The words in the dataset were lowercased using inbuilt Python function. The lowercase words were stemmed using 'snowball' stemming algorithm in nltk library (Porter; 2001). Snowball stemming algorithm is an improved algorithm of porter stemming (Porter; 2001). Stop words like 'and', 'the', 'like' etc. were removed from stemmed text using 'nltk' module (Bird et al.; 2009). A list of unigrams was created in this process. These unigrams were converted to numerical feature vectors in the feature extraction module.

4.3 Feature Extraction

For feature extraction sklearn CountVectorizer and tfidfVectorizer functions were used. Countvectorizer converted train and test dataset into a matrix of token counts (Buitinck et al.; 2013). A vocabulary size of 1000 was set for the train and test dataset. These vocabularies were a list of unique words in the dataset also called features of the dataset

which determines the class of the transcripts. Sklearn Fit_transform function was used which assigned a unique integer index to each unigram in the vocabulary of the training dataset. This unique integer corresponds to a column in the matrix (Pedregosa et al.; 2011). Transform function counts the number of times a word occur in the vocabulary for each instance in the training dataset. A sparse matrix of instances as transcripts and binary count features of the words was assigned to each word in the vocabulary. A sparse matrix of numpy array of train and test was created which was into fed to the classifier. Sklearn TfidfVectorizer was used to convert the train and test dataset into td-idf factorial features which were weights of each term in the transcript. A vocabulary list of 1000 was set. Term vectors were normalized with $l2$ norm which normalizes the weights between $0-1$. Fit_transform was used which computes the weights of each term in the vocabulary. The floating point numerical values were stored in a sparse matrix with vocabulary list as columns and weight of each term of transcripts as rows. This sparse matrix was then used for training a learning algorithm.
Distributed representation of words was obtained by taking average vectors from pre-trained vectors which were trained on Google News dataset which containing 300-dimensional vector for 3 million words (Mikolov, Chen, Corrado and Dean; 2013). The features of pre-trained vectors were added to the transcript words if they are present in the pre-trained model's vocabulary and an average of these new feature vectors was taken by dividing the feature vectors by the total number of words in the transcript. Stop words were not removed to obtain average feature vectors since they are important to preserve the syntax of the sentence. The average vectors were created for train and test dataset. The average feature vectors were fed to the learning algorithm.

4.4 Classifiers

The features extracted from each training dataset was passed in the fit method for each classifier. An estimator instance 'clf' was called which learns from the training examples. The test dataset was passed in predict function on which an instance estimator 'clf was implemented to predict the classes of the unseen test examples.

Different feature vectors from dataset created using Google ASR were passed in fit method for each classifier. Estimator instance 'clf' was called which learnt from the different feature vectors. 'clf' was implemented on predict method to predict the classes of test dataset. The fit function and predict functions were used to learn from the training examples and predict unseen data.

The classifiers trained for comparison were Logistic Regression, Support Vector Machines, Bernoulli Naive Bayes, and AdaBoost.

1. Logistic Regression: Logistic regression was used in one-vs-all (OVA) multi-class settings. Since the training datasets was small in size, 'liblinear' solver was used. 'liblinear' is an algorithm used to solve optimization problem and is used with small datasets (Buitinck et al.; 2013). $l2$ regularization was used since it reduces model overfitting caused by high variance in the dataset. Other parameters were set to default in sklearn LogiticRegression function.

2. Support Vector Machines: SVM classifier was trained using the sklearn LinearSVC function. LinearSVC is a linear support Vector Classifier. SVM was used in the one-vs-all setting for multiclass classification. $l2$ regularization was used to prevent

overfitting of the data. Default linear kernel was used to linearly separate classes in OVA settings. Other parameters were set to default in LinearSVC function.

3. Bernoulli Naive Bayes: sklearn BernoulliNB function was used to train Bernoulli Naive Bayes classifier. Bernoulli Naive Bayes uses binary occurrence marker as feature vectors from countvectorizer. Laplace smoothing parameter was used, which captured important patterns in the text of the training and test dataset. Other parameters were set to default values.

