SMS based Natural Language Interface for Locating Health Care Service Providers

Rohit Banga, Akhil Langer, Ankush Mittal, Parikshit Sondhi

Abstract:
In this paper we present an implementation strategy for an automated system to address the problem of locating a health care provider. The solution that we propose is designed to strengthen and complement the weak health systems existing at the time of this writing. The motivation for solving this problem comes from the absence of an automated service to obtain the details of a doctor. To locate a doctor for a specific health condition, people currently use ad hoc approaches. While this kind of information can be readily searched for using the Internet, this medium does not reach the masses. Also there are very few websites offering this service in India. In contrast, a common medium of communication that can be leveraged for this problem is the mobile phone which offers SMS as an inexpensive channel of sharing information. The challenge that we foresee in such a situation is the ability of the system to respond to natural language queries. We have developed a collection of techniques for addressing these problems. The design approach is guided by the expected usage of the system by a typical user in India and the use of a mobile phone.

1. Introduction

The Health Care Sector in India is largely unorganized. The people in villages thus have inadequate access to health services. Additionally, the number of physicians per 1000 people is just 0.6 in India as compared to 2.3 in United States (1). Therefore, in order to better equip people with existing health facilities we need to dovetail the use of current technological advancements to the health care sector.

The recent trends in telecommunications sector indicate that cheap mobile phones have reached even remote areas of the country. The total number of wireless subscribers in India stood at 362.3 million at the end of January, 2009 (2) which comprises about 30% of the population. About 8 million mobile phone subscribers are added every month (3). This indicates the presence of a good communication channel which can be leveraged by novel applications to achieve good penetration and usage.

This paper deals with the design of one such application utilizing the Short Messaging Service (SMS) for providing information about health care service providers.

The system that we have designed accepts natural language queries as input from which we extract parameters of interest that help us to rank the doctors.

To summarize, the contributions of our paper are as follows:
1. To the best of our knowledge, the system is the first to map health problem descriptions to health care specializations using natural language queries as input.
2. The system is tolerant to common spelling errors.
3. The system uses existing geo-coding services to map an address to a position on the map and hence uses it to find the closest doctor.
4. We present a simple ranking function for selecting doctors based on the various input terms in the query.
2. Prior Work and Existing Systems

For locating a doctor people currently use ad hoc approaches. For example through the
information learnt from the social network of the person or through services like yellow
pages. Access to such information is best achieved through the use of systems which
inherently involve humans answering questions in the backend (4).

There are a large number of websites on the Internet providing a form based interface for
finding doctors. A typical example is Revolution Health (5). This website requires the user
to specify the location in one of the fields (example – Austin, TX) and an option to find a
doctor by specialty or by health condition. In the first case the user is presented with a
drop down list to select the specialization of the doctor required (example – Orthopedics,
Otology). When searching by treatment or condition the user is presented with two drop
down lists to further narrow down the search conditions. The interface also allows the user
to optionally specify the last name of the doctor. A similar functionality is provided by Web
MD (6). Moreover, this website caters to US users only. An initiative for doctors in India
has been started by Healthcaremagic (7). However, it allows searching for doctors based
on specialty only.

Clearly, the drop down menu based interface adopted by these websites is unsuitable for
mobile phones and particularly SMS which do not provide UI interfaces like drop-down
menus. Logical extensions of the method to a multi-message selection procedure are both
inconvenient, time consuming and needlessly expensive for the end user.

A user would usually prefer a doctor near her location. The service provided by (5) expects
the user to enter her pin code or specify the city and state. It is currently available for
locations in US only. Healthcaremagic (7) allows a user to filter the results based on state
and city and is available for India. These services do not allow a user to search at a finer
level except when a pin code is provided. However a user may not be too sure of the pin
code.

A search system for locating doctors in particular or local businesses in general from Yellow
Pages can provide ready access to this information. Such a system based on natural
language queries has been discussed in (8). However this system is meant for locating
exact matches. Thus a user may use this system for getting the contact details of a doctor,
locating any doctor in a locality but may not be able locate a doctor in a nearby locality if
there is none in the locality specified in the query. However this system does claim to map
natural language words to the kind of business that one is looking for.

