A Machine Learning Approach to Topic and Sentiment Recognition in Customer Service Call Audio

Colorado School of Mines
Department of Computer Science

19 June 2018

Matthew Bussing
Junior, B.S. in Computer Science, CSM
mbussing@mines.edu

Tyler Doll
Senior, B.S. in Computer Science, CSM
tylerdoll@mines.edu

Kai Nichols
Senior, B.S. in Computer Science, CSM
nichols1@mines.edu

Sidney Johnson
Senior, B.S. in Computer Science, CSM
sajohnson@mines.edu

1 INTRODUCTION

Natural language processing (NLP) is a broad discipline concerning the problems of capturing, analyzing, and interpreting human language. In particular, NLP can be used to evaluate human speech in order to determine different qualities about its speaker, context, and sentiment. These evaluations can be used to inform human-computer interaction, accessibility, automation, and statistical analysis of natural language.

This NLP project was proposed by the Edge Intelligence group at ProKarma, an IT solutions company headquartered in Nebraska offering digital transformation and analytics consultancy. ProKarma's Edge Intelligence group is researching ways to improve customer service by incorporating machine learning to augment customer service calls in the telecommunications industry. One approach to improve customer service satisfaction includes determining customer attitudes toward particular products and by performing sentiment and topic analysis on customer call data, effectively allowing for automated customer service surveys.

Automating customer satisfaction data collection would provide actionable information about consumer attitudes toward products and services at much higher response levels than simply relying on customers staying on the line to finish a phone survey. This automated system could run parallel to the traditional phone surveys currently in place at many wireless carriers. Automated sentiment analysis on products mentioned during the call could be validated against instances where the customer finished the post-call phone survey.

Because of the huge number of variables involved in natural language, traditional analytic methods would be prohibitively difficult to adapt to this purpose. Machine learning algorithms allow simple linear interpolation of audio data as well as powerful pattern finding tools. For this reason, neural networks were developed for both topic and sentiment classification. These models find patterns between product types and consumer attitude which provides the information required to generate automated customer satisfaction surveys from customer call audio.

2 METHODS

Three models were required to accomplish the ultimate goal of an automated survey process: speech-to-text transcription, sentiment modeling, and topic modeling.

- *Speech-to-text transcription*
  In order to automate transcription of call data, a strategy for automated speech recognition (ASR) was required. An accurate and high-performance implementation of ASR was crucial in order to perform later topic and sentiment analyses, which take plain-language transcripts as input. Transcripts were preferable to pure audio as inputs for two reasons: firstly, topic modeling cannot be performed directly on audio, as topics are strings which must be pulled directly or interpreted from within a text source; secondly, audio files can be useful in detecting vocalized emotion but cannot detect unemotional sentiment (such as a monotonous customer saying "thank you for your help" or a bubbly customer happily saying "my phone won't turn on").

- *Sentiment model*
  As sentiment analysis is important to determine customer satisfaction, it was required to develop a machine learning model to detect sentiment (positive, negative, or neutral) towards the customer service representative based on transcribed text input. ProKarma's Edge Intelligence group previously implemented a bi-directional long short-term memory recurrent neural network (LSTM RNN) trained using open-source datasets such as the Interactive Emotional Dyadic Motion Capture (IEMOCAP) database in order to perform emotion and sentiment classification on acoustics features extracted from audio clips. During this project, our transcript-based model is intended to capture overall sentiment rather than from vocal features like the previous acoustic-based model captured.

– *Topic Model*
To cross-analyze customer sentiment with topics of conversation, classification models were produced to identify concepts and specific subjects in transcripts. Trends in conversational topics can be used to understand both individual customer attitudes toward products and services as well as to understand those of the general customer base.

These models culminated in an engine for end-to-end audio analysis of customer call recordings. All project code was written using Python 3.6, used Google's TensorFlow machine learning library via the Keras interface, and used the Python/R Anaconda distribution. Version control was performed using git and GitHub was used for hosting all project repositories. The project was developed using an Agile workflow, and team collaboration and communication was facilitated using the software, Slack.

3 SYSTEM ARCHITECTURE AND DESIGN

The aim of this project was to prove it is possible to take a customer service call as input and get data relevant for a customer satisfaction survey as output. The system constructed to accomplish this task, pictured in figure 1, consists of three main models: a speech-to-text model, a topic model, and a sentiment model. The speech-to-text model was trained on web-scraped customer service call audio files. The topic model was trained on a web-scrapped corpus of a wireless carrier's FAQs and forum posts. The sentiment model was trained on a combined dataset of annotated IEMOCAP transcripts and electronics reviews. The customer service call audio was then passed through the speech-to-text program to generate transcripts, which were then fed into the topic and sentiment models. These two latter models output predictions of overall topics and sentence-by-sentence sentiment tagging. These two outputs were then aggregated to produce visualizations of customer satisfaction indicators such as sentiment over time during the call and sentiment associated with specific topics.