4. AdaBoost: The AdaBoost classifier was trained using sklearn AdaBoostClassifier function. Base classifier used for AdaBoost was Decision tree. SAMME.R algorithm was used since it gives lower test error with fewer iterations (Buitinck et al.; 2013). Other parameters in the function were set to default.

   The result of prediction was saved in 'predict' variable. The actual classes of test dataset and prediction were then used to find accuracy and f-score of each classifier. The accuracy of the classifiers on each training dataset and different feature vectors was compared by plotting a bar graph using matplotlib module.

5 Evaluation

Evaluation measures play a major role in constructing a classification model (Huang; 2006). To answer the research question, classifiers were trained on ASR generated transcripts and tested on transcripts of initial utterances of calls. The classifiers were evaluated on accuracy, F-score, precision, and recall. These are the standard evaluation metrics in measuring classifiers performance (Garfield and Wermter; 2006).

5.1 Evaluation metrics

Accuracy measures how often classifier is correct in call classification. It is the ratio of correctly classified transcripts to the total classified transcripts. Accuracy is often affected if the count of one class is less than other classes (Garfield and Wermter; 2006). Since the dataset has unbalanced classes, using accuracy as the only evaluation metric will produce false results. F-score is a more appropriate measure for evaluating classifiers on unbalanced dataset since it takes both precision and recall into consideration. Moreover, the classifier should have low misclassification rate which is given by '$1 - accuracy$', and less False negatives since a call with actual high priority should not be classified as low priority call i.e. it should have high recall value.

Precision and recall are calculated using the equation:

$$Precision = \frac{\text{Number of relevant transcripts retrieved}}{\text{Total Number of relevant transcripts in the dataset}}$$

$$Recall = \frac{\text{Number of relevant transcripts retrieved}}{\text{Total Number of transcripts retrieved}}$$

Precision and recall values are measured from 0 to 1 where 1 is best value and 0 is worst value (Pedregosa et al.; 2011). F-score combines precision and recall and its values lie between precision and recall values. F-score does not favour either precision or recall and consider both values for computing f-score (Garfield and Wermter; 2006). F-score is given as

$$Fscore = \frac{2 \times \text{precision} \times \text{recall}}{\text{precision} + \text{recall}}$$

The value of f-score is measured between 0 and 1.

5.2 Evaluation of transcripts

In real time application, the classifier will need accurately transcribed text from ASR for training to create relevant features for the corresponding classes. ASR output was evaluated on word error rate (WER). Lower WER of a transcript means a better transcript. The WER was found by comparing ASR transcript with reference transcript. Reference transcripts are often manually transcribed since it has to have true words. WER is computed as:

$$WER = \frac{\text{Number of Substitutions + Number of Deletions + Number of Insertions}}{\text{Number of words in the reference}}$$

An audio file was manually transcribed and used as reference transcript to find word error rate of the ASR generated transcript of the same audio file. Using a Python application created by belambert (n.d.), the WER for ASR generated transcript was computed, and a comparison of ASR output on WER is shown in Fig. 3. From the Fig. 3 it can be seen that CMUSphinx ASR transcript has highest Word Error Rate of above 40% compared to other ASR generated transcripts. Background noise and multiple speakers can be the reason high word error rate in CMUSphinx ASR transcript. The transcript produced by Google ASR has minimum WER about 19% and has less incorrect words in the transcript compared to IBM and CMUSphinx generated transcripts. WER is an important evaluation metric for ASR performance. The successful classification of a call will depend on the accurate transcripts from ASR. Transcripts with low WER rate will be correctly classified as compared to transcripts with high WER. Training dataset created from Google generated transcript had more accurate features for classes than training dataset created using IBM and CMUSphinx generated.