Automating this process provides scalability in terms of the number of people the system is
able to support.

3. Typical Use Case for our System

Since majority of the people in India still use cheap handsets with limited features and
providing information about doctors requires only text, therefore we have chosen SMS as
our medium.

We have designed the system keeping the following use case in mind (Error! Reference
source not found.).
4. System Architecture

The SMS query undergoes the following transformation steps as it passes through the system:

1. The query is passed through three Information Extractor modules which determine location related information, medical terms and doctor names occurring in the query. There is no flow of data among these modules. This classification is not deterministic. There may be unclassified terms (not belonging to any category), ambiguous terms (belonging to more than one category).
2. Using the results of Step 1 we apply various heuristics to resolve the ambiguities and classify the remaining terms to the extent possible.
3. In Step 3 the heuristics are used to reformulate the query with the location related terms removed.
4. The query is reevaluated against the medical terms extraction module and doctor names extraction module.
5. The system generates several possible interpretations of the query.
6. An interpretation is chosen and is used to rank the doctors based on the ranking function discussed in Section Error! Bookmark not defined.
7. A ranked list of doctors for this interpretation is sent to the caller which may choose to rerank the doctors for some other interpretation from among the ones in Step 5 above.

5. Implementation

In this section we detail the design of the components outlined in Section Error! Reference source not found.. Central to our system is the idea of being able to search for various keywords. For this task we have used Apache Lucene (9). Lucene is an open source software library for keyword based text searching applications. In section Error! Reference source not found. we briefly review the basics of Lucene. Sections Error! Reference source not found. and Error! Reference source not found. discuss the Specialty Mapper, Doctor Names Extraction Module and Location Extraction Module respectively. In Section Error! Reference source not found. we describe our algorithm for parsing queries and utilizing these knowledgebases. Section Error! Reference source not found. discusses the ranking function for selecting the best doctors for a query.

5.1 Lucene
Lucene is a high-performance java open source text search engine API. Working with Lucene requires us to build an Index. Index is a set of files in a particular format that stores data stored in it. Writing to an Index involves adding Document objects. All data that has to be stored in a document is stored as chunks of Field objects. A field is a name/value pair (10) along with certain properties that indicate how to store and access the fields in the index. For Example, to create an index of all books in a library one can create a document object for each book. Documentation on Lucene can be found at (11).

5.2 Specialty Mapper
We started with the goal of being able to map problem description to a health care specialization. For this we need a knowledgebase and this subsection outlines our strategy for creating one. We used a combination of two sources for collecting terms for our knowledgebase. The first source is the MeSH (Medical Subject Headings) hierarchy and the other is an unstructured collection of sentences for a doctor specialty.

5.2.1 MeSH hierarchy
MeSH is a controlled vocabulary for the purpose of indexing journal articles and books in the life sciences. It has been created by United States National Library of Medicine (NLM) (12). It can also serve as a thesaurus that facilitates searching. The medical terms are organized in the form of a hierarchy (tree). Each node in the tree has a descriptor or subject heading. A given descriptor may occur at several places in the hierarchical tree. The tree locations carry systematic labels known as tree numbers and consequently one descriptor can carry several tree numbers (13). The hierarchy is semantically organized. Consider the examples of MeSH nodes given in Error! Reference source not found.. A part of the MeSH subtree is shown. It is important to note that only one instance of the Node in the MeSH tree is indicated in the table. Each node may occur at multiple places in the hierarchy and corresponding to each occurrence there is a tree number for the node
Blister (A) and eczema (B) have skin diseases as the least common ancestor.
Conjunctivitis (D) and Myopia (E) have eye diseases as the least common ancestor.
The tree number has a dotted notation corresponding to the hierarchical structure. Thus (D) and (E) have C11 as a common prefix for the tree number which happens to be the tree number for eye diseases. Similarly (A) and (B) have C17.800 as the common prefix. 'C' denotes the diseases ancestor, 'C17' – skin diseases, 'C17.800' – skin and connective tissue diseases.

