# SympGraph: A Framework for Mining Clinical Notes through Symptom Relation Graphs

Parikshit Sondhi[†], Jimeng Sun[‡], Hanghang Tong[‡], ChengXiang Zhai[†]
[†] Univ. of Illinois, Urbana, USA
[‡]IBM TJ Watson Research Center, USA
sondhi1@illinois.edu, jimeng@us.ibm.com, htong@us.ibm.com, czhai@illinois.edu

## ABSTRACT

As an integral part of Electronic Health Records (EHRs), clinical notes pose special challenges for analyzing EHRs due to their unstructured nature. In this paper, we present a general mining framework SympGraph for modeling and analyzing symptom relationships in clinical notes.

A SympGraph has symptoms as nodes and co-occurrence relations between symptoms as edges, and can be constructed automatically through extracting symptoms over sequences of clinical notes for a large number of patients. We present an important clinical application of SympGraph: *symptom expansion*, which can expand a given set of symptoms to other related symptoms by analyzing the underlying SympGraph structure. We further propose a matrix update algorithm which provides a significant computational saving for dynamic updates to the graph. Comprehensive evaluation on 1 million longitudinal clinical notes over 13K patients shows that static symptom expansion can successfully expand a set of known symptoms to a disease with high agreement rate with physician input (average precision 0.46), a 31% improvement over baseline co-occurrence based methods. The experimental results also show that the expanded symptoms can serve as useful features for improving AUC measure for disease diagnosis prediction, thus confirming the potential clinical value of our work.

## Categories and Subject Descriptors

H.3 [**INFORMATION STORAGE AND RETRIEVAL**]: Information Search and Retrieval

## Keywords

symptom graphs, physician notes,random walk,patient records

## 1. INTRODUCTION

Electronic Health Records (EHRs) have become a standard practice in modern healthcare delivery. Beyond storing patient records, EHRs are an ever-growing data repository that enables clinical data mining applications, especially comparative effectiveness research. Heterogeneous sets of EHR data are available in both structured and unstructured formats. For example, diagnosis information such as International Classification of Diseases - ICD9 codes[1], medication information such as generic drug names, and lab results are commonly stored in structured formats. Such structured EHR data are readily usable for data mining applications. However, clinical notes, arguably the most important part of the patient records, are largely unstructured. Despite the standard practice based on which physicians write the notes, such as SOAP notes(sec 2.1), by and large the information in physician notes remains unstructured, making it a significant challenge to extract and understand clinical concepts both within and across patients. The following example illustrates some of the needs and specific challenges in mining clinical notes.

**Clinical motivating example:** Congestive Heart failure (CHF), affecting 1 in 5 US adults, is the single most costly health problem for Centers for Medicare & Medicaid Services (CMS). With an aging population, the individual and societal burden is expected to increase. CHF is a complex syndrome with multiple pathophysiological manifestations that frequently overlap with those of other diseases, making early diagnosis challenging. Thus it is usually diagnosed in primary care, often as a late stage disease. In order to increase the chance of early diagnosis of CHF, it is important to collect all directly and indirectly relevant symptoms from a patient, and have a good understanding of the criteria for diagnosis. Unfortunately, the only criteria currently available were published by the Framingham investigators in 1971 based on clinical data acquired in the 1950 and 60s [11]. Fortunately, clinical notes are excellent sources from which we can potentially extract relevant symptoms to diseases such as CHF.

Automatic identification of Framingham criteria can greatly enhance the understanding of their prevalence as well as incidence in a patient population over time. Clinicians are also interested in understanding in general how all signs and symptoms are related to each other in order to possibly expand Framingham criteria to a broader set of symptoms that serve as robust signals for early detection of CHF. The need for extracting clinical signs and symptoms from patient records and further analyzing them also exists for many other diseases.

In this paper, we study the following questions: How to *capture* the relations among different symptoms extracted from clinical notes? How to *expand* a set of known symptoms to other relevant symptoms? How to efficiently *update* such symptom expansion when the underlying symptom relations change? To the best of our knowledge, no previous work has systematically studied these questions.

To this end, we present a general mining framework SympGraph, that turns patient encounter notes into a symptom graph to system-


*KDD'12,* August 12–16, 2012, Beijing, China.