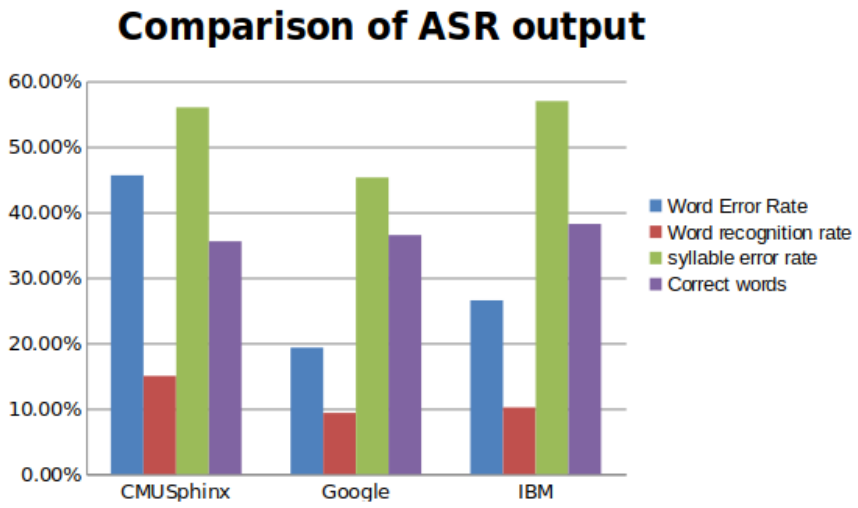

Figure 3: Comparison of ASR transcripts

5.3 Analysis of dataset

The research analyzed 60 transcribed calls to examine the callers' dialogue behavior. In the majority of calls, the first utterance of caller contained between 25-35 words. The

longest callers' first utterance was about 45 words in length. While the length of the first utterance varied it only contained few content words which are the keywords extracted from the dataset relevant to the class of the call. Most of the Calls contained less than five content words in the first utterance of the caller. The distribution of word length of the first utterance and content words is shown in the fig 4

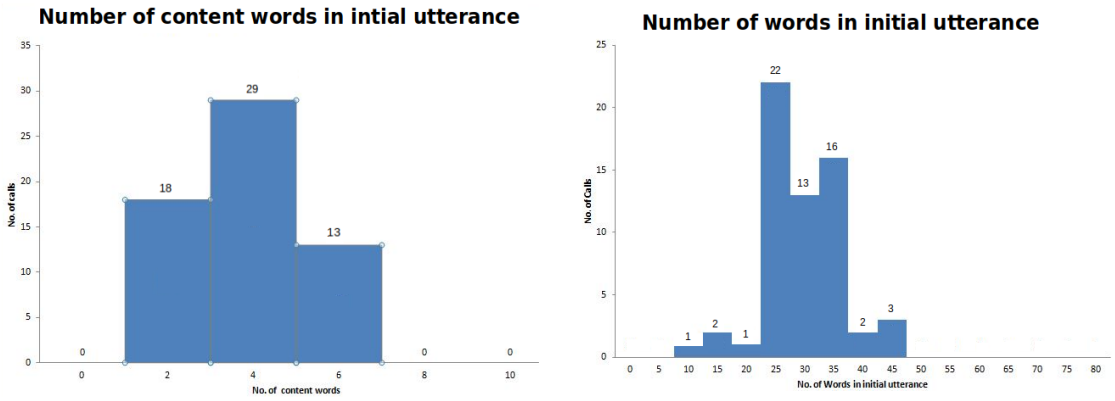

Figure 4: Number of words and content words in initial utterance

5.4 Evaluation of classifiers on different ASR generated transcripts

The classifiers were trained on training dataset created using transcripts from IBM, Google, and CMUSphinx ASR and tested on manually transcribed first utterances of 80 audio calls. The Binary occurrence of feature vectors from the bag-of-words model was used in training and testing the classifiers. The classifiers accuracy on three datasets is shown in table 1.

