Investigation into Automated Topic and Speaker Identification for Clinical Communication

by

Rachel Fong

Submitted to the Department of Electrical Engineering and Computer Science

May 17, 2012

In Partial Fulfillment of the Requirements for the Degree of

Bachelor of Science in Electrical Engineering and Computer Science

Contents

1 Background

Many research studies show that better physician-patient communication is associated with positive outcomes, including better medication adherence, better health outcomes in chronic conditions, improved patient satisfaction, and reduced hospital readmissions. However, relatively little research focuses on specific improvement of physician-patient communication. Research in physician-patient communication is tedious and costly, usually involving audio recording, transcription, and analysis.

Verbal communication relies on a large body of shared knowledge, or 'common sense' knowledge. Many difficulties in natural language processing can be attributed to the fact that it is difficult for computers to acquire and utilize such knowledge as well as apply it toward processing documents.

2 Goals

Automation is an important step in reducing the cost of analyzing physician-patient communication. The first step toward this goal is giving a computer the context and 'common sense' necessary to understand such communication.

The overarching goal, to be continued by an MEng student, is to be able to automatically identify speakers and topics in transcriptions of physician-patient

conversations.

The scope of this particular project, however, is to conduct preliminary research on the corpus and begin constructing the code framework necessary for the following tasks.

2.1 Topic identification

We will need to perform statistical analyses of the data and look for indicative patterns. Our findings at this stage will help us determine our approach. Sentiment analysis will be necessary to identify emotions and other classifiers relevant to evaluating physician-patient communication; the software described below in section 4.1 is capable of this.

Once we have obtained automatic analyses, they can be evaluated against results from existing manual analyses, which adhere to tagging systems for medical communication, outlined below in section 3.

2.2 Speaker identification

The CASES system outlined below in section 3 specifies conceptually (though not programmatically) low-level cues meant to assist in manual analysis. Those guidelines will help us construct a Markov model to identify speakers. Many of these cues rely on having knowledge of the current conversational threads and topics.

3 Corpus

We use a corpus of 50 transcribed conversations between physicians and patients in HIV care, with identifying details such as names and specific locations anonymized. The conversations have been manually divided into utterances, analyzed, and tagged, using two encoding systems for medical communication, both designed by Bart Laws.

GMIAS (Generalized Medical Interaction Analysis System)

This system is designed to encode topics that are likely to come up in medical conversations, such as prescriptions, physical exams, health insurance, and side effects. It also encodes 'speech act' functions of utterances, such as requests for clarification, demonstrations of empathy, compliments, expressions of preference, etc. Most such systems only assign a single encoding to each utterance, but GMIAS assigns both a topic code and a speech act code, making it possible to compare similar interactions across different topics.

CASES (Comprehensive Assessment of Clinical Encounters)

This system, a complement to the GMIAS, is designed to follow biomedical threads in physician-patient conversations. It designates speakers, and distinguishes between distinct parts and categorizations of conversational threads, such as small talk, resolution of a

thread, and informational exposition.

4 Technologies and Tools

The majority of the code for this project was written using Python and its standard libraries. In addition, we used many open-source toolkits which were invaluable in helping us analyze our data.

4.1 Open Mind Common Sense

The Open Mind Common Sense (OMCS) project, developed by the MIT Media Lab, attempts to address the issue of common sense knowledge acquisition by collecting knowledge in multiple languages through crowdsourced games and the semantic web, and processing the information into ConceptNet, a semantic network of concepts, relations, lexical knowledge, and other obtainable information.

AnalogySpace is a reduced-dimensionality matrix representation of ConceptNet which reveals large-scale patterns in the data, smooths over noise, and predicts new knowledge.

Luminoso is a toolkit and interface which enables users to explore patterns in natural language data using the OMCS corpus, the ConceptNet space, and AnalogySpace.

Divisi2 is a sparse SVD toolkit designed for use with semantic networks. Since it uses ConceptNet data, it can help find related concepts, features, and relation types.

The OMCS tools will immensely simplify the task of building a system backed by real-world knowledge and the capability to analyze conversational topics.

4.2 NLTK

NLTK is an open-source collection of Python modules, linguistic data, and documentation on natural language processing and text analytics. It will be useful for exercising a wide variety of statistical NLP methods.

5 Methods

5.1 Initial inspection

We first formatted and sanitized the corpus. To give us ideas on how to proceed, we parsed the documents using NLTK and generated word frequency distributions of different parts of the corpus as partitioned by speaker, by GMIAS tag, and by CASES tag.

5.2 Common sense inference

We can analyze our corpus in the context of ConceptNet's broad knowledge base by using Divisi2 to project onto spaces representing particular concepts in ConceptNet. We begin by trying out several groupings of similar and opposing concepts described in the GMIAS speech act codes: for example, happy, joy, and love versus sad.

5.2.1 Scoring algorithm

In order to consider related chunks of conversation while accounting for variation in conversational topics, we analyze a rolling window of groups of utterances, feeding the terms in each window through Divisi2 in order to evaluate their relatedness to the concepts we chose.