[1] *http://icd9cm.chrisendres.com/*

atically reveal all the symptoms and symptom relations buried in the notes. A SympGraph is a graph with extracted symptoms as vertices, and the associations of symptoms as edges. It provides a general representation of symptoms and their relations and helps support many different mining applications under a unified framework. As an example of the applications enabled by a symptom graph, we present an important clinical application: *symptom expansion* which expands a small initial set of known symptoms to a bigger set of relevant symptoms. We present two different algorithms for solving symptom expansion in different contexts. When the underlying symptom graph is static, we propose *static symptom expansion*, which adopts a random walk type of algorithm to rank related symptoms based on their relevance to the initial set. When the underlying symptom graph changes (e.g., due to addition of new symptom relations or updating existing ones), we propose *dynamic symptom expansion*, which is based on a novel matrix update method for low rank approximation of the symptom graph.

We evaluate the proposed SympGraph framework on a large-scale real world clinical database with 1 million clinical notes of over 13K patients over 7 years, and the experimental results are quite promising. We show that SympGraph can successfully expand a known set of CHF related symptoms with a high agreement rate compared to physician judgements (average precision 0.46), providing a 31% improvement over baseline methods that do not use SympGraph. We further demonstrate that the expanded symptoms can help achieve over 10% improvement of area under the curve (AUC) measure for predicting the onset of CHF. In terms of the scalability, we compare static and dynamic symptom expansion methods, where interesting workload trade-off is revealed. In general, dynamic symptom expansion is preferred when multiple initial symptom sets are present and underlying relation changes are small.

## 2. BACKGROUND

### 2.1 Clinical Notes

A clinical note contains unstructured details regarding a patient's encounter with a healthcare professional. The content of the note varies depending upon the type of encounter, and there are many different types of encounters. For example, a note for a *Pharmacy Visit*, contains information regarding medications and their side effects, while a *Physician Visit* contains complete details of patient symptoms, medications, social history etc. Other example types of encounters are *Laboratory Visit* and *Case Manager Visit*. Of all these, the most important and abundant are the *Physician Visit* notes. These are generally loosely organized into four SOAP sections: *Subjective*, *Objective*, *Assessment* and *Plan*. Each section contains a specific kind of information regarding the patient. A sample is shown in Figure 1.

**Subjective:** details of a patient's condition in her own words. Also other details such as social history, family history, current medications etc.

**Objective:** observable information such as findings from physical examination, test results, vital signs, age, height, weight etc.

**Assessment:** a list of potential diagnoses.

**Plan:** information on the subsequent steps of actions of the patient, such as a treatment plan.

### 2.2 Symptom Extraction

Clearly, in order to analyze and mine symptom-related knowledge from clinical notes, we first need to go through a Symptom Extraction step, in which we process each encounter to identify the symptom mentions present and filter out negated symptoms. In

| Subjective: | Objective: |
|---|---|
| ANXIETY STATE NOS 300.00<br>DEPRESSIVE DISORDER NEC 311<br>ATRIAL FIBRILLATION 427.31<br>OLD MYOCARDIAL INFARCT 412<br>CONGESTIVE HEART FAILURE 428.0<br>Current outpatient prescriptions<br>** LOPRESSOR 50 MG PO TABS 1 tab two times a day 60 5 | 250.00 DM, CONTROLLED, TYPE II (primary encounter diagnosis)<br>428.0 CONGESTIVE HEART FAILURE<br>585.3 KIDNEY DZ,CHRONIC (GFR>30–59) STAGE III<br>412 OLD MYOCARDIAL INFARCT<br>715.09 GENERAL OSTEOARTHROSIS<br>427.31 ATRIAL FIBRILLATION |
| **Assessment:** | **Plan:** |
| BP 122/68 \| Pulse 78 \| Temp (Src) 98.1 (Oral) \| Resp 22 \| Wt 227 lbs<br>Abdomen: abdomen soft, non–tender, obese and no masses or organomegaly<br>Back: No CVA tenderness<br>Extremities: No edema | Continue present medication(s):<br>Referral(s) to: eye<br>Injection(s) ordered: b12<br>Schedule labs: Labs on return. |

