Topic Modeling with Medical Transcripts Data

Austin Lasseter
Udacity Machine Learning Engineer Nanodegree
Final Report

1a. Project Overview

This capstone project uses machine learning methods to explore a dataset of medical transcripts. The exploration of medical data is one of the fastest growing fields in technology and especially in machine learning and data science. Technological advances in handling and analysis medical data can provide meaningful advances to improve healthcare and well-being. But a great deal of medical information is often locked up in text data, making it difficult to analyze. Traditionally, text from medical transcriptions required a human reader to review and summarize information. Now, thanks to Natural Language Processing (NLP) methods, it is possible to automatically review and summarize large amounts of medical text data.

Topic modeling is a strong approach for this field because it does not require labeled data; it is an unsupervised method that clusters documents according to similarity, and extracts key terms.

The data source for this analysis comes from Kaggle (https://www.kaggle.com/tboyle10/medicaltranscriptions). It includes several thousand rows of medical transcription data scraped from the website `mtsamples.com`. Medical data most of the time is very hard to find due to HIPAA privacy regulations which protect individuals. This dataset offers medical transcription samples that have been anonymized, stripping all personally identifiable information so that they can be publicly released for analysis. This dataset contains sample medical transcriptions for various medical specialties.

1b. Problem Statement

The analysis tackles a common problem faced by medical agencies, which is the need to rapidly process and categorize large amounts of medical transcripts that have no labels. Typically after an appointment, consultation or surgery, the medical practitioner takes the time to record a summary of the activities completed in text form. The transcript is saved but often it is not processed or tagged appropriately, making it difficult to analyze further or group with similar transcripts.

The purpose of this analysis is to provide a machine learning method for clustering unlabeled transcripts into similar topic clusters, identified by keywords. The LDA model can allow a practitioner to quickly summarize thousands of transcripts, and to accurately classify new transcripts as they come in.

1c. Metrics

There are multiple ways to evaluate an LDA topic model, and there are also several metrics to choose from. For my analysis, I have chosen three: topic coherence, perplexity, and classification accuracy.

Topic Coherence measures score a single topic by measuring the degree of semantic similarity between high scoring words in the topic. These measurements help distinguish between topics that are semantically interpretable topics and topics that are artifacts of statistical inference. Higher is better.

Perplexity captures how surprised a model is of new data it has not seen before, and is measured as the normalized log-likelihood of a held-out test set. The benefit of this statistic comes in comparing perplexity across different models with varying s. The model with the lowest perplexity is generally considered the “best”.

A third way to evaluate an LDA model is to re-assign the topics to the original dataset, then conduct a train-test split and see how well a classification algorithm performs with the new data. If it classifies the test data with a high degree of accuracy, then the topics are consistent and well-chosen.

In my analysis, I began with a benchmark model and then compared it to six other models with regard to coherence, perplexity and accuracy.

2a. Data Exploration

Prior to carrying out the topic analysis, I chose to explore and clean the dataset, and then preprocess it for NLP. There were several steps to the data exploration.

First, I examined file for any missing data. There were 33 rows that did not have any transcript text, so I deleted these. I also renamed the columns by removing the leading space from the beginning of each column name. I then examined the most important columns for patterns and trends; the visualizations are mentioned in the next section.

In addition to cleaning the dataset and removing rows with missing data, it was necessary to store the data to an S3 bucket using Amazon web services (AWS). The built-in SageMaker LDA algorithm is not compatible with CSV data and can only ingest the recordIO protobuf format, so I had to convert the dataset to this format prior to saving it on S3.

The corpus of text data, after it has been vectorized using gensim, is stored as the object `corpus` in sparse matrix format. However, I had to convert it to a dense matrix prior to converting it to recordIO protobuf, as follows:

```
dense_matrix = gensim.matutils.corpus2dense(corpus, num_terms=len(dictionary)).T
```

Once it was in dense format, I converted to recordIO as follows.

```
# convert documents_training to Protobuf RecordIO format
recordio_protobuf_serializer = numpy_to_record_serializer()
fbuffer = recordio_protobuf_serializer(dense_matrix)
```

At this point the data was ready to be processed by Amazon SageMaker using the LDA algorithm. Ultimately I tried several different versions of LDA and did not use the built-in SageMaker version but instead the `gensim` version.