	IBM	CMUSphinx	Google
Logistic Regression	48.45%	28.76%	69.23%
Bernoulli Naive Bayes	50.01%	47.23%	71.32%
AdaBoost	60.53%	53.84%	68.23%
SVM	65.38%	38.46%	73.07%

Table 1: Accuracy of classifiers on ASR generated datasets

The overall performance of all classifiers on Google ASR generated dataset was better compared to accuracy on other two datasets. This was an expected result since Google ASR generated transcript contained less incorrect words as it had low WER below 20%. While the accuracy of classifiers on CMUSphinx generated dataset was lowest since the dataset had high WER of above 45%. Classifiers accuracy on IBM ASR generated dataset was better than accuracy on CMUSphinx dataset. SVM is showed better results on both IBM and Google dataset and was the best classifier compared to other classifiers. While SVM and logistic regression performed poorly on CMUSphinx dataset. These are the best classifiers on Google dataset with the accuracy of 73% and 69% respectively. Using $l1$ regularization in SVM to prevents overfitting of data which resulted in better accuracy on both Google and IBM dataset. SVM performed poorly on CMUSphnix since it considers

the importance of particular features of text which are missing in the dataset due to high WER (Dix; 2009). Bernoulli Naive Bayes used binary occurrences of feature vectors as input and had shown better accuracy compared to logistic regression and AdaBoost on Google dataset. AdaBoost which used decision tree as base classifier had the least accuracy result on Google dataset. However, boosting technique worked on CMUSphinx created dataset which showed better result compared to other classifiers. Fig 5 show comparison accuracy on the graph.

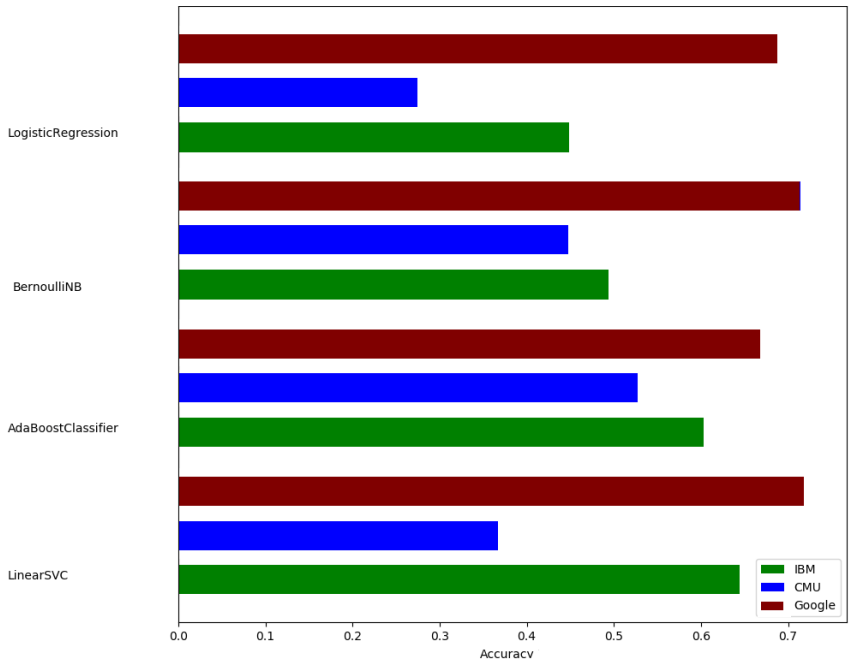

Figure 5: Comparison of accuracy of the classifiers on different IBM, Google and CMUSphinx generated training dataset

Table 2 shows precision, recall and F-score values of the classifiers on the dataset. Since the datasets were unbalanced, F-score is an appropriate measure to evaluate the classifiers. F-score is computed by taking harmonic mean of precision and recall and is measured from value 0-1 (Garfield and Wermter; 2006). SVM performed better than other classifiers since it has the highest f-score value in Google and IBM dataset compared to other classifiers.

