Building Medical corpus for transforming unstructured information to structured information

Authors: Awanish Ranjan and Rabindra Bista

Affiliation: Department of Computer Science and Engineering, Kathmandu University, Dhulikhel, Nepal

E-mail - awa.ran@gmail.com and rbista@ku.edu.np

Abstract: In recent data age, we get a huge amount of data from everywhere. Medical domain is also one of the areas producing a large amount of data related to patient status like diagnosis, procedure, drugs etc. One of the challenges is to extract meaningful information from this rich set of data which is quite unstructured. In order to get meaningful structured information, we need a solid Named Entity Recognition (NER) technique. Building a medical corpus is one way to impose NER in the medical data. This paper discusses about the method of building medical corpus. It also discusses the method to implement the corpus in NER and extract information from the unstructured data.

Key-words: *unstructured texts, structured texts, nature language processing, Named Entity Recognition, Corpus, accuracy*

BACKGROUND: Natural Language Processing (NLP) is quite emerging field of Artificial Intelligence. There are quite a myriad of researches going on for tackling several challenges related to the NLP systems. We'll focus on English as Natural Language for this paper. One of the challenges that we face during NLP implementation is the quite unstructured content of natural languages. Though the English language has definite structures governed by the rules of grammar, in the normal communication, it becomes quite unstructured. Use of abbreviations, ambiguity, not always following the correct grammar rules, incomplete sentences are some of the factors due to which it becomes quite unstructured. Human brains a very smart enough to capture such information and convert into the meaningful information for their use. But it is quite hard for the automated computerized system to extract meaningful information on such unstructured texts.

The main discussion point of this paper is focused on the healthcare and clinical sector where the use of unstructured texts is prevalent. There are several instances where the medical practitioners like doctors, pharmacists and nurses generate such unstructured texts while collecting family history, prescribing drugs and so on. These unstructured texts are rich in clinical information and those needs to be extracted as meaningful knowledge for further processing. Lack of any system to convert those information into structured one makes the medical practitioners' job tedious by needing to read all notes and history each time manually and extracting information from them and again storing it some other place. It consumes a considerable amount of time for them. Its more tedious when there is any transfer between one department of hospital to another like from

emergency to operation theatre. So if there is any system, which extracts information from already entered notes and make them readily available to the physicians whenever they need it, it would save a lot of time and effort. The research about converting the unstructured text to structured information will solve this problem to a great extent. Following is the basic block diagram of the proposed system,