2b. Exploratory Visualization

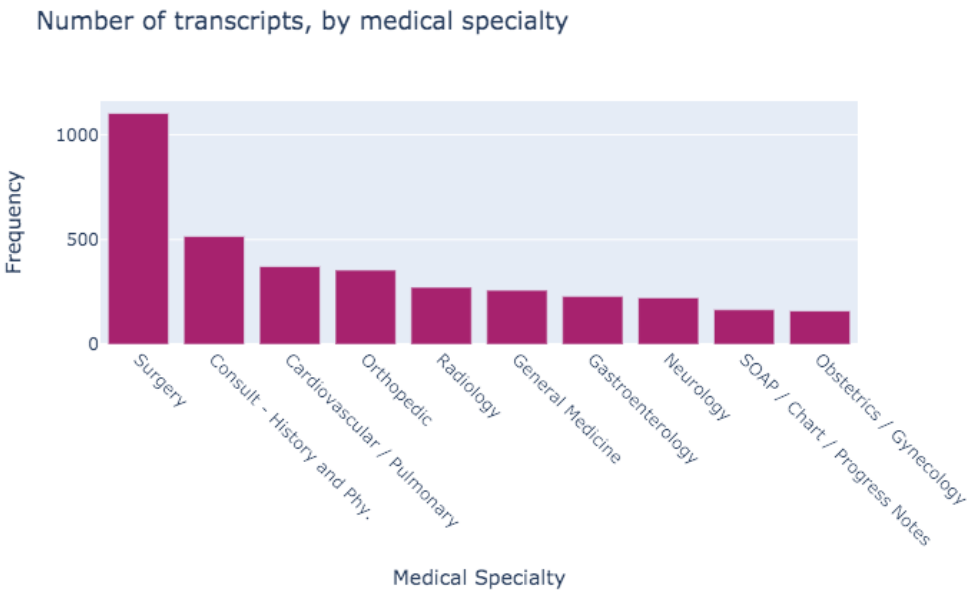
Prior to beginning the LDA analysis I wanted to examine trends in the dataset using explanatory visualizations. The data is in text format so there's not much numeric data that can be visualized, but I did want to examine the transcripts for any patterns, and also choose an appropriate subset of data.

The exploratory visualizations can be viewed on my project website here: http://sagemaker-lda-medical.s3-website-us-east-1.amazonaws.com/

The code associated with the exploratory analysis is available on github in the following jupyter notebook: https://github.com/austinlasseter/sagemaker_lda_medical/blob/main/eda-medical.ipynb

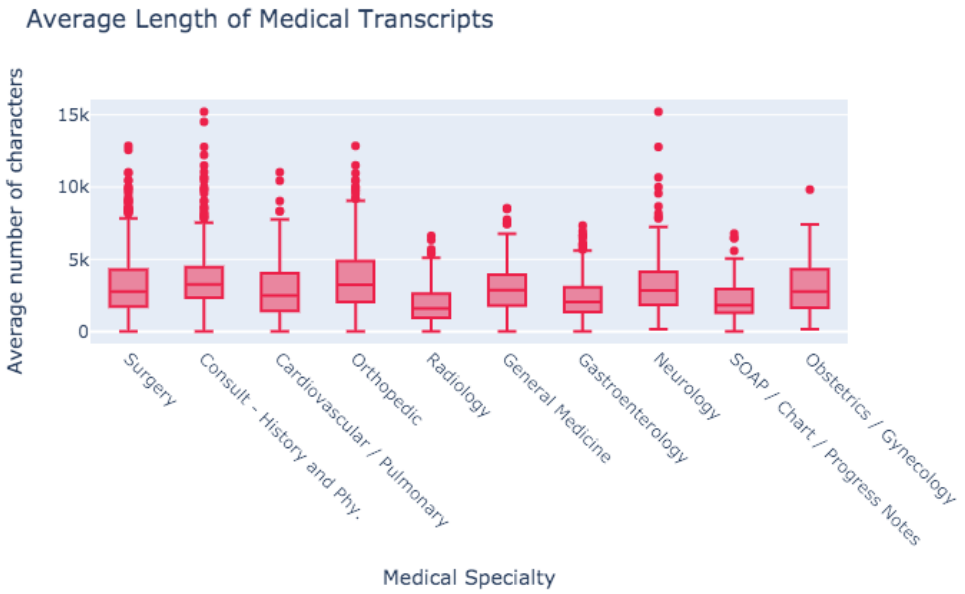
The transcripts are clustered according to a large number of medical specialties. The great majority of the transcripts are from "Surgery." Other common categories were "Consultation" and "Cardio-Pulmonary." I looked at all of these, and tried some exploratory LDA analayses with several categories of medical specialty. I found that it did not make sense to conduct LDA on the