	IBM			CMUSphinx			Google		
	Precision	Recall	F-score	Precision	Recall	F-score	Precision	Recall	F-score
Logit	0.46	0.46	0.46	0.28	0.27	0.27	0.59	0.69	0.63
BNB	0.37	0.50	0.42	0.39	0.42	0.39	0.73	0.73	0.73
Ada	0.51	0.62	0.55	0.29	0.54	0.37	0.75	0.69	0.71
SVM	0.63	0.65	0.63	0.32	0.38	0.34	0.78	0.73	0.74

Table 2: Precision, recall and F-score of classifier on different dataset

In conclusion, SVM classifier performance was better than other classifiers on both IBM and Google dataset but was poor on CMUSphinx dataset. Since classifiers had shown better performance on Google ASR dataset which also had the lowest WER, the dataset was used to evaluate the performance of the classifiers on different feature vectors to test the hypothesis.

5.5 Evaluation of classifiers on different feature vectors

To test the hypothesis that preserving semantic similarity of words in feature vectors will improve the accuracy of the classifiers, the Google dataset was used to compare the performance of classifiers on different feature vectors. The goal is to find if there is an improvement in the performance of the classifiers when word2vec feature vectors are used. Table 3 shows the accuracy of classifiers on bag-of-words, tf-idf, and word2vec feature vectors. It can be seen that all classifiers have shown better performance when numerical feature vectors from bag-of-words were fed. However, Classifiers performance did not improve when feature vectors from word2vec model were used and in fact, have performed worse on feature vectors from word2vec. SVM showed better performance compared to other classifiers on feature vectors from bag-of-words. While using feature vectors from word2vec model on Bernoulli Naive Bayes is the poorest performance with 27% accuracy. However, it performed much better on count features vectors and tf-idf feature vectors. There was a significant improvement in the accuracy of AdaBoost and SVM when count feature vectors from bag-of-words were used. Fig 6 shows bar graph of comparison of classifiers accuracy.

	Bag-of-words	tf-idf	Word2Vec
Logistic Regression	69.23%	66.76%	42.30%
Bernoulli Naive Bayes	71.32%	69.23%	26.92%
AdaBoost	68.23%	53.84%	42.30%
SVM	73.07%	53.88%	45.83%

Table 3: Accuracy of classifiers on different feature vectors

Table 4 shows precision, recall and f-score values of the classifiers. F-score, precision and recall values for SVM is highest when feature vectors from bag-of-words were used. SVM performed substantially better than other classifiers when count features vectors were used. However, SVM performance was poorest when tf-idf feature vectors were used. Bernoulli Naive Bayes performed poorly when feature vectors from word2vec were used while its performance significantly improved when count features were used.

	Bag-of-words			tf-idf			word2vec		
	Precision	Recall	F-score	Precision	Recall	F-score	Precision	Recall	F-score
Logit	0.59	0.69	0.63	0.73	0.73	0.72	0.46	0.42	0.43
BNB	0.73	0.73	0.73	0.68	0.69	0.67	0.25	0.27	0.25
Ada	0.75	0.69	0.71	0.58	0.54	0.55	0.41	0.42	0.41
SVM	0.78	0.73	0.74	0.29	0.54	0.37	0.35	0.48	0.46

Table 4: Precision, recall and F-score of classifier on different feature vectors