**Figure 1: A sample clinical note, highlighting the four sections**

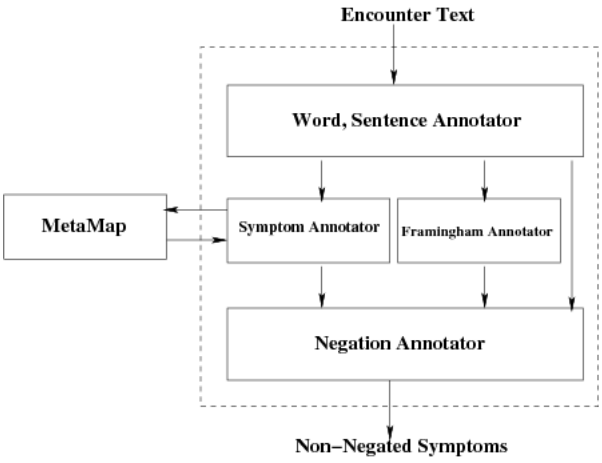


**Figure 2: UIMA pipeline architecture for symptom extraction**

the proposed SympGraph framework, we would use the extracted symptoms and their statistical relations to constract a graph, which can then support various analysis tasks, especially symptom expansion, an application that we would study in detail in this paper.

Since improving the accuracy of symptom mention extraction is not our focus, we mostly rely on existing techniques and extend them systematically to extract symptom mentions. Specifically, we use the Apache Unstructured Information Management Architecture (UIMA) toolkit(http://uima.apache.org/) to construct a Natural Language Processing (NLP) pipeline for the task. The main idea behind a UIMA pipeline is to construct a series of annotators. Each annotator is a program that parses the document text and marks the start and end locations for a certain type of entity in the text. We adopt the UIMA toolkit as it allows us to efficiently create and interconnect such annotators. Figure 2 shows the architecture of our UIMA pipeline.

**Word, Sentence Annotator** At the lowest level we start with annotators, which identify the word, sentence and SOAP section boundaries in the text. These annotations, along with the original text are then supplied to the subsequent levels.

**Symptom Annotator** Symptom annotator annotates all symp-

tom mentions present in the text. Symptoms are identified using MetaMap (http://metamap.nlm.nih.gov/), a tool developed at the National Library of Medicine, for mapping raw English text to standardized medical concepts in the Unified Medical Language System (UMLS) metathesaurus(http://www.nlm.nih.gov/research/umls/). UMLS contains over a million medical and general English concepts, which are further organized under a hierarchy of 134 semantic types. Each concept is assigned one or more high level semantic types. For example "dsyn" type, contains all concepts belonging to the category of "Disease or Syndrome".

Since MetaMap returns all types of concepts present in the text, we filter out the returned concepts based on their semantic types. More specifically, we only keep concepts belonging to the following semantic types as symptom concepts:

{ *sosy, dsyn, neop, fngs, bact, virs, cgab, acab, lbtr, inpo, mobd, comd, anab* }

**Framingham Annotator** The general Symptom Annotator described above does not accurately identify all mentions of Framingham Symptoms for heart failure in the text. Since our experiments are focused on Congestive Heart Failure (CHF), we also built a separate annotator, which identifies Framingham Symptoms more accurately in the text using a dictionary of symptom mentions and simple heuristics. The symptom dictionary was built by manually analyzing clinical records. We used the Framingham Annotator to extract 15 out of 16 Framingham symptoms in total. One symptom - *Weight loss* $\geq$ *4.5 kg in 5 days in response to treatment*, needed deeper text analysis and was not extracted.

**Negation Annotator** Presence of a symptom in a clinical note does not necessarily imply an *Assertion* which is its presence in a patient. The negation annotator looks at the surrounding text of each symptom annotation and filters out symptom mentions found in negated contexts based on simple heuristics such as presence of negation related words like "denies", "without", and "no". In principle, negative symptom findings can also indicate interesting relationships. However for this work we will only focus on positive symptom occurrences.

# 3. SYMPGRAPH

From a medical application perspective, it is very important to study clinical signs and symptoms in clinical notes, especially their relations both within a patient and across patients. Recognizing mentions of symptoms in the clinical notes is only the first step; after recognizing all the mentions of potential symptoms, we also need to develop methods for further organization, abstraction, and analysis of the extracted symptoms. A major challenge here is how to support different applications in a general and efficient way. To this end, we propose a general symptom mining framework called SympGraph, which provides a general graph representation of symptoms and their relations extracted from the clinical notes. SympGraph can be regarded as an abstraction of clinical notes from the perspective of symptom analysis and mining in the sense that it contains essential and sufficient information about symptoms and their relations required by many different mining applications.