dataset as a whole, because transcripts from different specialties had little or nothing in common. The LDA technique would simply cluster the transcripts according to their respective specialties, which was not useful. In order to take full advantage of the LDA technique, I limited my analysis only to transcripts from the "Surgery" specialty. It was the largest category, with enough data to create some meaningful and robust topic clusters.
Below is a graph showing the number of transcripts, by specialty



I also chose to explore the length of the transcripts. I wanted to see if there was sufficient data in the average length, to conduct a robust analysis using Topic Modeling. I also wanted to see if some specialties were longer than others. The Surgery transcripts were about average compared to the other specialty transcripts. They had a median length of 2,761 characters, and outliers as large as 12K characters.

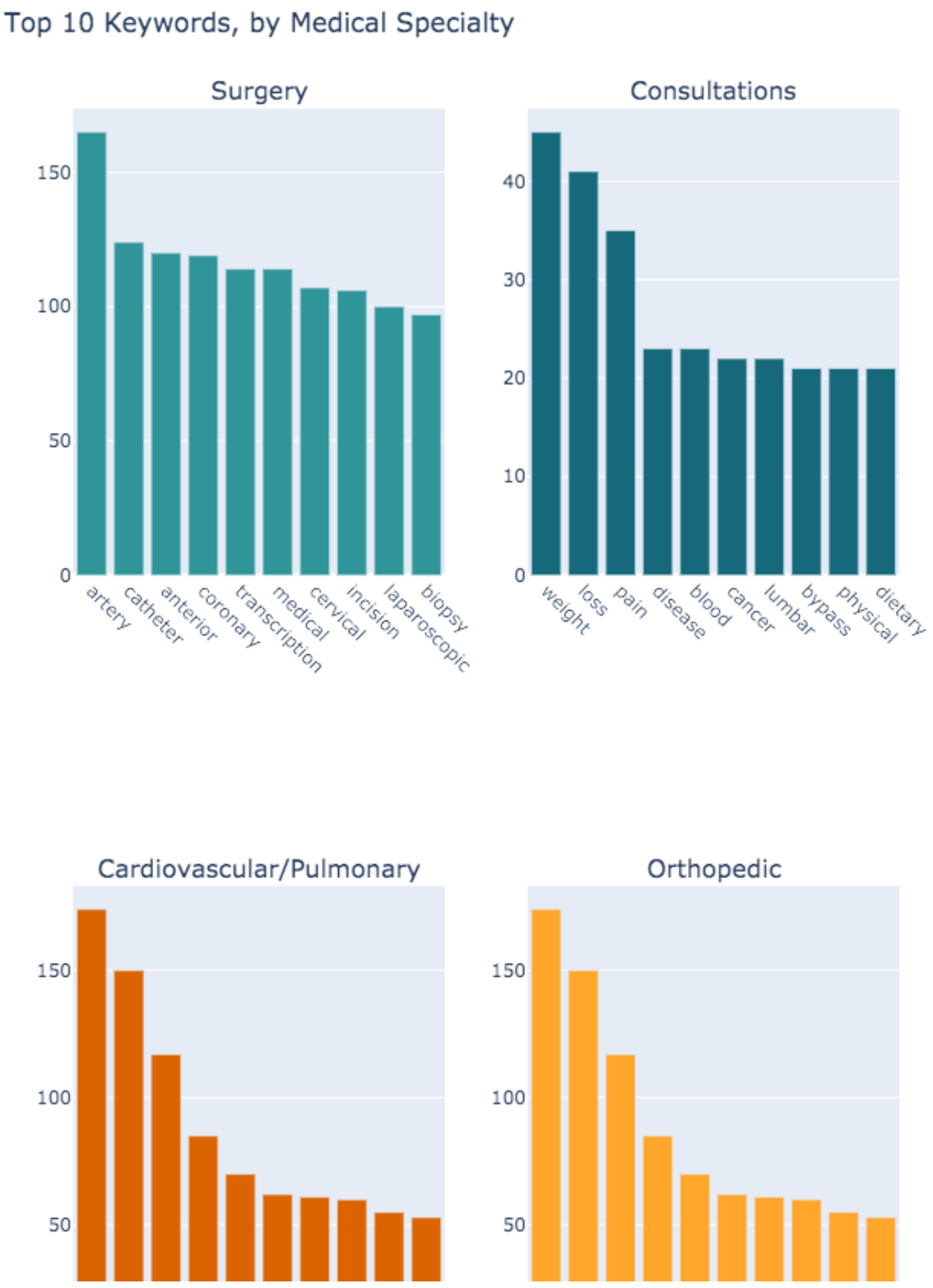
Length of transcripts, by specialty

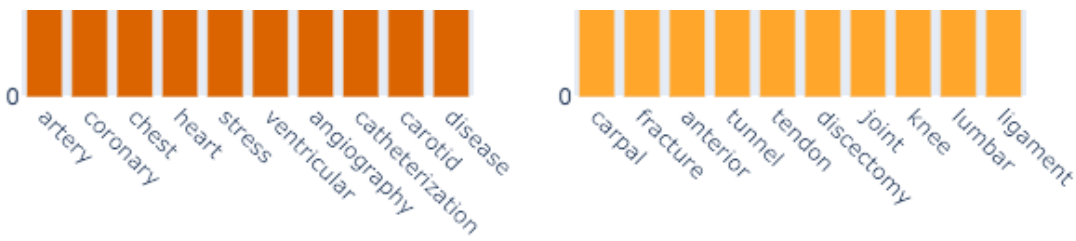


I also examined the four largest medical specialties by their top 10 keywords. The dataset includes a column of keywords extracted from the transcripts; these are not the same for every row. Using a simple count vectorizer technique, I extracted the list of