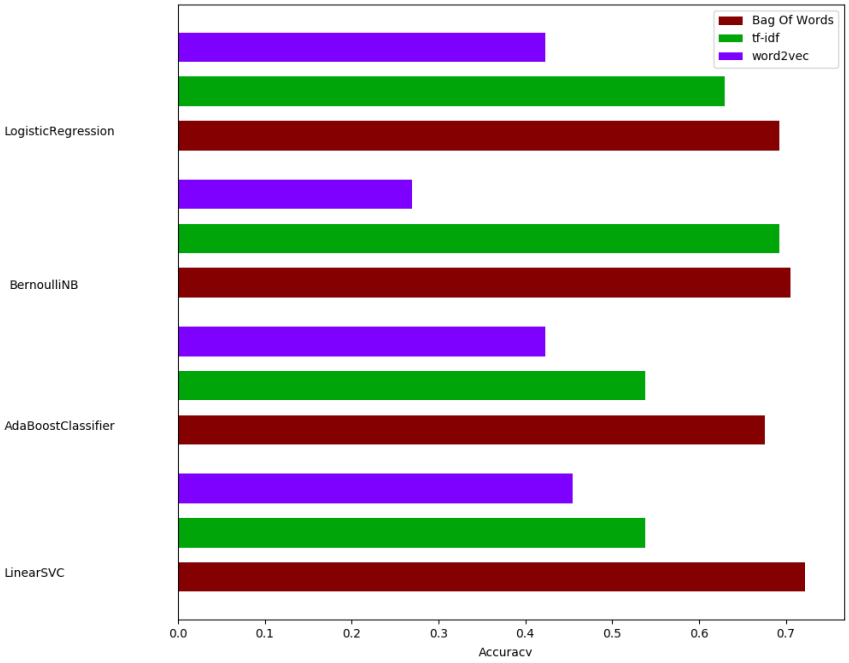

Figure 6: Comparison of accuracy on different feature vectors

In conclusion, the assumptions that word2vec feature vectors will improve the accuracy of the classifier was proved wrong since the feature vectors from word2vec did not improve classifiers accuracy when compared to feature vectors from bag-of-words and tf-idf. SVM performed substantially better than other classifiers even when the training dataset was unbalanced and significantly smaller in size. SVM and Naive Bayes gave the best result in the classification of utterance. The Accuracy of the classifiers can be improved by adding more training transcripts.

6 Conclusion and Future Work

The research paper presented utilization of speech utterance classification for triage of 911 emergency calls. For analyzing the performance of the classifiers for utterance classification four widely used text classification algorithms were evaluated. These classifiers were trained on 911 audio calls transcripts and tested on the first utterance transcripts of the callers. The classifiers were also evaluated on three different feature vectors to test the hypothesis. The results showed SVM had outperformed other classifiers when count features of Bag-of-words from Google ASR generated dataset were used. Hypothesis testing found that preserving semantic relationships in feature vector did not improve the accuracy of the classifiers over other feature vectors and the classifiers performed worst on those feature vectors. Classifiers performed much better when Simple bag-of-words vector representation of words as features vectors was used over other feature vectors representations. This means that meaning of words does not matter in the speech utterance classification task and the classifier's performance is independent of the choice of words of the utterance. Linear SVM used with One-vs-all techniques and Bernoulli Naive Bayes gave better classification accuracy and f-score compared to boosting and logistic regression.

For improving the accuracy of the classifier, a large number of transcripts will be

needed for training the classifier. The size of the training dataset is limited by the number due to a limited number of publicly available audio calls on the internet. Moreover, the recordings must be annotated to decide the class of the calls. The classifier needs to be trained using supervised learning method which will require training examples with the class labels. Determining the class label for the annotated audio recordings is a time-consuming process. The call recordings are often poor quality which results in a bad quality transcripts from ASR. Poor quality transcripts affect the training of the learning algorithm.

For practical application of speech utterance classification for triage of 911 calls, the classification model should have low classification error rate and low false negatives i.e. a call with high priority should not be labeled as low priority. The correct priority label of a call will depend on the accurate transcript of the utterance generated by the ASR. The ASR model should be robust to changes in speech level and correctly transcribe spontaneous speech. The system should constantly be re-trained constantly by adding new data to improve on certain calls. The limitations of this system are 1. The system will not be able to discern deceptive calls, and a deceptive will still be given a priority label. Moreover, an accidentally dialled call will also be given a priority label which will unnecessarily increase queue length of the calls. 2. The system will be restricted to only English speaker callers.