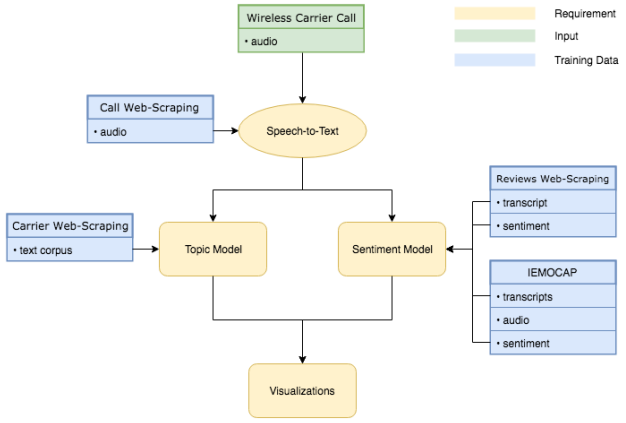

Figure 1: Diagram of general system architecture.

4 TECHNICAL DESIGN

The use of Convolutional Neural Networks (CNN) for natural language processing is a fairly recent development. CNNs are most well known for their success in image classification from filters' natural proficiency for identifying components of an image like lines and shapes. Whereas a CNN used on images can successfully find patterns in shapes, the filters of a CNN used on language is analogously able to find patterns in n-grams.

The network used for testing, pictured in figure 2 and elaborated on in figure 3, has two convolutional layers. The first convolutional layer has 64 filters with a filter size of 4, and the second has 32 filters with a filter size of 6. These were starting points to test that the model has functionality. These parameters could be further tuned to improve the accuracy of the model. A larger number of filters and filter sizes in the range of 3-5 are recommended for sentiment analysis using a CNN in order to capture a large amount of contextual information based on the relations between closely adjacent words. The dropout layer in the diagram is intended to prevent overfitting.

Latent Dirichlet Allocation (LDA) topic models are widely used for identifying underlying topics in a set of documents. Given K topics to find, an LDA model will sort words in a document into various topics based on the words they appear most frequently with and their respective topics. This works through a generative process where the LDA model assumes the following: a document is composed of an assortment of topics and each topic is composed of an assortment of words. It then reverse engineers this process in order to identify the topics in a corpus. Figure 4 is the graphical plate notation where:

– M is the number of documents
– N is the number of words in a document
– α is the per-document topic distributions
– β is the per-topic word distribution
– θ_m is the topic distribution for document m
– ϕ_k is the word distribution for topic k
– Z_{mn} is the topic for the nth word in document m
– W_{mn} is the specific word

For the purposes of sorting customer service transcript data into topics, an online technical support discussion board and FAQ section of a wireless carrier was used. By using this dataset, the topics could be predetermined and the model could be tuned to identify specific topics. Figure 5 provides examples of topic information which can then be used to identify topics. In order to do this, β was defined in the following way:

– For each topic, all words were initially given a 1 / *corpus vocabulary size* distribution
– Then words that were predetermined to be more frequent for a given topic were heavily biased (1000 times more weight)

This enabled the model to bias toward finding specific topics in a document rather than determining its own.

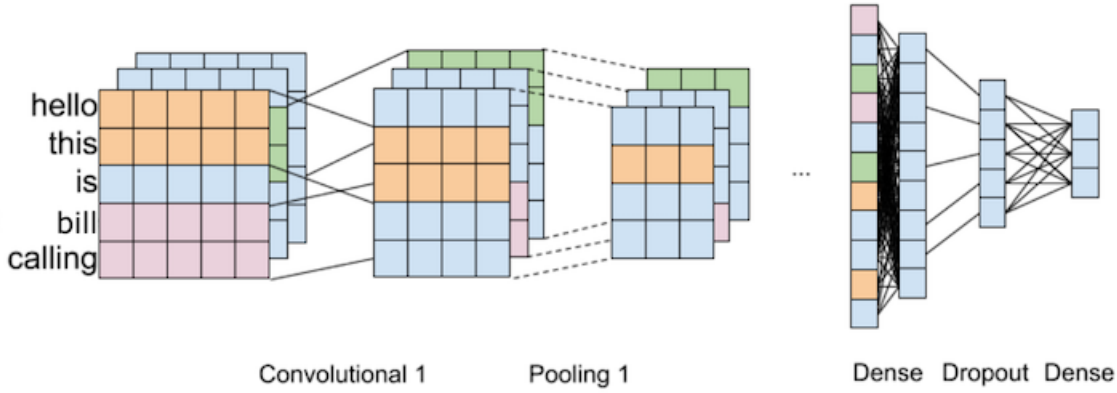

Figure 2: Convolutional neural network architecture.