keywords and their frequency across each medical specialty. I also eliminated any keywords that were the same as the specialty itself -- for example, "surgery" was the most common word in the "Surgery" specialty. The top 10 keywords for the surgery specialty are shown in the graphic below. Of these, "artery" appears over 150 times. These words are a good indicator of the topics that I would extract from the transcripts using LDA.

Top 10 keywords, by specialty





2c. Algorithms and Techniques

After I completed my exploratory analysis, I used two algorithms to analyze the text topic models. I used Latent Dirichlet Allocation (LDA) to cluster the transcripts according to similar topics, and to extract the keywords associated with them. I used multiclass classification to determine the best number of topics, and to confirm that the topics were well-chosen. In the following paragraphs I describe these techniques.

Latent Dirichlet Allocation (LDA)
Topic modelling refers to the task of identifying topics that best describes a set of documents. These topics will only emerge during the topic modelling process (therefore called latent). And one popular topic modelling technique is known as Latent Dirichlet Allocation (LDA). LDA imagines a fixed set of topics. Each topic represents a set of words. The goal of LDA is to map all the documents to the topics in a such a way that the words in each document are mostly captured by those imaginary topics.

LDA is based on word frequencies and topics distribution in texts. To put it simply, this method counts words in a given data set and groups them based on their co-occurrence into topics. Then the percentage distribution of topics in each document is calculated.