Future work should focus on using alternative approach of classification techniques such as neural networks to obtain better classification accuracy. However, a large amount training examples will be needed to train neural network models. Latent Semantic Analysis can be used for feature representation as an alternative to common feature representation like bag-of-words. The future work should also focus on obtaining clean transcript by sanitizing ASR output

Acknowledgements

I would like to thank my supervisor Lisa Murphy for her support and guidance throughout this research. I would also like to thank Dr. Simon Caton for showing me the right approach to successfully implement this research.

References

Aly, M. (2005). Survey on multiclass classification methods, *Neural Netw* **19**.

Attwater, D., Edgington, M., Durston, P. and Whittaker, S. (2000). Practical issues in the application of speech technology to network and customer service applications, *Speech Communication* **31**(4): 279–291.

belambert (n.d.). Python module for evaluating asr hypotheses (e.g. word error rate, word recognition rate) [software]. `https://github.com/belambert/asr-evaluation`.

Bird, S., Klein, E. and Loper, E. (2009). *Natural Language Processing with Python*, O'Reilly Media.

Buitinck, L., Louppe, G., Blondel, M., Pedregosa, F., Mueller, A., Grisel, O., Niculae, V., Prettenhofer, P., Gramfort, A., Grobler, J., Layton, R., VanderPlas, J., Joly, A., Holt,

B. and Varoquaux, G. (2013). API design for machine learning software: experiences from the scikit-learn project, *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*, pp. 108–122.

Burns, M. B. and Moffitt, K. C. (2014). Automated deception detection of 911 call transcripts, *Security Informatics* **3**(1): 8.

Chelba, C., Mahajan, M. and Acero, A. (2003). Speech utterance classification, *Acoustics, Speech, and Signal Processing, 2003. Proceedings.(ICASSP'03). 2003 IEEE International Conference on*, Vol. 1, IEEE, pp. I–I.

Christopher D. Manning, Prabhakar Raghavan, H. S. (2009). *Introduction to Information Retrieval*, Cambridge University Press.

Chu-Carroll, J. and Carpenter, B. (1999). Vector-based natural language call routing, *Computational linguistics* **25**(3): 361–388.

Dix, A. (2009). Human-computer interaction, *Encyclopedia of database systems*, Springer, pp. 1327–1331.

Faruquie, T. A., Rajput, N. and Raj, V. (2007). Improving automatic call classification using machine translation, *Acoustics, Speech and Signal Processing, 2007. ICASSP 2007. IEEE International Conference on*, Vol. 4, IEEE, pp. IV–129.

Freund, Y., Schapire, R. and Abe, N. (1999). A short introduction to boosting, *Journal-Japanese Society For Artificial Intelligence* **14**(771-780): 1612.

Freund, Y. and Schapire, R. E. (1995). A desicion-theoretic generalization of on-line learning and an application to boosting, *European conference on computational learning theory*, Springer, pp. 23–37.

Garfield, S. and Wermter, S. (2006). Call classification using recurrent neural networks, support vector machines and finite state automata, *Knowledge and information systems* **9**(2): 131–156.

Horvitz, E. J., Kapoor, A. and Basu, S. (2007). Automated call classification and prioritization. US Patent App. 11/770,921.

Huang, J. (2006). *Performance measures of machine learning*, University of Western Ontario.

Huang, Q. and Cox, S. (2004). Automatic call routing with multiple language models, *the Proceedings of HLT-NAACL 2004 Workshop on Spoken Language Understanding for Conversational Systems and Higher Level Linguistic Information for Speech Processing, Boston, MA*.

Joachims, T. (1998). Text categorization with support vector machines: Learning with many relevant features, *Machine learning: ECML-98* pp. 137–142.

Jurafsky, D. (2000). *Speech & language processing*, Pearson Education.

Kannan, S. and Gurusamy, V. (2014). Preprocessing techniques for text mining.