Below, we first present the overall process of building symptom graphs for clinical notes and then present the algorithms for constructing and aggregating them.

## 3.1 Overview

The construction of symptoms can be done in a multi-level fashion as shown in Figure 3. At the patient level, clinical notes of a patient can be modeled as event sequences. The events in this paper are mentions of the symptoms in the clinical notes during a specific time interval (e.g., a day). SympGraph summarizes the

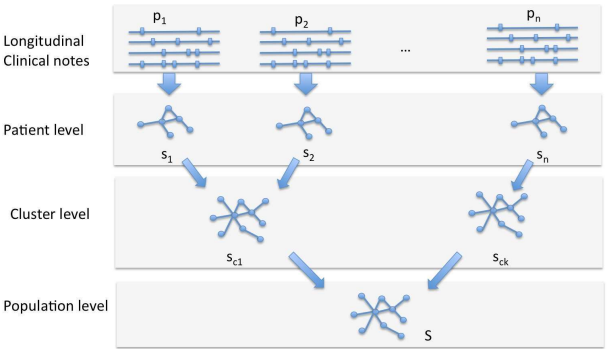


**Figure 3: Overview of Multiple levels of SympGraphs**

event sequences of a patient into a compact graph representation named patient (symptom) graph. A patient graph captures symptom co-occurrence relations for a given patient. The distance between patients can be defined based on the similarity of the underlying symptom graphs. We can aggregate individual patients' symptom graphs into a population-level symptom graph.

A symptom graph $G$ is represented as $G = \{V, E, w\}$, where every node $v \in V$ represents a symptom, the edge $e = (v_1, v_2)$ represents the relationship between two symptoms $v_1$ and $v_2$, and weighting function $w : e \rightarrow \mathbb{R}^+$ maps the edge $e$ to a positive value $w(e)$.

For a given patient $p$, such a symptom graph can be constructed, where nodes correspond to the symptoms extracted from $p$'s clinical notes, and edges correspond to the relationship of symptoms exhibited in $p$'s clinical notes.

## 3.2 Graph Construction

One simple way to construct the graph is to form edge $(i, j)$ in $G$ if symptoms $i$ and $j$ co-occurr in the same encounter note. Or formally, given a sequence of clinical encounters $c_i$ of $p$ over time $(1 \leq i \leq C_p)$, we can construct a symptom to encounter matrix $\mathbf{N}^p \in \mathbb{R}^{S \times C_p}$, where $S$ is the total number of symptoms and $C_p$ is the number of encounters for patient $p$, and the element in $\mathbf{N}^p$ is defined as

$$\mathbf{N}^p_{ij} = \begin{cases} 1 & \text{if symptom } i \text{ is in encounter note } j, \\ 0 & \text{otherwise.} \end{cases}$$

The $i$-th row vector $\mathbf{N}_{i:}$ of $\mathbf{N}$ corresponds to the patient $p$'s encounters that contain symptom $i$. The $j$-th column vector $\mathbf{N}^p_{:j}$ corresponds to the symptoms extracted from the encounter note $j$. With the encounter matrix $\mathbf{N}^p$, the symptom co-occurrence matrix $\mathbf{G} \in \mathbb{R}^{S \times S}$ is

$$\mathbf{G} = \sum_p \mathbf{N}^p \cdot \mathbf{N}^{pT}. \quad (1)$$

The nonzero elements in $\mathbf{G}$ correspond to all the edges of the symptom graph $G$. Intuitively, the symptoms extracted from the same encounter note are connected to each other, and form a clique. All the cliques from different encounter notes of a patient can be summed together to form the symptom matrix $\mathbf{G}$. The weighting function $w(i, j)$ is exactly the element $\mathbf{G}_{ij}$.