<table>
<thead>
<tr>
<th>Sl. No.</th>
<th>Node</th>
<th>Ancestors of Node</th>
<th>Tree Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>Blister</td>
<td>“skin diseases, vesiculobulous” -&gt; “skin diseases” -&gt; “skin and connective tissue diseases” -&gt; “diseases”</td>
<td>C17.800.865.187</td>
</tr>
<tr>
<td>B</td>
<td>Eczema</td>
<td>“dermatitis” -&gt; “skin diseases” -&gt; “skin and connective tissue diseases” -&gt; “diseases”</td>
<td>C17.800.174.620</td>
</tr>
<tr>
<td>C</td>
<td>Nose</td>
<td>“head” -&gt; “body regions” -&gt; “Anatomy”</td>
<td>A01.456.505.733</td>
</tr>
<tr>
<td>D</td>
<td>Conjunctivitis</td>
<td>“Conjunctival diseases” -&gt; “eye diseases” -&gt; “diseases”</td>
<td>C11.187.183</td>
</tr>
<tr>
<td>E</td>
<td>Myopia</td>
<td>“refractive errors” -&gt; “eye diseases” -&gt; “diseases”</td>
<td>C11.744.636</td>
</tr>
</tbody>
</table>

Table 1 Examples of MeSH Nodes

We have utilized this structure in the MeSH to build our knowledgebase. We manually created rules to achieve this. Examples of such rules are given in Error! Reference source not found..
<table>
<thead>
<tr>
<th>Category</th>
<th>MeSH Number</th>
<th>Specialization</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Back</td>
<td>A01.176</td>
<td>Orthopedic</td>
<td>lumbosacral region</td>
</tr>
<tr>
<td>calcium metabolism disorders</td>
<td>C18.452.174</td>
<td>Orthopedic</td>
<td>osteomalacia; rickets</td>
</tr>
<tr>
<td>orthopedic procedures</td>
<td>E04.555</td>
<td>Orthopedic</td>
<td>amputation; bone transplantation</td>
</tr>
<tr>
<td>eye diseases</td>
<td>C11</td>
<td>Ophthalmologist</td>
<td>conjunctivitis; cataract; myopia; vision, low; iris</td>
</tr>
<tr>
<td>Eye</td>
<td>A09.371</td>
<td>Ophthalmologist</td>
<td>cornea; eyelids; eyelashes; pupil; retina</td>
</tr>
<tr>
<td>vision disorders</td>
<td>C11.966</td>
<td>Optometrist</td>
<td>blindness; vision, low; color vision defects</td>
</tr>
<tr>
<td>respiratory tract diseases</td>
<td>C08</td>
<td>Pulmonologist</td>
<td>asthma; bronchitis; pneumonia; lung diseases, fungal</td>
</tr>
<tr>
<td>behavior and behavior mechanisms</td>
<td>F01</td>
<td>Psychiatrist</td>
<td>enuresis; mental fatigue; fingersucking;</td>
</tr>
<tr>
<td>psychological phenomena and processes</td>
<td>F02</td>
<td>Psychiatrist</td>
<td>dreams; ego</td>
</tr>
</tbody>
</table>

Table 2 Example Rules to Classify MeSH nodes

By noting the tree numbers as in Error! Reference source not found., we can ease the task of categorization of medical terms occurring in the hierarchy. There are some things worth noting about this method of classification:

- All MeSH nodes that have a tree number starting with C11 fall in the category of eye diseases and are hence point to an ophthalmologist. But a more fine classification is required for vision disorders occurring in this subtree which point to an optometrist. Thus “vision, low” occurs in both categories as can be seen in Error! Reference source not found.

For classification, we store these rules in an appropriate data structure and scan the MeSH descriptors. We look for the longest prefix match to allow this scenario of fine grained classification. This is possible because of the systematic tree numbers assigned by MeSH.