It's common to visualize the LDA topics for interpretability using a popular visualization package, `pyLDAvis` which is designed to help interactively to interpret the results of the model. It can help with better understanding and interpreting individual topics, and better understanding the relationships between the topics. It can also help the user learn about how topics relate to each other, including potential higher-level structure between groups of topics.

Multi-label classification
Multiclass problems involve classifying something into one of N classes (e.g. “red”, “white” or “blue”, etc). It involves a classification task with more than two classes; e.g., classify a set of images of fruits which may be oranges, apples, or pears. Multi-class classification makes the assumption that each sample is assigned to one and only one label.

There are a number of possible classification metrics that I could have chosen. Ultimately, I decided to use multinomial naive bayes because it performs well with a multiclass problem like the one at hand.

Typically, accuracy is a metric used to examine a multiclass classification model. With imbalanced classes, it’s easy to get a high accuracy without actually making useful predictions. So, accuracy as an evaluation metrics makes sense only if the class labels are uniformly distributed. I felt that in my case, the topics were sufficiently well distributed to justify accuracy as an evaluation metric.

2d. Benchmark

I had a couple of benchmark models. The first, and simplest, were just the keywords that were included with the dataset, and associated with each of the medical specialties. A medical professional could use these keyword groups to conduct a simple classification of text topics, by simply searching for keywords. However, this is a simplistic method that is not likely to produce well-defined groups and which lacks precision. Any machine learning model would be an improvement over a keyword search method.

The second, and more important benchmark, was my first LDA model. There are several hyperparameters which one can modify in an LDA model, but the most signficant is the number of topic clusters. I arbitrarily chose to have 10 topics in my first benchmark model. I then examined the clusters, and they seemed to make sense and hang together. I also took a look at the perplexity score,

the coherence score, and the classification accuracy. This provided a baseline from which to examine other hyperparameter settings, in an effort to determine the optimal model.

3a. Data Processing

Preprocessing

It was necessary to preprocess the text data before I could conduct LDA. The first step of any NLP analysis is always text preprocessing. Text preprocessing transforms text into a more digestible form so that machine learning algorithms can perform better.

To preprocess text simply means to bring it into a form that is predictable and analyzable for the machine learning task. The LDA algorithm - or any other ML model - requires numeric format, not raw text. We must convert the text into a numeric vector. Generally, there are 3 main components to text preprocessing: Tokenization, Normalization, and Noise removal.

Tokenization is about splitting strings of text into smaller pieces, or “tokens”. Paragraphs can be tokenized into sentences and sentences can be tokenized into words. Normalization aims to put all text on a level playing field, e.g., converting all characters to lowercase. Noise removal cleans up the text, e.g., remove extra whitespaces or punctuation. From there, before I could dig into analyzing, I had to do some cleaning to break the text down into a format the computer could easily understand.

Of these steps, tokenization is the most important and cannot be skipped. It is the process of splitting the given text into smaller pieces called tokens. Words, numbers, punctuation marks, and others can be considered as tokens. I used the natural language tool kit (NLTK) library for tokenization.

As a part of normalization, it is important to remove stop words. Stop words are a set of commonly used words in a language. Examples of stop words in English are “a”, “the”, “is”, “are” and etc. By removing low information words from text, we can focus on the important words instead. It is possible to remove stopwords using NLTK.

Vectorization
Vectorization is the process of converting text data into numeric data using a vector which captures some information about the text itself. There are several methods of doing this, such as Count Vectorizer, TF-IDF vectorizer, and Bag of Words (BoW). Count Vectorizer is a simple count of word appearances. TFIDF, short for term frequency-inverse document frequency, is a numerical statistic that is intended to reflect how important a word is to a document in a collection or corpus. I chose to use the last method, Bag of Words.

Bag of Words (BOW) is a method to extract features from text documents. These features can be used for training machine learning algorithms. It creates a vocabulary of all the unique words occurring in all the documents in the training set. BoW is a collection of words to represent a sentence with word count and mostly disregarding the order in which they appear.
BoW creates a vocabulary from all the words in a document, with their respective word count, which will be used to create the vectors for each of the sentences.