Layer (type)	Output Shape	Param #
input_7 (InputLayer)	(None, 101)	0
embedding_7 (Embedding)	(None, 101, 100)	762400
conv1d_13 (Conv1D)	(None, 98, 64)	25664
max_pooling1d_13 (MaxPooling	(None, 49, 64)	0
conv1d_14 (Conv1D)	(None, 44, 32)	12320
max_pooling1d_14 (MaxPooling	(None, 22, 32)	0
flatten_7 (Flatten)	(None, 704)	0
dense_13 (Dense)	(None, 100)	70500
dropout_7 (Dropout)	(None, 100)	0
dense_14 (Dense)	(None, 3)	303

Total params: 871,187
Trainable params: 871,187
Non-trainable params: 0

Figure 3: CNN layer type, output shape, and number of parameters.

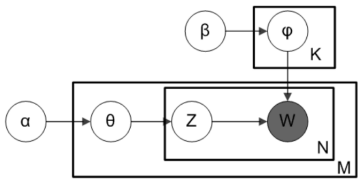

Figure 4: Diagram of smoothed LDA model.

5 DESIGN DECISIONS

Some tools were across all sections of the project. The general purpose data science tools used throughout the project include:

- *Python*
  Python is a powerful, general-purpose programming language with a robust set of libraries. It is an industry standard with wide support networks.
- *TensorFlow*
  TensorFlow is an open source library for deployment of high performance numerical computation across different platforms.
- *Anaconda*
  The Anaconda distribution contains a plethora of data science packages, including conda, which is an OS-independent package and virtual environment manager, allowing for easy package installation across machines.
- *Jupyter Notebook*
  Jupyter notebooks are documents which contain text and embedded code which can be easily run within the document. They are simple to share and provide high-quality, interactive output for our python code. The prose-like format of Jupyter notebooks' interpreter, markdown, and visualizations environment facilitates code and concept collaboration.
- *Pandas*
  Pandas is a powerful python library which is ubiquitous in data science. In this project it was used for managing training and testing data.
- *Matplotlib*
  Matplotlib is a python library for data visualization. In this project it is used for final visualization of sentiment analysis and topic modeling data.
- *Pickle*
  Pickle is a python library for object serialization and data storage. In this project it is useful for exporting model configurations and related dataframes.

As this was a far-reaching project with many moving parts, a variety of technical tools were required to specifically accomplish the goal of each major utility:

- *Speech-to-text transcription*
  ffmpeg: Due to the way speech-to-text engines process audio files, it is important to use audio encoding that is consistent with that expected by the engine. FFmpeg is a free, lightweight command-line audio encoder that fits seamlessly into our audio processing workflow. The final encoding pattern consisted of monochannel audio with a 16 kHz sampling frequency, a 16-bit bit depth, and 300/3000 Hz high/low pass filtering to specifically capture voice audio

	Topic	Words
0	Account__services	0.037*"account" + 0.036*"service" + 0.028*"digit" + 0.012*"call" + 0.008*"line"
1	Android_products	0.088*"lg" + 0.081*"galaxy" + 0.079*"android" + 0.012*"video" + 0.012*"one"
2	Apple_products	0.078*"i" + 0.068*"apple" + 0.066*"io" + 0.010*"update" + 0.007*"unlock"
3	Network__Coverage	0.116*"coverage" + 0.113*"network" + 0.008*"tower" + 0.007*"device" + 0.006*"block"
4	Other_products	0.010*"data" + 0.009*"plan" + 0.009*"update" + 0.008*"work" + 0.008*"problem"

Figure 5: Example of raw output from the LDA model.

Audacity: Beyond encoding, our data required a small amount of editing, mostly in the form of phrase partitioning. Audacity's "Silence Finder" utility allowed us to break long audio clips into phrase-long sound bites and export them easily for immediate transcription processing in DeepSpeech.

DeepSpeech: DeepSpeech is a free speech-to-text engine with a high accuracy ceiling and straightforward transcription and training capabilities. DeepSpeech also comes with pretrained models that can be refined via transfer learning to improve accuracy on any particular type of audio, avoiding the need for extended training times or an extremely large dataset. Our transfer learning process involved removing the final interpretive layer of DeepSpeech, replacing it with a randomly-initialized layer, freezing all sublayers and training the final layer.