- Many of the MeSH descriptors are complex medical terms that may not be known by all people. For example – “somnambulism” meaning “sleepwalking” is not a common term. While inclusion of these terms might be useful to medical professionals who have the required knowledge, it is better to include the common word “sleepwalking” also so that it is useful for the common man as well. We thus add the definition of these medical terms by looking up Wordnet (14) along with the term.

5.2.2 Unstructured Definitions

MeSH is a controlled vocabulary and thus it does not contain certain commonly used medical terms. As an example “dandruff” does not occur in MeSH. To make the knowledgebase more complete, we also populate it with the description of medical
specialty. This unstructured text allows us to add some common words that are not found in the MeSH hierarchy. The description has been obtained from Wrongdiagnosis.com (15).

5.2.3 Addition to Index
As noted earlier we use Lucene to store the data of the knowledgebase. We create a Lucene Document corresponding to each doctor specialty. This document is populated with the following fields –

a. Title – Doctor specialty
b. Description – the text prepared for the specialty as described in Error! Reference source not found..
c. Node – MeSH nodes that have a corresponding rule defined as illustrated in Error! Reference source not found..

Preprocessing Text – Terms that are added to the index must be preprocessed to make the index case insensitive, independent of different forms of the same word and unaffected by stop words like is, are, a, the etc. Lucene allows us to pass the text through an analyzer (16) to perform these functions. To treat different forms of the same word as equivalent (For e.g. alcoholism and alcohol have the same word root and hence should be treated as similar) we use Porter Stemming Algorithm (17). The Porter Stemmer Filter is provided with Lucene. Similarly a stop word filter is also provided with Lucene to remove words that have no effect on determining the kind of specialist that is required.

5.3 Doctor Names Extraction Module
To test our system we have obtained a database of doctors’ details from (7). We store the information of doctors also in a Lucene Index. In this case a document contains the information of a single doctor. The fields that we have included are – name, address, qualification, city, state, latitude, longitude etc. Although not all fields are of interest and indexed. We focus are search on the name field.

We utilize Lucene's fuzzy query feature to achieve approximate matching of names. Thus misspelled names are automatically matched to the nearest match. Another problem with matching names is that the same name may be specified in different forms. For example – Dr. Anuj Kumar Patel may be specified as Dr. AK Patel, Dr. A Patel, Dr. Patel etc. We add all these forms as synonyms to the original term. This feature is also implemented using Lucene.

5.4 Location Extraction Module
It is natural for a user to prefer a doctor that is near to her location. The database of doctors contains the addresses of doctors. To measure nearness of two addresses, we need to encode these addresses in a mathematical form. Thus we encode all addresses in the doctors’ database in the form of Latitude and Longitudes. This feature is required by many applications and is called geo coding. We have used Google Maps Geocoding API (18) to achieve this. The Google Maps Geocoding API is accessed using HTTP request. If the service is able to find the address in its database, it returns an HTTP 200 OK message otherwise it returns an error code. In addition if the address is valid the service indicates the accuracy of the result rated on a scale of 0 to 9.

In this manner we can obtain the location of all doctors and the location specified in the query in the form of latitudes and longitudes. To find the closest doctor, we perform a range query on the Lucene index which has the fields latitude and longitude to restrict the search over the doctors which are within some radius of the user’s location. For each doctor obtained by this range query search, we find the approximate distance along the earth’s surface by using Haversine’s Formula (19). However, this approach does not take into account the actual distance along the road and we assume this distance is a reliable estimate of the nearness of a place. The formula is given in Error! Reference source not found.Error! Reference source not found.. Using these equations we find the distance, \(d\) between the user and each doctor in the list. We later explain how this distance is used by our ranking function.
However, Google Maps API may not be able to geocode all locations, specially the remote villages in India. Thus, we have built a database of all the villages, sub-districts, districts and towns in India to resolve this issue using the information provided at the website for Census 2001 (20). The latitude and longitude information also needs to be obtained for these locations. An approximate value for a village’s geocoded location can be taken to be the same as that for the sub-district in which the village lies. Thus while using an approximate value; our system can identify the locations of a large number of small and remote villages. This data provides us a way to hint at the presence of names of cities, states, district names. This provides a way to hint at the which terms in the query refer to the location, hence providing a starting point for parsing.