Note that we use both the positive and negative concepts in a grouping to generate one score; by summing the vectors corresponding to the positive concepts, and subtracting the sum of the vectors corresponding to the negative concepts, we obtain more foolproof results than if we had only used one concept.

```
1  global conceptnet = matrix representing ConceptNet
2  global concepts = list of concept groups, consisting of two lists of opposing concepts

SCORE(utterances, windowSize, concepts)
3  weights = count term frequencies for term in utterance in utterances
4  // normalize and sqrt
5  maxFreq = max( weight.values )
6  for term in weight.keys
7        weight[term] = sqrt( weight[term] ) / sqrt( maxFreq )
```

```
// rolling window
scores = [ ][ ]
for i in [0, utterances.length - windowSize]
    window = aggregate terms in utterances[i ... i+windowSize-1]
    windowVec = zero vector
    for term in window
        if term is a valid ConceptNet concept
            windowVec += conceptnet.get_row(term) * weight[term]
    for c in concepts
        scores_{window,c}.push( SCORE-HELPER(windowVec, c.pos, c.neg) )
    window: dequeue first utterance, push next utterance
```

SCORE-HELPER($windowVec, posConcepts, negConcepts$)

```
conceptVec = zero vector of same length as a ConceptNet vector
for concept in posConcepts
    conceptVec += conceptnet.get_row(concept)
for concept in negConcepts
    conceptVec -= conceptnet.get_row(concept)
return conceptVec · windowVec
```

5.2.2 Peakiness

We would also like to optimize the window size so that the 'peakiness' of the graph of feature scores is maximized. Peakiness is a common concept in spectrum analysis. Given the following:

$maxima_0$ = highest maxima

$maxima_1$ = next highest maxima

$minima$ = lowest point between max_0 and max_1

We define peakiness as follows, using the $scores_{w,c}$ computed above:

$$P_{w,c} = \frac{min[scores_{w,c}(maxima_0), scores_{w,c}(maxima_1)]}{scores_{w,c}(minima)} \quad (1)$$

The peakiness algorithm is straightforward; below is a rough sketch of our peak-finding algorithm.

FIND-PEAK-INDICES(L)

```
peaks = [ ]
seenPosDeriv = FALSE
for x, index in L
    if seenPosDeriv
        if the current derivative is negative
            peaks.push( index-1 )
            seenPosDeriv = FALSE
    elseif current derivative is positive
        seenPosDeriv = TRUE
    if x is last in L, and the current derivative is positive
        peaks.push( index )
return peaks
```

5.3 Code capabilities

The corpus and concepts are stored separately and can be easily changed as long as they are formatted as expected. The code is fairly modular, allowing different heuristics such as peakiness, term weights, and window ranges to be changed very easily.

6 Results

6.1 Cursory overview from frequency analysis

A subjective overview was suggestive for the 30 most frequent words in the speaker categories, including generic words ('the', 'a', 'and', 'is') and punctuation, which we ignored. We also noted that doctors provided 35% more utterances than patients.

High-frequency patient words

- words of affirmation: yeah, okay
- 3 first person references (I, me, my) vs. 1 second person reference (you)

High-frequency doctor words

- 3 second person references (you, your, you're) vs. 1 first person reference (I)

The majority of the (human-tagged) topic categories have at least twice as many non-generic words, although the small sample sizes resulted in much lower frequencies showing up in the top 30, and thus less obvious trends. The highest frequency in each category ranged from roughly 5 to 600, as opposed to the doctor and patient categories, where the top 30 frequencies ranged from 641 to 4597 and 391 to 4329, respectively.

6.2 Feature scores

We ran the scoring subroutines to obtain scores for each concept, for each window. We outputted the results to a series of CSV files for later analysis.

6.3 Window size optimization

For several groupings of opposing concepts relevant to GMIAS, we optimized the peakiness function by varying window size, sweeping through the range of possible window sizes with a step size of 5 utterances. Improvements could easily be made by making another sweep near the best result, but with a step size of 1. We obtained the following approximations to optimal window sizes, bearing in mind that our heuristic may need tweaking to more ideally serve our end goal:

+	-	window size
happy, joy, love	sad	5
empathy	apathy	5
hope	despair	10
agree	disagree	10
compliment, praise	criticize	5
calm	concern, fear, anxiety	115
comfort	pain, discomfort	125
surprise	awe	5
satisfaction		5
	frustration, anger	115
	annoyance	10

The data suggests that certain emotions are likely to be present throughout, while

others fluctuate more rapidly.

7 Conclusions and Future Work

Since GMIAS and CASES are both fairly expansive tag sets, we did not expect to be able to fully achieve topic and speaker classification within the scope of the semester. However, we have developed a solid direction for the project to move in, written code to analyze the corpus, and obtained data relevant to classification.

References

[1] Catherine Havasi, Ira Wilson, and Carolyn Rose. "Understanding Doctor/Patient Dialogue: Phase One." 2011.

[2] Bart Laws. "CASES Coding System Manual." 2011.

[3] Bart Laws. "Speech Act Codes." 2009.

[4] Bart Laws. "Topic Act Codes." 2009.