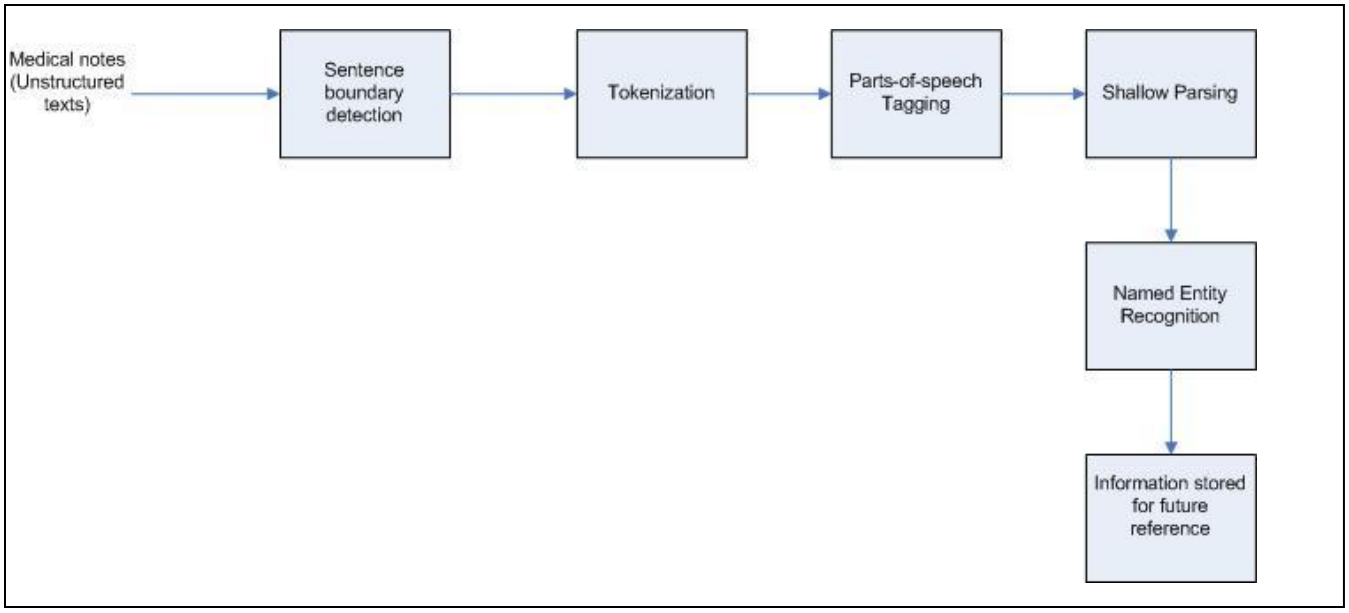

Fig 1: - Bock diagram

In this paper, we are going to discuss about building a medical corpus for medical named entity recognition. Corpus is a large and structured set of texts which is used for the analysis of data.

The sole purpose of the corpus building is to apply it in NER. The idea is to generate a rich tagged set of corpus from the available set of data. This corpus will be then used to tag the unstructured text automatically by the system. For the purpose of this research, we need to identify the entities like Diagnosis, Procedure and Drug. So the corpus is focused around these named entities. The corpus is built to recognize one of the following entities within any note-

Entity Type	Description	Example
Diagnosis	Disease associated with the patient	Diabetes, hypertension, cancer etc.
Procedure	Any procedure done for identification or cure of the disease	MRI, CT Scan, Lab Tests, Therapies etc.
Drug	Medications taken by the patient	Metformin, Lantus, Insulin etc.
Habits	Different habits related to health	Exercise, smoking, jogging etc.
Vitals	Vital signs associated with patient	Weight, height, blood sugar etc.

Table 1:- Entity Types

OBJECTIVES:

A. To collect data in the form of clinical notes.
B. To generate the medical corpus for the use of entity recognition.
C. To handle the redundancy in the corpus generated.
D. To evaluate the performance of corpus generation.

METHODS: Following methods were involved in building medical corpus data –

A. Data Collection
B. Manual Annotation
C. Generating corpus file
D. Redundancy Handling

Detail on each step is discussed is presented below -

A. Data collection – Corpus is generally associated with some domain. Here we are dealing with medical texts so the domain is limited to medical domain. Also the entities that we are interested to recognize are diagnosis, procedures and drug so the data should be rich in this information. For this purpose, we collected a large set of texts entered by nurses during patient visits or patient calls with the details of their diagnosis, procedure, medication, vitals and habit. Around 13k of such notes is used for building the corpus.

B. Manual annotation – After data collection, the second task is manually annotating the notes to tag the relevant texts. The relevant text is tagged within the span of <START:{type}> (Text) <END>.

Entity Type	**Annotation Span**
Diagnosis	<START:diagnosis> <END>
Procedure	<START:procedure> <END>
Drug	<START:drug> <END>
Habit	<START:habit> <END>
Vitals	<START:vital> <END>

Table2:- Entity span

Each text is read manually and put one of these tags for appropriate words with human knowledge as shown below –