– *Topic model*
gensim: gensim is a general topic modeling library for python that has many useful features. Gensim was used primarily for its LDA model as it is very powerful and easy to train and tweak to our needs.

NLTK: NLTK stands for Natural Language Toolkit and is a python library used for natural language processing. This library was used for our topic model to normalize documents before running them through our LDA model.

pyLDAvis: pyLDAvis is a python package used for visualizing LDA models and was used for this project because it interfaces well with gensim. pyLDAvis allowed us to easily visualize our LDA model's performance.

Stanford NER: Stanford Named Entity Recognizer model was used for identifying named entities in a corpus such as persons, locations, and organizations. This helped us identify the subject of a document.

Stanford POS Tagger: Stanford Part of Speech Tagger was used for identifying the part of speech for each word in a corpus. This helped us identify the subject, verb, and object of call transcript sentences.

– *Sentiment model*
Word Embedding: Tensorflow_hub's word embedding modules were chosen as input for our neural network classification models since they allow us to map sentences to numerical values and preserving word order information. These modules are easy to implement through tensorflow_hub and come pretrained.

Manual Feature Selection *(MFS)*: Manual feature selection was used for the SVM and Random Forest classifier models because it allowed faster training, and therefore could handle a larger number of features. This allowed us to train our model to recognize vocabulary specific to the topic at hand.

(MFS) NLTK Stemming: In manual feature selection, words were stemmed so that words with the same root would be recognized as the same word which reduced our feature space. NLTK's Stemming module was chosen for ease of use.

(MFS) Filtering: Words longer than 25 characters, words shorter than 3 characters, words appearing more than 2000 times, and words appearing less than 5 times were all filtered out to reduce the feature space.

Bigrams: Bigrams were added to the feature space to embed some information on word order.

Classification Models: A variety of classification algorithms were explored to have comparisons for the accuracy of our models on the training data. The models explored were an SVM classifier, Random Forest classifier, LSTM Recurrent Neural Network, and Convolutional Neural Network.

Sklearn, Tensorflow, and Keras: Tensorflow and Keras were used for neural network training. Sklearn was used for the other classification algorithms.

NLTK and NLTK-trainer: NLTK was used for sentiment analysis, since it has several prebuilt sentiment models. This lets us quickly set up a standard model and compare against our own model iterations.

Classifier	Accuracy (Train/Test)	Accuracy (Train/Test)
	IEMOCAP	Amazon Reviews
SVM	0.804/0.520	0.999/0.623
RandomForestClassifier	0.860/0.539	0.988/0.627
DNNClassifier		
word2vec 250v	0.757/0.550	0.790/0.672
word2vec 500v	0.812/0.549	0.831/0.681
universal sentence encoder	**0.913/0.578**	0.902/0.740
nnlm-en	0.874/0.574	0.821/0.665
CNN (30 epochs)	0.820/0.474	0.992/0.656
LSTM (30 epochs)	0.853/0.534	**1.000/0.749**
CNN-LSTM (30 epochs)	0.876/0.516	1.000/0.708

Figure 6: Accuracy results of different classifiers for sentiment.

Feature Implementation	DNN	Accuracy for Random Forest	CNN
Manual Feature Selection	0.869/0.542	**0.863**/0.548	xxx
Tf_hub word embedding module	**0.913/0.578**	xxx	**0.921/0.635**
Keras Tokenization	xxx	0.627/**0.598**	0.820/0.474

Figure 7: Accuracy results of different feature implementations.

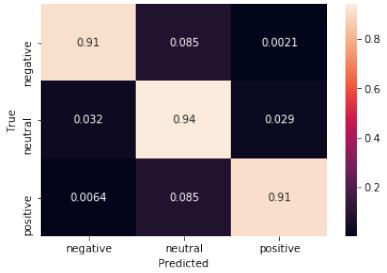

Figure 8: Confusion matrix for training data on the final LSTM model.

Figure 9: Confusion matrix for testing data on the final LSTM model.

6 PERFORMANCE RESULTS

– *Speech-to-text transcription*
DeepSpeech, the automated speech recognition (ASR) engine preferred by our client, is an end-to-end trainable, character-level LSTM recurrent neural network. DeepSpeech has been used to produce a very low word error rate (WER), particularly a 6% WER on LibriSpeech, an open-source 1000 hour dataset of natural speech. 6% is approximately the human error rate. Out of the box, DeepSpeech's generic models returned a 65.4% WER on our customer service call testing data, more than 10x the WER of human processing.
This WER was ultimately reduced to approximately 18.7% on the custom model generated using a very small corpus of training data.
DeepSpeech functions best when translating sentence-length, <10 second audio clips. The processing time required by DeepSpeech to create a transcript per audio clip is between 4 and 9 seconds, with a mean of 7.14 seconds. Therefore, 5 minutes of audio could take DeepSpeech almost 4 minutes of processing time.