Once the vectorizer is complete, each row of text is converted to a vector. The length of the vector will always be equal to the vocabulary size - i.e.,. the list of words in the corpus of texts. A big document where the generated vocabulary is huge may result in a vector with lots of 0 values. This is called a sparse vector. Sparse vectors require more memory and computational resources when modeling. The vast number of positions or dimensions can make the modeling process very challenging for traditional algorithms.

Using the `gensim` method `doc2bow` I converted the medical transcripts to a sparse matrix of vectors that capture the number of times each word in the vocabulary appeared in each document. The code was as follows:

```
corpus = [dictionary.doc2bow(doc) for doc in docs]
```

A bag-of-words is a representation of text that describes the occurrence of words within a document. It involves two things:

- A vocabulary of known words.
- A measure of the presence of known words.
  It is called a “bag” of words, because any information about the order or structure of words in the document is discarded.

3b. Implementation

After experimenting with a couple of options, including the AWS SageMaker LDA algorithm, I chose to use the `gensim` version of the LDA algorithm for my purposes. LDA builds a topic per document model and words per topic model, modeled as Dirichlet distributions. The code to build an LDA model with 6 clusters is as follows:

```
# build a model with 6 topic clusters
lda_model = models.ldamodel.LdaModel(corpus=corpus,
         id2word=dictionary,
         num_topics=6,
         eval_every=10,
         passes=50,
         iterations=5000,
         random_state=np.random.RandomState(15))
```

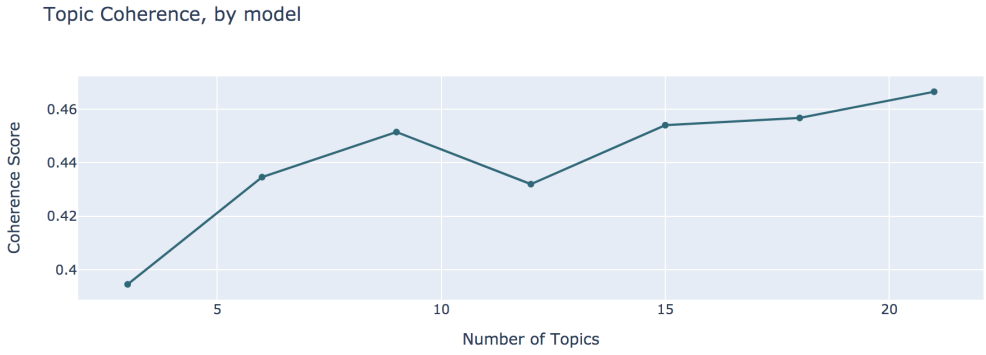
Once the gensim model was complete, for every transcript it produced a probability of belonging to each topic cluster. The cluster with the highest probability is the one that the transcript gets assigned to. The next step after creating the clusters is to assign a new column to the dataset, with the predicted primary topic associated with each transcript.

3c. Refinement

My baseline, benchmark model had a coherence score of 43% and a perplexity score of -7.376. These numbers are not useful by themselves, but when compared to a series of models, they helped me to choose the optimal number of clusters for the final model.

Using a python forloop, I iterated through 7 different models to find the optimal number of topic clusters. I compared each model using accuracy and coherence score. Below are the graphics displaying the comparison of the 7 models.

Coherence scores



Accuracy scores

The full results of the model comparison are available on my final website, here: http://sagemaker-lda-medical.s3-website-us-east-1.amazonaws.com/

The code associated with the refinement of the model is available on my github repo here: https://github.com/austinlasseter/sagemaker_lda_medical/blob/main/surgery-lda-coherence.ipynb

Ultimately I found that the best model had 9 topics. This model had a coherence score of 45% and a classification accuracy of 73%, which outperformed any of the other models.