5.5 Parsing the Queries
The query is obtained in the form of a single string. We have to separate the various parameters of interest from the query. The parameters as noted before are –
1. Location of user
2. Medical terms
3. Name of the doctor

The queries are naturally structured. This means that the user can list the above three parameters in any order. But it is reasonable to base our parsing heuristics on the "well formed" queries. Since the motive of the user is to obtain information about a doctor using SMS, the user may not provide complete sentences and only the keywords may be provided.

Although the parameters may occur in any order, the terms of a particular parameter would be present contiguously. Such assumptions lead to the following:

- We give a high importance to the location parameter. The first stage through which our query is passed is Location Extraction Module. This is used to hint the last word contributing to the location information.
- The last word in the query to hit against this result is considered to be the last occurrence of location information in the query. We call this index (no. of the word) as the 'lastLocIndex'.
- To find the 'firstLocIndex' the index of the first word in the query that provides locational information, start with the first word in the query that provides locational information for the first result from among the hits. Words having index smaller than firstLocIndex may or may not contribute to the location information.
- Thus our aim is to find smallest x between 0 and firstLocIndex such that words having index between x and lastLocIndex contribute to location information only.
We scan the list of words from \((\text{firstLocIndex-1})\) down to 0 and apply the following heuristics:

- If addition of a word increases the accuracy of Geocoding reported by Google Maps then it contributes to location information.
- If the word is unclassified it is assumed to contribute to location information.
- If a word contributes to the highest ranking medical specialty then \(x = (\text{index of this word} + 1)\) as it may not be contributing to location information.
- If a word matches with the name of the highest ranking doctor whose specialty is the same as that indicated by the medical terms occurring in the highest ranking specialty then we again assume that \(x = (\text{index of this word} + 1)\) and break out of the loop.
- For example in the query “skin doctor iit delhi” the word “delhi” is identified as location term in the first pass. The word “iit” increases the accuracy reported by Google Maps while the term doctor is considered as a stop word and the term “skin” contributes to the highest ranking specialty hence is not contributing to the location.

- Remove the terms between \(x\) and lastLocIndex from the query and rerank the reformulated query against the specialty mapper and doctor names extraction module. Reranking is required as in our experiments we observed that sometimes words occurring in the address part influenced the ranking of specialty or doctor names. For example in the query “skin doctor subroto park new delhi” the word “park” is ambiguous as it matches with “parkin” a derivative of Parkinson’s disease occurring in the medical terms database. Similarly in the query “skin doctor vikas puri new delhi” the term “vikas” matches with “Dr. Vikas” thus requiring reranking after address identification.

- Then rank the doctors on the basis of distance as explained in Error! Reference source not found..

5.6 Ranking Module

We use the following procedure to rank the doctors.

1. Initially the scores of all doctors is zero.
2. When the query contains the name of a doctor, then that doctor's document would be ranked high. Many doctors might have the same or similar names. Thus there would be multiple hits. But Lucene would assign different scores to each hit. The highest ranking document would receive the highest score. Doctors with same names would be assigned the same score. Normalize these score by dividing each score by the highest score. Thus the scores obtained are on a scale of 1. Call the score obtained as a result of this set of doctors, \(\alpha_1\).

3. Now by performing a range query as explained in Section Error! Reference source not found., obtain a list of doctors within a small radius of the user. Find the distance of all these doctors from the user. The score of these doctors is also normalized as follows –

\[
\text{score}(i) = 1.0 - \frac{\text{dist}(i)}{\text{maxdist}}
\]

The rationale behind this formula is that the doctors which are closer are ranked higher, that is, score of the \(i^{th}\) doctor is more if the distance of the \(i^{th}\) doctor from the user is less. Moreover the distances are divided by the distance of the farthest doctor from the user, so that score is normalized on a scale of 1. Call the score obtained as a result of this set of doctors, \(\alpha_2\).
4. Give a score to all doctors having some specialty indicated in the query normalized on a scale of 1.0 based on the score returned by Lucene during the second pass through the Specialty Mapper. Call the scores obtained as a result of this set of doctors, $\alpha_3$. As the number of doctors with a given specialty may be large, an implementation can choose to focus only on those doctors that have been found in steps 2 and 3 as they are likely to be ranked higher than the others.