– *Sentiment model*
Accuracy of various models were compared on 80/20 split training/testing data made up of the IEMOCAP dataset and a web-scraped dataset of electronic reviews with the sentiment classification accuracies contained in figure 6. These were 3-class classifiers where the sentiment classes were *positive*, *negative*, and *neutral*. Accuracies of different feature implementations were then compared, with results contained in figure 7. Some feature implementations did not easily lend themselves to being run on different classification models, so that data is absent.
Among the different datasets and implementations the best results were on Neural Network Classifiers using the `tf_hub` word embedding module. Of the word embedding modules, the universal sentence encoder outperformed the other modules. A deeper neural network outperformed sklearn's DNNClassifier under the same feature implementation, and was the highest performer of all the implementations at about 10% higher test set accuracy than the second contender. The best model iteration was an LSTM RNN that used `tf_hub's` universal sentence encoder as feature input. This model had a test set accuracy of 77.6% on a combined IEMOCAP and electronic review dataset. Baseline accuracy for randomly choosing a class for 3 classes is 33.3%.

– *Topic model*
Using data scraped from a wireless carrier's online discussion forums and FAQ section, the training data was organized into 5 topics: accounts and services, android products, apple products, network and coverage, and other products. Each of these topics also included some keywords that were seeded into the LDA model with bias (such as "iphone" for apple products, "galaxy" for android products, etc). Using this training data to train the LDA model, it was able to successfully predict the topics on the testing data 57.6% of the

time. Baseline accuracy for randomly choosing a class for 5 classes is 20%. A visualization of this LDA can be found in figure 10.

The part-of-speech suject identifier (PoSSI) model was trained using corpora included with the Natural Language Toolkit (NLTK) and then tested on the web-scraped wireless carrier data described above. This model was able to produce coherent subject-verb-object (SVO) results, but because the web-scraped dataset is unlabeled, there is not a quantitative way of analyzing the accuracy unless all web-scraped data was hand-labeled for part-of-speech. Hand labeling the entire web-scraped dataset was not in the scope of this project.

– *Visualization and analysis of customer call data*
Visualizing sentiment over the course of a call showed trends we might expect: a green, positive, first sentence introduction followed by a red, negative, introduction of the problem, and darker greens (i.e. more strongly positive sentiment) towards the end of the call.
These are promising results that suggest sentiment analysis could be applicable in the context of automating customer satisfaction surveys.
A visualization of call topics is available in figure 10, and visualizations for sentiment analysis available in figures 11 and 12. To see more call sentiment analysis overviews see appendix C.

7 CONCLUSION

– *Speech-to-text transcription*

DeepSpeech is highly trainable but, with 120 million parameters, requires massive training corpora, hundreds of hours of training, and expensive equipment. Transfer learning was performed in the form of freezing all sublayers of the model and retraining the final layer. Experimentation with altering previous layers was unable to create improvements. Transfer learning is a viable course of action to customize DeepSpeech for future transcription requirements but should be trained using a greater number of high-quality audio clips. DeepSpeech performed significantly better on our testing data than alternative ASR engines such as Kaldi and CMUSphinx as well as traditional ASR approaches such as Hidden Markov Models.

Future work:
Further improvements could likely be made in preparing low-quality call data for transcription, as well as in post-processing the transcriptions to produce more correct speech (as DeepSpeech is character-level, it sometimes produces nonsense). Retraining the weights of DeepSpeech on a large, industry-specific copora would also likely produce greater accuracy, but would be significantly more expensive - both in computation time and amount of data required.

– *Sentiment model*

Testing results for the LSTM sentiment model are available in figures 8 and 9. The two biggest impacts on the accuracy of the tested sentiment analysis models were data preparation and model selection. Data preparation and input formatting has a larger impact on a smaller dataset. Model selection has a larger impact on a larger data set. Throughout the various model iterations we tried, three different types of formatting for data input were used:

– Manual Feature Selection (used in SVM and RandomForestClassifier)
Manual Feature Selection was done by using the techniques stemming, filtering, and tokenization. Words were stemmed using NLTK's word stemming module, so all words of different tenses would correspond to the same token. Words were not included in the training/testing sets when their length was less than 3, greater than 25, when they appeared more than 2000 times or less than 5 times in the full corpus. Tokens were then created for all viable unigrams and bigrams in the corpus.
– Pre-trained Word Embedding Modules (used in DNNClassifier)
  * Pre-trained word embedding modules from `tensorflow_hub` were used for the DNN Classifier.
  * The NNLM embedding is based on a neural network language model with two hidden layers. The least frequent tokens are hashed into buckets.
  * Word2vec is a token based embedding module trained on a corpus of English Wikipedia articles.
  * Universal-Sentence-Encoder encodes sentences for use in text classification. It was trained using a deep averaging network and has the feature of encoding semantic similarity.
– Tokenization
Words with a length less than one were removed, then sentences were vectorized using Keras' text preprocessing module. A token was chosen for every word in the training input. All sentences were padded to be of equal length, and words were replaced with tokens.