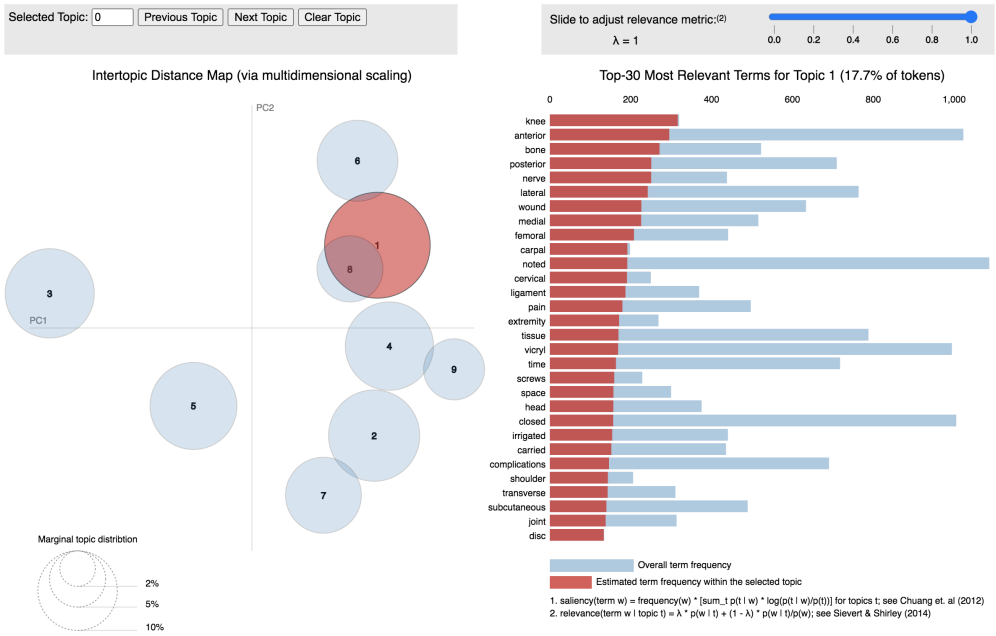
4a. Model Evaluation and Validation

The results of the final model are available on my website, here: http://sagemaker-lda-medical.s3-website-us-east-1.amazonaws.com/

There were ultimately 9 topics in the final model, all drawn from the Surgery specialty of transcripts. Below is an image of the first and largest topic, which includes words like "knee", "bone", "femoral", and "carpal" -- suggesting that most of the transcripts in this topic related to joint and leg surgery (a common procedure). It's notable that nearly all transcripts with the word "knee" (about 250) are clustered into this topic

Likewise, topic 2 includes keywords such as "suture," "tube", "needle," and "vicryl" (which is a rebsorbable, synthetic suture). These words indicate elements associated with a wide variety of surgery procedures, and were indicative of transcripts associated with topic 1.

Explanation of the pyLDAvis results



What stands out to me about the final model is the fact that the topics can be easily explained in human terms. For example, Topic 3 seems to have words associated with heart surgery: it includes words like "artery", "catheter", "coronary", "vein", and "aortic". About 900 of the 1000 transcripts with the word "artery" fall into this topic.

4c. Justification

The final model includes seven hyperparameters. As mentioned above, the most important of these is the number of clusters. After an analysis of 7 different model options, including the benchmark, I determined that 9 clusters was the optimal model. The other hyperparameters -- such as the number of passes and iterations -- I kept at the default values from the `gensim` package as these performed adequately. The final model specifications are listed below.

```
# compute the final model
final_model = models.ldamodel.LdaModel(corpus=corpus,
         id2word=dictionary,
         num_topics=9,
         eval_every=10,
         passes=50,
         iterations=5000,
         random_state=np.random.RandomState(15))
```