Beyond this simple construction, we also extend the formulation to a location sensitive construction. The location sensitive idea arises out of the intuition that a physician will likely evaluate the presence of related symptoms in succession and they will likely appear closeby in the clinical notes. Thus if two symptoms are far away from each other in the encounter note, the edge weight be-

tween them should be small. If they are close, the weight should be large. In this paper, we use the inverse of the distance as the weight. Such formulation can still be captured by generalizing the multiplication operator as $\mathbf{u} \cdot \mathbf{v}^T = \sum_{1 \le i \le C_p} f(\mathbf{u}_i, \mathbf{v}_i)$ where $f(\mathbf{u}_i, \mathbf{v}_i) = 1/d$ and $d$ is the location offset in words between symptom $u$ and $v$ in the encounter $i$.

**Graph Normalization:** In order to alleviate the problem where some symptoms are much more common than others, we further normalize a symptom matrix $\mathbf{G}$ constructed using a method described above through the following

$$\tilde{\mathbf{G}} = \mathbf{G}\mathbf{D}^{-1}$$

where $\mathbf{D}$ is a diagonal matrix with $\mathbf{D}_{ii} = \sum_j \mathbf{N}_{ij}$. The resulting matrix $\tilde{\mathbf{G}}$ will be Markovian, namely the column sums to 1. In this case, the popular symptoms will be penalized since the outgoing edges from them will be normalized. Note that the normalization does not preserve symmetric property of the original matrix $\mathbf{G}$. There are other normalization methods that can preserve the symmetric property such as the normalized Laplacian normalization $\tilde{\mathbf{G}} = \mathbf{D}^{-1/2}\mathbf{G}\mathbf{D}^{-1/2}$. In this paper, we pick the asymmetric normalization because we believe the relations are not always symmetric. For example, $v$ is a common symptom with many neighbors including $u$, but $u$ is only connected to $v$. In this case, the importance of $u$ to $v$ (i.e., $w(v,u)$) is small, but the importance of $v$ to $u$ (i.e., $w(u,v)$) is significant.

**Aggregation:** Finally, given a set of normalized symptom graphs $G_1, G_2, \ldots, G_K$, we can also easily combine them into an aggregated symptom graph $G$ by averaging the symptom matrices:

$$\mathbf{G} = \frac{1}{K}\sum_i \mathbf{G}_i.$$

Since the symptom matrices $\mathbf{G}_i$ $(1 \le i \le K)$ are already normalized, the simple averaging will preserve the Markovian property. Aggregation provides a general and flexible way to analyze the data with different "resolutions"; this in combination with graph analysis algorithms naturally supports many different ways to analyze and exploit symptoms and their relations in a variety of applications.

# 4. STATIC SYMPTOM EXPANSION

Given a symptom graph, one important clinical application is to automatically explore and expand an existing set of symptoms. Such expansion is necessary to improve the diagnosis of diseases (one use case is described in Section 1 about extending Framingham criteria for CHF diagnosis). To facilitate the discussion, we define the following notations: Let $ES$ be the set of the existing symptoms that one constructed (e.g., symptoms included in the Framingham criteria). Let vector $\mathbf{e} \in \mathbb{R}^S$ be the indicator vector of the existing symptom set, where

$$\mathbf{e}_i = \begin{cases} 1 & \text{if symptom } i \in ES, \\ 0 & \text{otherwise.} \end{cases}$$

Let $\mathbf{r} \in \mathbb{R}^S$ be the relevance score vector of all symptoms with respect to the existing set $ES$, which we hope to compute.

Intuitively, to perform symptom expansion, we would be interested in obtaining symptoms that co-occur frequently with the existing set of symptoms. Formally, such co-occurring symptoms can be captured through $\mathbf{r} = \mathbf{G}\mathbf{e}$, where highly ranked values in $\mathbf{r}$ correspond to the most commonly co-occurring symptoms with $ES$. Mathematically, this can be achieved by a one-step random walk from the existing set $ES$ based on the symptom graph $G$.