Given the relatively small vocabulary size of our data set, using a Pre-trained word embedding module gave the best results, outweighing the effect of what model was used. Alternative models were also developed, such as:

– SVM and Random Forest: These two models were the first to be run as a way to gauge what sort of accuracy was attainable, and what more complex models should be expected to beat. The sklearn SVM Classifier and RandomForestClassifier were used for this task. These are are two types of commonly used high performing classifiers.
– DNN: This model is a prebuilt Deep Neural Network from Tensorflow. This model was used to evaluate the performance of a Neural Network on the dataset, and to test use of `tf_hub's` word embedding modules.
– LSTM: LSTMs are currently an industry standard model for sequential data like natural language.

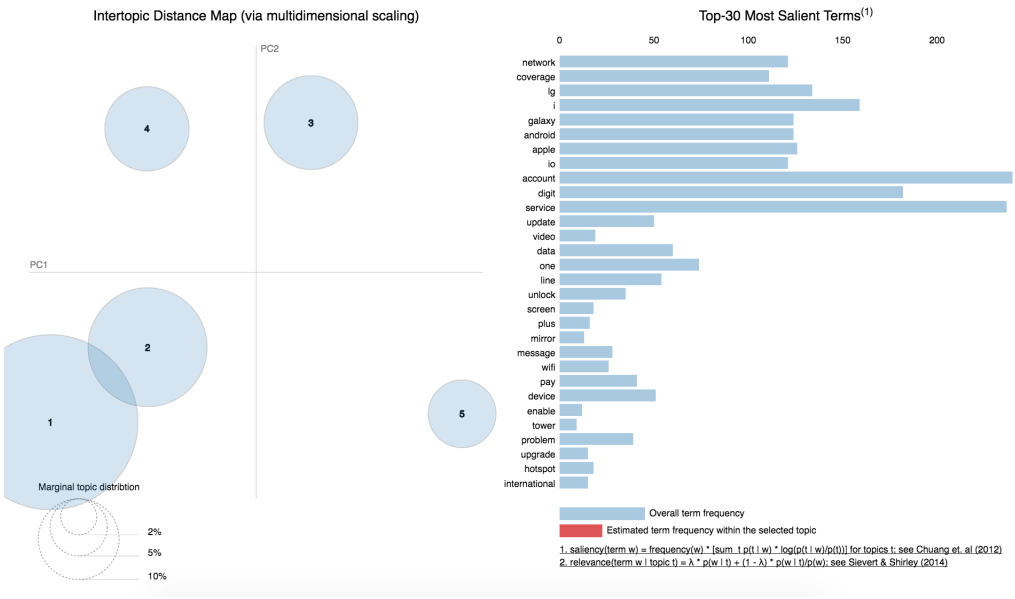

Figure 10: A visualization of the topics and the vocabulary found in the training corpora.

Figure 11: Top left: sentiment for each clip of the call displayed over the call transcript; Top right from top down: a heatmap of the intensity of the sentiment of each sentence of the call, a line graph displaying the intensity of the sentiment during each sentence of the call; Middle: the distribution of probabilities the call belongs into one of the five topic categories; Bottom: the distribution of sentiment among keywords for each category.

- CNN: Convolutional neural networks have been shown to provide excellent results on sentence classification by capturing information about n-grams through the use of convolutional layers.
- CNN-LSTM: a combined CNN-LSTM model has been shown to work well by encoding the regional information with convolutional layers and long distance dependencies across sentences using LSTM layers.