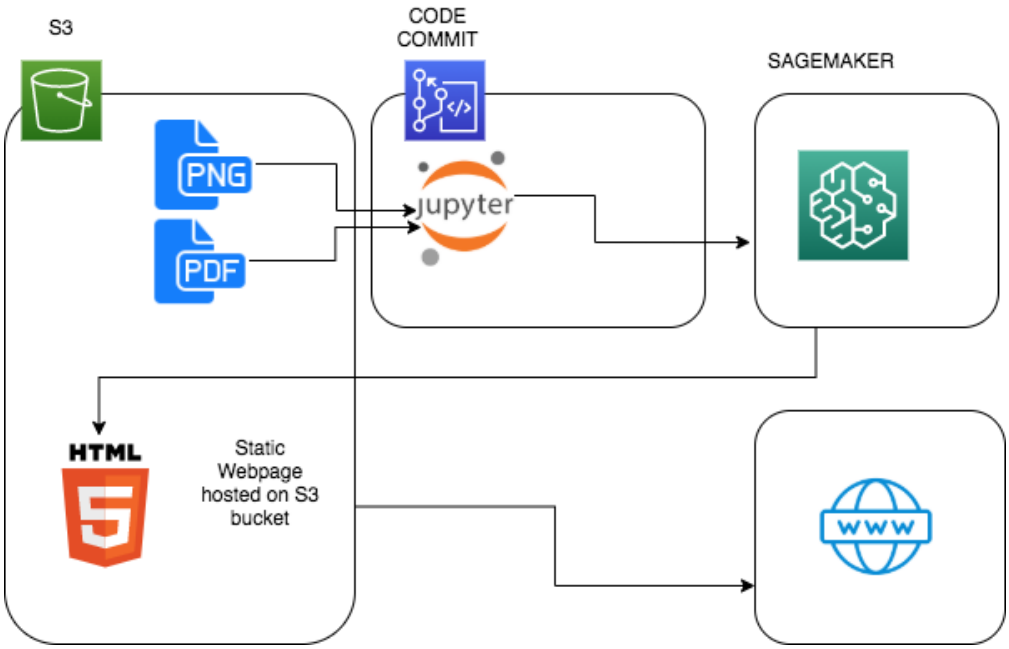
As mentioned above, the final model outperformed the benchmark model (6 topics) with regard to accuracy and coherence. The third metric was perplexity, and both models performed about the same on this metric. Classification accuracy of course is only one possible measure of the multiclass model's performance - it would also be important to look at the confusion matrix, ROC AUC curve or F1 score - but accuracy was sufficient for determining the optimal number of clusters. If I were to go further and use this model to classify additional transcripts as they came in, it is likely that I would explore other classification algorithms than naive bayes. A neural network or XGBoost would be strong choices for future classification of text data.

Ultimately, the LDA model as presented solves the original problem, which was the need to rapidly sort unlabeled transcripts into topic clusters. This was an unsupervised machine learning problem, as the transcripts had no indication, other than their text, about which surgery-related topic they should belong to. The summary of each topic, provided by the top keywords, would allow a medical practitioner to quickly review and summarize large numbers of transcripts, adequately sorted by the LDA model.

5. Conclusion

Once the model was complete, the last step was to deploy the finished model to the internet so that endusers can make use of it for further analysis. It is not sufficient to just have the results on my own machine, as they are not usable by others. The model should be easily accessible to outside users.

To this end, I used Amazon Web Services to deploy the finished model. First, I saved my graphics as html files. Then I incorporated them into a master html file called `index.html`. All of these resources I then saved to an S3 bucket on AWS. The final step was to change the S3 settings to 'public' and then to host a static website with the S3 bucket as its endpoint. Below is a graphic showing the workflow of the project, leading to final deployment.



The finished website is availabe, and the enduser can explore the results of the LDA model here: http://sagemaker-lda-medical.s3-website-us-east-1.amazonaws.com/

In conclusion, medical practitioners have a need to explore the text data associated with written transcripts of surgeries and other medical procedures. Often the information they need is difficult to access because it is stored in text format, not easy to analyze. Natural Language Processing is an ideal solution for this problem, converting text data to a format that can be analyzied using machine learning. In this analysis, I used the LDA method to vectorize text and cluster unlabeled medical transcripts into topic clusters, thus reducing that manual labor required to analyze these transcripts. The possible contributions of machine learning to medical data analytics are endless.