In addition to the immediately co-occurring symptoms, we also perform the expansion recursively through random walk with restart [20]. Intuitively, random walk with restart (RWR) can be regarded as modeling a random particle that with probability $c(0 < c < 1)$ traverses randomly along edges of $G$, and probability $1 - c$ jumps back to one of the initial symptoms in $ES$. The recursion of random walk with restart is defined as

$$\mathbf{r} = c\mathbf{G}\mathbf{r} + (1-c)\mathbf{e}. \quad (2)$$

The solution to the linear system eq-(2) is $\mathbf{r} = (1-c)(I - c\mathbf{G})^{-1}\mathbf{e}$. Many effective algorithms exist to solve eq-(2) when the underlying graph $\mathbf{G}$ is fixed. One simple and scalable method for solving this linear system is to use the power method to iteratively apply eq-2 with the initialization $\mathbf{r} = \mathbf{e}$ (referred to as 'Power-method'). Another effective way to solve eq-(2) is to use a low-rank approximation, followed by a matrix inversion of size $l \times l$ (where $l$ is the rank of the low-rank approximation) to get all possible relevance scores. This solution [20], called NB_Lin, is summarized in Alg. 1 for completeness, where it is divided into two stages: NB_LIN_Pre() and NB_LIN_OQ(). In NB_LIN_Pre() (steps 1-3), a low-rank approximation is performed for the normalized adjacency matrix $\mathbf{A}$ and a matrix inversion $\mathbf{\Lambda}$ (referred to as 'core matrix') is computed. Next, in NB_LIN_OQ() (steps 4-5), only a small number of matrix-vector multiplications are computed to output the ranking vector. Therefore, NB_Lin is much more efficient for the query response, especially when we have multiple queries. (We can view the "seed symptoms" used for symptom expansion as a "query" and the task of symptom expansion as to "retrieve" additional related symptoms to the query.)

**Algorithm 1** NB_Lin (repeated from [20] for completeness)

**Input:** The normalized adjacency matrix $\mathbf{G}$, the query vector $\mathbf{e}$ and $c$.
**Output:** The ranking vector for the source node $\mathbf{r}$.
1: **Pre-Compute Stage** (NB_LIN_Pre())
2: do low-rank approximation for $\mathbf{G} = \mathbf{U}\mathbf{V}$
3: pre-compute and store the core matrix $\mathbf{\Lambda} = (\mathbf{I} - c\mathbf{V}\mathbf{U})^{-1}$
4: **On-Line Query Stage** (NB_LIN_OQ())
5: output $\mathbf{r} = (1-c)(\mathbf{e} + c\mathbf{U}\mathbf{\Lambda}\mathbf{V}\mathbf{e})$

In terms of applications, symptom expansion can be used at different levels of symptom graphs. At the patient level, users like physicians, nurses or even patients themselves can expand the existing set of symptoms from the current encounter to a more general set of symptoms based on the symptom graph built on the patient history. Those additional symptoms can be used as personalized query suggestions for diagnosis. One can also take the current symptoms from a patient $p$ and perform symptom expansion based on the cluster symptom graph of $p$ to immediately leverage patterns extracted from similar patients like $p$. Finally, at the population level, symptom expansion enables understanding of typical symptoms to a disease. "Typical symptoms" as described in medical books such as Framingham symptoms for CHF are just a small initial set of symptoms for diagnosing a disease such as CHF. Starting from an existing set of symptoms for a given disease, additional sets of symptoms can be obtained based on the population-level symptom graph using random walk with restart. The expansion set of symptoms can then be used as additional features for building predictive modeling for that disease, and also potentially enhance knowledge and understanding about the disease. We will present some initial clinical case studies in section 6 to show that the expansion set of symptoms discovered using our algorithm are indeed helpful for improving the prediction accuracy of the onset of CHF.

# 5. DYNAMIC SYMPTOM EXPANSION

In this section, we address the challenges in computing symptom expansion when the underlying symptom graph is changing. We first present our algorithm, and then analyze its accuracy as well as the efficiency.

## 5.1 Proposed Algorithm for Dynamic RWR

When the underlying symptom graph $\mathbf{G}$ changes over time, the results of symptom expansion would also have to be updated. In the case of very frequent changes, which can happen, e.g., due to additions of many new patient records daily, updating all the results can be computationally expensive. Specifically, let $\hat{\mathbf{G}}$ indicate the change of the symptom graph (i.e., $\hat{\mathbf{G}} + \mathbf{G}$ is the new normalized adjacency matrix). If we still want to apply NB_Lin to update the corresponding ranking vectors straight-forwardly, we need to re-compute the low-rank approximation as well as the core matrix. In other words, the computational cost in NB_Lin_Pre() now becomes part of the on-line query cost in the dynamic case, which might be too expensive.