Kim, C., Misra, A., Chin, K., Hughes, T., Narayanan, A., Sainath, T. and Bacchiani, M. (2017). Generation of large-scale simulated utterances in virtual rooms to train deep-neural networks for far-field speech recognition in google home.

Kuo, H.-K. and Lee, C.-H. (2003). Discriminative training of natural language call routers, *IEEE Transactions on speech and audio processing* **11**(1): 24–35.

Landauer, T. K., Foltz, P. W. and Laham, D. (1998). An introduction to latent semantic analysis, *Discourse processes* **25**(2-3): 259–284.

LEITSC (2006). Law enforcement information technology standards council, standard functional specifications for computer assisted dispatch (cad) systems, pp. 1–56.

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

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S. and Dean, J. (2013). Distributed representations of words and phrases and their compositionality, *Advances in neural information processing systems*, pp. 3111–3119.

Ng, A. Y. and Jordan, M. I. (2002). On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes, *Advances in neural information processing systems*, pp. 841–848.

Niyogi, P., Pierrot, J.-B. and Siohan, O. (2000). Multiple classifiers by constrained minimization, *Acoustics, Speech, and Signal Processing, 2000. ICASSP'00. Proceedings. 2000 IEEE International Conference on*, Vol. 6, IEEE, pp. 3462–3465.

Patil, U., Shirbahadurkar, S. and Paithane, A. (2016). Automatic speech recognition models: A characteristic and performance review, *Computing Communication Control and automation (ICCUBEA), 2016 International Conference on*, IEEE, pp. 1–7.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M. and Duchesnay, E. (2011). Scikit-learn: Machine learning in Python, *Journal of Machine Learning Research* **12**: 2825–2830.

Porter, M. F. (2001). Snowball: A language for stemming algorithms.

Saon, G., Kuo, H.-K. J., Rennie, S. and Picheny, M. (2015). The ibm 2015 english conversational telephone speech recognition system, *arXiv preprint arXiv:1505.05899* .

Sarikaya, R., Hinton, G. E. and Ramabhadran, B. (2011). Deep belief nets for natural language call-routing, *Acoustics, Speech and Signal Processing (ICASSP), 2011 IEEE International Conference on*, IEEE, pp. 5680–5683.

Sarikaya, R., Kuo, H.-K. J., Goel, V. and Gao, Y. (2005). Exploiting unlabeled data using multiple classifiers for improved natural language call-routing, *Ninth European Conference on Speech Communication and Technology*.

Schapire, R. E. (1999). A brief introduction to boosting, *Ijcai*, Vol. 99, pp. 1401–1406.

Schapire, R. E., Rochery, M., Rahim, M. and Gupta, N. (2005). Boosting with prior knowledge for call classification, *IEEE transactions on speech and audio processing* **13**(2): 174–181.

Tyson, N. and Matula, V. (2004). Improved lsi-based natural language call routing using speech recognition confidence scores, *Computational Cybernetics, 2004. ICCC 2004. Second IEEE International Conference on*, IEEE, pp. 409–413.

Uysal, A. K. and Gunal, S. (2014). The impact of preprocessing on text classification, *Information Processing & Management* **50**(1): 104–112.

Wang, Y.-Y., Lee, J. and Acero, A. (2006). Speech utterance classification model training without manual transcriptions, *Acoustics, Speech and Signal Processing, 2006. ICASSP 2006 Proceedings. 2006 IEEE International Conference on*, Vol. 1, IEEE, pp. I–I.

Zhang, A. (n.d.). Speech recognition (version 3.7) [software]. `https://github.com/Uberi/speech_recognition#readme`.

Zitouni, I., Kuo, H.-K. J. and Lee, C.-H. (2001). Natural language call routing: towards combination and boosting of classifiers, *Automatic Speech Recognition and Understanding, 2001. ASRU'01. IEEE Workshop on*, IEEE, pp. 202–205.