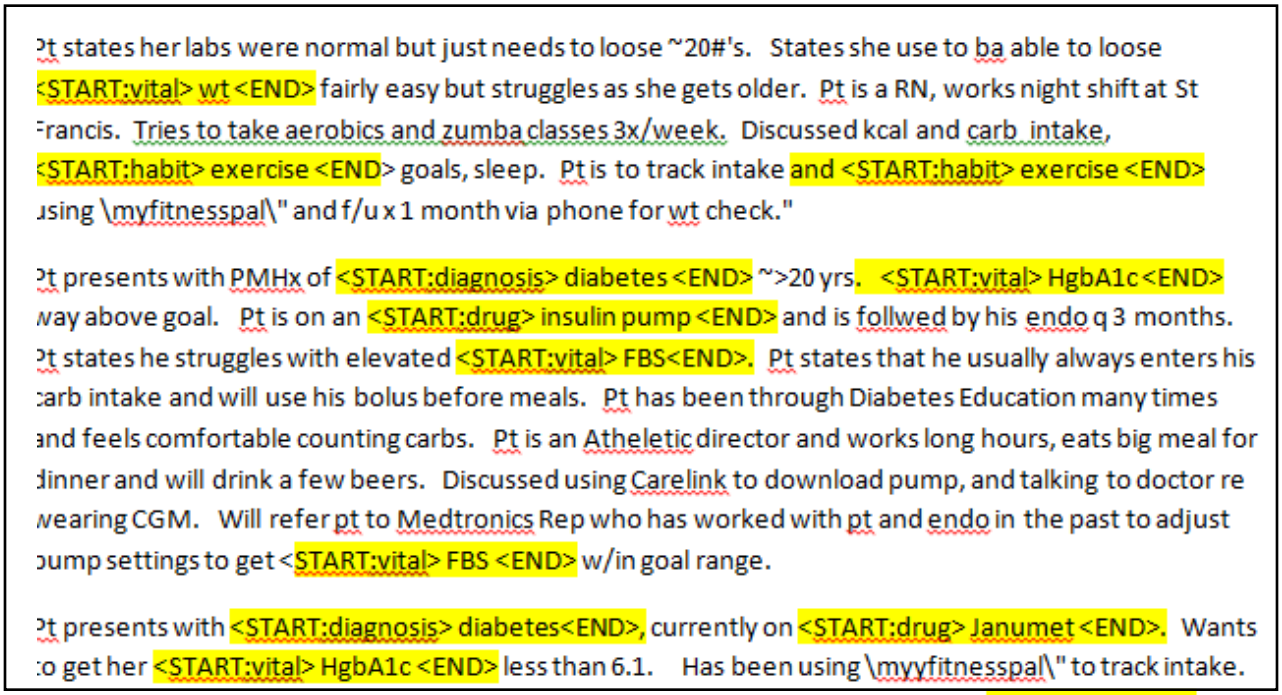

Fig 2:- Manual tagging

C. Generating corpus file- After the human annotation is completed; a computer program is built in order to generate the corpus file. Each type of corpus is saved in its separate file name *{type}.ner*. For example, diagnosis corpus is saved in a file named diagnosis.ner and so on. Following algorithm is used in order to generate the corpus file –

```
1  Corpus Algorightm
2  ----------------------
3  For each line in training file
4      Find nth position of string "<START:corp" where corp ={"habit","procedure","diagnosis","drug","vital"}
5      remaining_line -> line_substring [n:end of line]
6      extract the corpus data
7      corp_data -> substring_of _remaining_line[startpos:endpos]
8      [where startpos = index of ">" in string remainig_line +1, endpos = index of "<" in string remainin_line]
9      remove leading and trailing spaces
10     Write in corpus_temp file
11 end
```

Fig 3: – Corpus Algorithm

D. Redundancy Handling - While going through such huge amount of test data collected, we end up having same corpus repeated several times in the file bringing redundancy in corpus file as shown in the figure below –

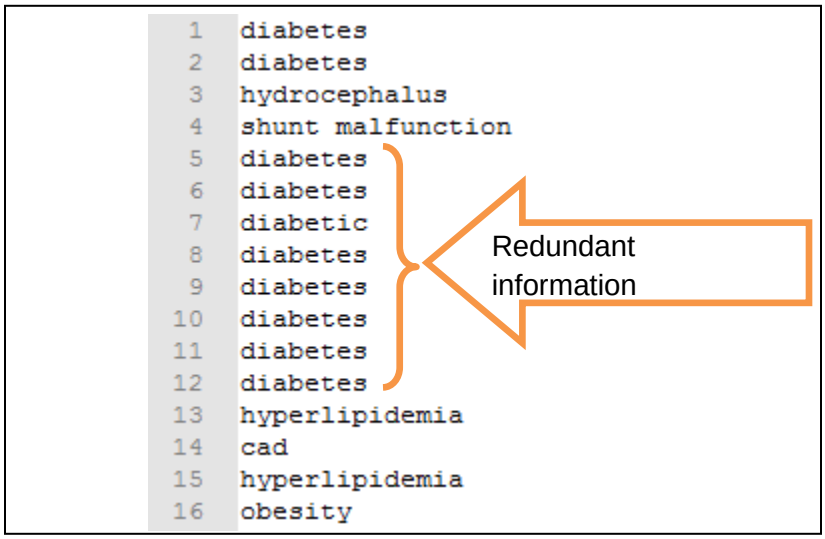

Fig 4: - Redundancy in corpus file

Following simple duplication removal algorithm is used to handle this redundancy in the corpus file –

```
1  Removing duplicate entry and store in corpus file
2  ------------------------------------------------
3  create a list of seen lines say line_seen.
4  read line from corpus_temp file
5  if line is not in line_seen
6      write in corpus file
7      add in the list of line_seen
8  end
```

Fig 5:– Redundancy removal algorithm

Analysis – This analysis part deals with calculating % redundant information which is removed using the redundancy handling technique. Following parameters are used for this –

Element – The tag elements like diagnosis, drug, procedure, vital and habit.