With a small-to-mid sized training dataset, there is a much higher risk of overfitting to your training set when using neural networks with thousands of parameters, such as RNN and CNN neural networks. This can be seen by large discrepancies between testing and training set accuracies in

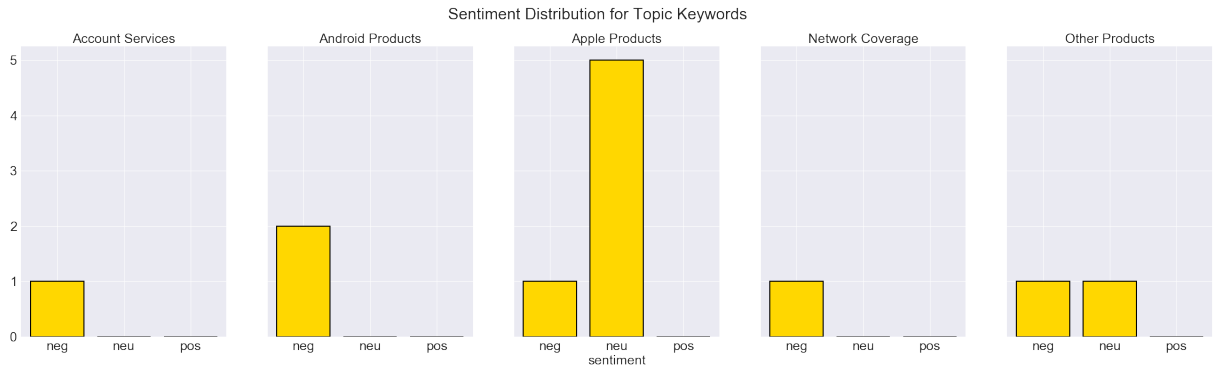

Figure 12: Sentiment distribution for sample service keywords.

our performance results. The ideal point at which to stop training is right as the training and testing scores begin to diverge where the training score continues going up, but the testing score starts going down. In the context of training a model, this means it is important to save checkpoints as you train, so the model can be reverted to an older checkpoint which is not yet overfitting. See Appendix B for an example of training and testing scores over the epochs of training. Overfitting can also be reduced by adding a regularization parameter to the loss function or to introduce dropout layers. Adding dropout to layers has seen success in most types of neural networks. However, this method does not work well on RNNs with LSTM units, which is the type of model that showed the greatest performance in this project. To mitigate overfitting on our small training dataset we apply dropout only to a non-recurrent layer.

Future work:

- Manual Feature Selection: When doing manual feature selection the NLTK stemming module was used to find word roots. Lemmatization would likely give better results, but requires part of speech tagging. Implementing this would create a more accurate feature space. This could be done using NLTK's WordNetLemmatizer and NLTK's `pos_tag` in conjunction.
- Training Data: Use a larger training dataset. The IEMOCAP data set has a vocabulary size of 3,000 words, which is relatively small, and testing data could easily contain unseen words. This was improved by the inclusion of the electronic review dataset which increased the vocabulary size to around 8,000 words. The data, furthermore, is not as domain specific as could be. IEMOCAP very general purpose, and performed by actors reading scripts about everyday situations. The electronic reviews are for electronic products similar to those talked about in a technical troubleshooting wireless carrier customer call, but are not recent. More data specific to the content discussed in wireless carrier customer service calls could be introduced, such as more recent electronic device reviews or manually annotated wireless carrier discussion forum posts.
- Pre-Trained Word Embedding Modules: Transfer learning could be implemented on these modules using a corpus of text relevant to the type of text data on which classification is being attempted. In this case the word embedding modules could be further trained on a corpus of wireless carrier related text, such as data scraped from a specific wireless carrier's online FAQ section and discussion forum posts.
- CNN and LSTM Neural Networks: The fine tuning of layers and their parameters, and feeding the model more data could be improved on the neural network models. These efforts would be for the purpose of increasing accuracy and decreasing overfitting.

- *Topic model*

  The low accuracy result of the LDA model is expected as LDA models are not meant to be forced into predefined topics and instead should be used for finding hidden topic information in the corpus. SVO extraction using POS and NER tagging was able to produce coherent results, but these results are not consistent across topics which presents a challenge when trying to sort documents by topics.

  Future work:
- The LDA model could be improved by using a more domain-specific part-of-speech tagger (rather than the default POS tagging function from NLTK) as well as better refined alpha (document-topic density) and beta (topic-word density) parameters.
- The PoSSI model could be improved in a few ways:
  * Using a POS tagger and a Named Entity Recognition (NER) tagger trained on domain specific corpora
  * A more domain-specific model for the POS tagger (currently uses a research model from Stanford)
  * A more domain-specific model for the NER tagger (currently uses a research model from Stanford)
  * A better method of extracting the subject from a tagged document such as a parser in combination with a context-free grammar for the domain specific corpora
  * Speed improvements as the model can be very slow for large documents.

Both of these models could be combined to identify topics in the generated SVOs.

8 BIBLIOGRAPHY

A. G. Tripathi and N. S, "Feature Selection and Classification Approach for Sentiment Analysis," Machine Learning and Applications: An International Journal, vol. 2, no. 2, pp. 01-16, 2015.