5. Find the union of the sets of doctors obtained in steps 2, 3 and 4 above. There final scores are computed by adding scores for their occurrence in either of the sets. The final score $\alpha$ is computed by adding $\alpha_1$, $\alpha_2$ and $\alpha_3$, giving weightage $\beta_1$, $\beta_2$ and $\beta_3$ to these scores respectively. These weights are decided by the following set of rules –

   a. Case 1 –
      All three parameters are present
      Assign a higher weight to location if present in the query. If the location is determined with a high accuracy, then a higher value should be given to $\beta_2$. The accuracy of location is reported as an integer between 1 and 8 with a higher value indicating more accuracy.
      $\beta_2 = \text{accuracy} / 10.0$
      $\beta_1 = (1 - \text{accuracy} / 10.0) / 2$
      $\beta_3 = (1 - \text{accuracy} / 10.0) / 2$

   b. Case 2 –
      Two parameters are present and one of them is location
      $\beta_2 = \text{accuracy} / 10.0$
      $\beta_1 = (1 - \text{accuracy} / 10.0)$ or $\beta_3 = (1 - \text{accuracy} / 10.0) / 2$

   c. Case 3 –
      Two parameters are present and is location is absent
      We reject the query in this case.

   d. Case 3 –
      Only location parameter is present
      Let the weight corresponding to location parameter be 1.0 and the others be zero.
      $\beta_2 = 1.0$
      $\beta_1 = \beta_3 = 0.0$

6. Final scores are calculated using

$$\alpha = \alpha_1\beta_1 + \alpha_2\beta_2 + \alpha_3\beta_3$$

The above scoring function allows us to find sets of doctors independently for each parameter and the weights allow us to combine these sets to obtain a cumulative ranking.
6. Results

- We obtained a data of over 15000 doctors in India from www.healthcaremagic.com.
- To show the advantage of using Google Maps we cut down the list to 361 doctors from Delhi from 30 different specialties.
- We have conducted alpha testing of the system at our end.
- We generated the example queries were generated by on our own.
- The testing has been done on command line program and not on the actual phones.
- We tried various spelling variations to test the approximate matching ability of Lucene.
- We assume that a single response may span several messages sent from the server side.
- Examples of the queries are indicated in Error! Reference source not found., Error! Reference source not found. and Error! Reference source not found..

Fig 5 Example 1 (all three parameters specified)
Fig. 6 Example 2 similar to Example 1 except name not present, notice that result 2 gets ranked higher as no name is specified and the second result in the previous example is closer to the location.
7. Future Work

This preliminary work has given us a lot of ideas not all of which could be incorporated in this project. In this section, we share these with the aim of giving direction to future work in this field.

- The facility can be extended to automatically fix appointments with a doctor that appears in the search result. In such a case, the availability of a doctor must be taken into account. This would require the server to sync itself with the doctor’s public calendar and in case a doctor is unavailable, a substitute doctor can be suggested. The feasibility and usefulness of these features needs to be studied.
- Other parameters can be used for ranking the doctors. For example – rating of doctors, feedback of patients, history of cases handled by a doctor, other preferences specified by a user.
- This paper has focused on the medical domain. In general a system based on the knowledgebase approach as outlined in this work can find businesses by related words. For example the complaint of a “broken pipe” would automatically be referenced to a plumber and so on.
- System must be verified by a medical practitioner before being deployed.
- The system can be extended to other Indian languages as well. The feasibility of this approach and the design changes required would need to be analyzed.
References