Dup. count – Total count of tag values including multiple occurrences

Dist. Count – Total distinct count of tag values

% redundant - $\frac{(Dup\ count - Dist.Count)X\ 100}{Dist\ count}$

Following table provides the data about the redundancy %.

Element	Dup. Count	Dist. Count	% Redundant
Diagnosis	917	210	336.67%
Drug	423	158	167.72%
Habit	1574	57	2661.40%
procedure	1152	122	844.26%
Vital	1794	79	2170.89%

Table 3: Redundancy Analysis

On an average, the redundant information % was 1236.19 which was removed by the redundancy handling algorithm discussed above.

RESULT: The result of corpus generation is saved in an entity file. Following is the results table

Input Notes Stats	
Number of Notes	2,248
Number of sentences	12, 551
Output Corpus stats	
Number of diagnosis	210
Number of procedure	122
Number of habits	57
Number of vitals	79
Number of drugs	158

Table 3: Results table

In order to test the corpus validity, a corpus reader program is developed. This program will take any text as input, iterate over the corpus file and signal match/no-match status. This is a very simple algorithm stated as below –

```
1  Corpus reader Aglorithm
2  ------------------------
3  get input srting
4  match the string in corpus file
5  if match found
6      return file name
7      drop the .ner part from the file name & show the result
8  else
9      return no_match status
10 end
```

Fig 6:– Corpus Reader algorithm

Performance Evaluation – This section deals with the execution performance of corpus generation. Following table shows the key performance metrics and execution time.

Platform used	
RAM	8 GB
Processor	Intel(R) Core(TM) i5-3320M CPU @ 2.60GHz, 2 Core(s), 4 Logical Processor(s)
Operating System	Microsoft Windows 7
Coding platform	Python 2.7
Number of Notes	2,248
Number of sentences	12,551
Input file size	~820 KB
Human Annotation	5,860
Average Execution time (5 runs)	13.6 seconds

Table 4: Performance evaluation

CORPUS ACCURACY: This section deals with the accuracy of manual annotation. Even while annotating manually, there are several human errors like typo, wrong tagging etc. These mistakes were uncovered and were corrected. Here are the parameters for this accuracy analysis.

Tag name: Name of tag entity like diagnosis, procedure, vital, habit and drug. This also includes the <Start: and <END> tags.

Manual annotation count: Count of each tag elements after manual annotation was done.

Actual count: Actual count of tag elements present in the notes.

Diff : Actual count – Manual annotation count

Accuracy %: $\frac{(Actual\ Count - Manual\ annotation\ count)\ X\ 100}{Actual\ count}$

Following two conditions were used in order to find the accuracy –

1. Count of *"<START:"* tag = count of *"<END>"* tag

2. Count of *diagnosis* tag + count of *procedure* tag + count of *vital* tag + count of *habit* tag + count of *drug* tag = count of *"<START:"* tag = count of *"<END>"* tag

The table given below shows the accuracy % of manual annotation –

Tag name	Manual annotation count	Actual count	Diff	Accuracy %
<*START:* tag	5857	5860	3	99.95%
<*END*> tag	5857	5860	3	99.95%
diagnosis tag	898	917	19	97.93%
procedure tag	1126	1152	26	97.74%
habit tag	1571	1574	3	99.81%
vtal tag	1789	1794	5	99.72%
drug tag	421	423	2	99.53%

Table 5: Accuracy analysis of Manual Annotation

Average accuracy % of the manual annotation is 99.23%.

FUTURE DIRECTIONS: Here is some future enhancement that is recommended to improve the accuracy for this system –

A. **N-gram analysis –** There are some corpuses which do not provide any significant meaning if analyzed as a single text. For example, in the diagnosis corpus "lung cancer", we can't say using just the first word "lung" as this is a disease. For this, we need to combine both words to get the meaning of diagnosis. For this purpose, when we apply this corpus in NER, we need to do N-gram analysis which will include more than one word during analysis.