B. C. Busso, M. Bulut, C. Lee, A. Kazemzadeh, E. Mower, S. Kim, J. Chang, S. Lee, and S. Narayanan, "IEMOCAP: Interactive emotional dyadic motion capture database," Journal of Language Resources and Evaluation, vol. 42, no. 4, pp. 335-359, December 2008.

C. Y. Bengio, R. Ducharme, P. Vincent, and C. Jauvin, "A Neural Probabilistic Language Model," Journal of Machine Learning Research , vol. 3, pp. 1137-1155, Mar. 2013.

D. T. Mikolo, K. Chen, G. Corrado, and J. Dean, "Efficient Estimation of Word Representations in Vector Space," CoRR, vol. abs/1301.3781, Jan. 2013.

E. D. Cer, Y. Yang, S.-yi Kong, N. Hua, N. Limtiaco, R. St. John, N. Constant, M. Guajardo-Cespedes, S. Yuan, C. Tar, Y.-H. Sung, B. Strope, and R. Kurzweil, "Universal Sentence Encoder," CoRR, vol. abs/1803.11175, 2018.

F. W. Zaremba, I. Sutskever, and O. Vinyals, "Recurrent Neural Network Regularization," CoRR, vol. abs/1409.2329, 2014.

G. M. Hu and B. Liu. "Mining and summarizing customer reviews." Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery Data Mining, Aug. 2004.

H. N. Jindal and B. Liu. "Opinion Spam and Analysis." Proceedings of First ACM International Conference on Web Search and Data Mining, Feb. 2008.

I. Qian Liu, Zhiqiang Gao, Bing Liu and Yuanlin Zhang. Automated Rule Selection for Aspect Extraction in Opinion Mining. Proceedings of International Joint Conference on Artificial Intelligence, July 2015.

J. Y. Kim, "Convolutional Neural Networks for Sentence Classification," CoRR, vol. abs/1408.5882, 2014.

K. TensorFlow. (2018). Text Modules | TensorFlow. [online].

L. X. Ma and E. Hovy, "End-to-end Sequence Labeling via Bi-directional LSTM-CNNs-CRF," Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, vol. 1, 2016.

M. J. Cheng, L. Dong, and M. Lapata, "Long Short-Term Memory-Networks for Machine Reading," Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, 2016.

N. Y. Goldberg, "A Primer on Neural Network Models for Natural Language Processing," Journal of Artificial Intelligence Research, vol. 57, Nov. 2016.

O. J. Wang, L.-C. Yu, K. R. Lai, and X. Zhang, "Dimensional Sentiment Analysis Using a Regional CNN-LSTM Model," Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers), 2016.

P. A. Y. Hannun, C. Case, J. Casper, B. Catanzaro, G. Diamos, E. Elsen, R. Prenger, S. Satheesh, S. Sengupta, A. Coates, and A. Y. Ng, "Deep Speech: Scaling up end-to-end speech recognition," CoRR, vol. abs/1412.5567, 2004.

Q. J. Kunze, L. Kirsch, I. Kurenkov, A. Krug, J. Johannsmeier, and S. Stober, "Transfer Learning for Speech Recognition on a Budget," Proceedings of the 2nd Workshop on Representation Learning for NLP, 2017.

R. S. Bansal and Natural Language Processing and Machine Learning, "Beginners Guide to Topic Modeling in Python," Analytics Vidhya, 29-Aug-2016. [Online].

S. J. Brownlee, "How to Develop an N-gram Multichannel Convolutional Neural Network for Sentiment Analysis," Machine Learning Mastery, 14-Feb-2018.

T. S. Axelbrooke, "LDA Alpha and Beta Parameters - The Intuition," LDA Alpha and Beta Parameters - The Intuition | Thought Vector Blog, 21-Oct-2015. [Online].

U. A. Crosson, "Extract Subject Matter of Documents Using NLP," Medium, 08-Jun-2016. [Online].

V. J. R. Finkel, T. Grenager, and C. Manning, "Incorporating non-local information into information extraction systems by Gibbs sampling," Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics - ACL 05, pp. 363-370, 2005.

W. K. Toutanova and C. D. Manning, "Enriching the knowledge sources used in a maximum entropy part-of-speech tagger," Proceedings of the 2000 Joint SIGDAT conference on Empirical methods in natural language processing and very large corpora held in conjunction with the 38th Annual Meeting of the Association for Computational Linguistics, pp. 63-70, 2000.

X. S. Bird, E. Klein, and E. Loper, Natural language processing with Python: Analyzing Text with the Natural Language Toolkit. Beijing: OReilly, 2009.

Y. Smoothed LDA. Wikipedia, 2009.

Z. S. Ruder, "Deep Learning for NLP Best Practices," Sebastian Ruder, 25-Jul-2017. [Online].