B. **Text Similarity for morphological redundancy removal –** There are two levels of redundancies appearing in the corpus file.
   i. **Level 1 : Duplication redundancy** -Redundancy due to duplicate word
   ii. **Level 2: Morphological Redundancy** -Redundancy due to words appearing in different form, like plural, participle etc. e.g, tumor-tumors, diabetes-diabetics, exercise-exercises-exercising etc.

So far, the first level of redundancy is handled in this system. But second level of redundancy is not handled. In order to tackle this second level of redundancy, we need to implement different techniques of text similarity to reduce this type of redundancy.

REFERENCES:

1. Xu, H.; Stenner, S.P.; Doan,S.;Johnson, K.B.; Waitman,L.R.;Denny, J.C *MedEx: a medication information extraction system for clinical narratives;* Journal of the American Medical Informatics Association (JAMIA), 2009; pp. 19-24
2. Savova G.K,; Masanz,J.J.; Ogren, V.P.; Zheng, J.; Sohn, S.; Kipper-Schuler, C.K.;Chute, G.C. *Mayo clinical Text Analysis and Knowledge Extraction System (cTAKES): architecture, component evaluation and applications; ;* Journal of the American Medical Informatics Association (JAMIA), 2010; pp. 507-513.
3. Garla, V.; Re, L.V. III; Dorey-Stein, Z.; Kidwai, F.; Scotch, M.; Womack,J.; Justice,A.; Brandt,C. *The Yale cTAKES extensions for document classification: architecture and application* Journal of the American Medical Informatics Association (JAMIA), 2011; pp. 1-7
4. Liddy, E.D. *Natural Language Processing;* Encyclopedia of Library and Information Science 2nd Edition,2001
5. Ronan Collobert, R.; Weston, J.; Bottou, L.; Karlen, M.; Kavukcuoglu,K.; Kuksa, P. Natural *Language Processing (Almost) from Scratch;* Journal of Machine Learning Research 12, 2011
6. Wolniewicz, R. *Auto-Coding and Natural Language Processing*; 3M Health Information Systems
7. Madnani, N.; *Getting Started on Natural Language Processing with Python*
8. Wu, Y.; Denny, C.J; Rosenbloom, S.T.; Miller, R.A.; Giuse, D.A.;Dr.Ing; Xu, H. *A comparative study of current clinical natural language processing systems on handling abbreviations in discharge summaries*; Department of Biomedical Informatics, Department of Medicine, School of Medicine, Vanderbilt University, Nashville, TN
9. Bodenreider, O.; Willis, J.; Hole, W. *The Unified Medical Language System; National Library of Medicine*, 2004
10. Klassen, P.; *Gate Overview and Demo;* University of Washington CLMA treehouse Presentation, 2010
11. Coffman, A.; Wharton, N.; *Clinical Natural Language Processing Auto-Assigning ICD-9 Codes*;2007
12. Jurafsky, D.; Martin, J.H.; *Speech and Language Processing*; second edition
13. OpenNLP, URL - *https://opennlp.apache.org/* (visited on December 2014)
14. NLTK, URL - *http://www.nltk.org/book/* (visited on August 2015)
15. Language Model to detect Medical Sentences using NLTK, URL- *http://sujitpal.blogspot.com/2013/04/language-model-to-detect-medical.html* (visited on August 2015)
16. Python Text Processing with NLTK 2.0: Creating Custom Corpora, URL- *https://www.packtpub.com/books/content/python-text-processing-nltk-20-creating-custom-corpora* (visited on August 2015)

17. Writing a custom Name Finder model in OpenNLP, URL- *http://nishutayaltech.blogspot.com/2015/07/writing-custom-namefinder-model-in.html* (visited on January 2016)
18. The Patient Information Leaflet (PIL) Corpus, URL - *http://mcs.open.ac.uk/nlg/old_projects/pills/corpus/PIL/* (visited on August 2016)
19. ABNER – A Biomedical Named Entity Recognizer, URL - *http://pages.cs.wisc.edu/~bsettles/abner/* (visited on December 2016)
20. Introduction to Natural Language Processing, Courseera; URL - *https://www.coursera.org/learn/natural-language-processing* (December 2016 to January 2017), University of Michigan.