Thus an interesting question is: how can we update the ranking vector efficiently? To address this issue, we propose an efficient algorithm to update the low rank approximation as well as the core matrix. Our key observation is as follows: for the application of symptom graph, $\hat{\mathbf{G}}$ itself can often be written or approximated by some low-rank approximation with a much lower rank size (say $k$, then $k << l$ where $l$ is the rank of $\mathbf{G}$). In this case, we can easily update the low-rank approximation of the whole, new symptom matrix as well as the core matrix.

**Algorithm 2** Update-RWR

**Input:** The previous low rank approximation $\mathbf{U}$ and $\mathbf{V}$, the previous core matrix $\mathbf{\Lambda}$, the change adjacency matrix $\hat{\mathbf{G}}$, the query vector $\mathbf{e}$ and $c$.

**Output:** The updated low rank approximation $\mathbf{U}$ and $\mathbf{V}$, the updated core matrix $\mathbf{\Lambda}$, and the updated ranking vector $\mathbf{r}$ for the query vector $\mathbf{e}$.

1: **Update $\mathbf{U}$ and $\mathbf{V}$** (Update_LowRank())
2: Re-form or approximate $\hat{\mathbf{G}}$ as: $\hat{\mathbf{G}} = \mathbf{XY}$
3: Update: $\mathbf{U} \leftarrow [\mathbf{U}\ \ \mathbf{X}]$; $\mathbf{V} \leftarrow \begin{pmatrix}\mathbf{V}\\ \mathbf{Y}\end{pmatrix}$
4: **Update Core Matrix** (Update_Core())
5: Compute $\mathbf{L} = (\mathbf{I} - c\mathbf{YX} - c^2\mathbf{YU\Lambda VX})^{-1}$
6: Compute $\mathbf{P} = c\mathbf{\Lambda VX}$; and $\mathbf{Q} = c\mathbf{YU\Lambda}$
7: Update $\mathbf{\Lambda} \leftarrow \begin{pmatrix}\mathbf{\Lambda} + \mathbf{PLQ} & \mathbf{PL}\\ \mathbf{LQ} & \mathbf{L}\end{pmatrix}$
8: **Update Ranking Vector** (Update_Rank())
9: Update $\mathbf{r} = \text{NB\_LIN\_OQ}(\mathbf{U}, \mathbf{V}, \mathbf{\Lambda}, c, \mathbf{e})$

## 5.2 Proofs and Analysis

The accuracy of the proposed Alg. 2 is summarized in Lemma 5.1.

LEMMA 5.1. **Effectiveness of Update-RWR.** *If $\mathbf{G} = \mathbf{UV}$ and $\hat{\mathbf{G}} = \mathbf{XY}$, Alg. 2 gives the correct ranking vector.*

PROOF. Let $\mathbf{A} = \mathbf{G} + \hat{\mathbf{G}}$, we have $\mathbf{A} = \hat{\mathbf{U}}\hat{\mathbf{V}}$, where $\hat{\mathbf{U}} = [\mathbf{U}\ \ \mathbf{X}]$; and $\mathbf{V} = \begin{pmatrix}\mathbf{V}\\ \mathbf{Y}\end{pmatrix}$.

Let $\hat{\mathbf{\Lambda}} = (\mathbf{I} - c\hat{\mathbf{V}}\hat{\mathbf{U}})^{-1}$. We have

$$\begin{aligned}\hat{\mathbf{\Lambda}} &= (\mathbf{I} - c\hat{\mathbf{V}}\hat{\mathbf{U}})^{-1}\\ &= \begin{pmatrix}\mathbf{I} - c\mathbf{VU} & -c\mathbf{VX}\\ -c\mathbf{YU} & \mathbf{I} - c\mathbf{YX}\end{pmatrix}^{-1} \qquad (3)\end{aligned}$$

Applying the block matrix inverse lemma [3] to eq-(3), we can verify that the step 7 in Alg. 2 gives the exact $\hat{\mathbf{\Lambda}}$ matrix.

Then, for the updated ranking vector $\mathbf{r}$, we have

$$\begin{aligned}\mathbf{r} &= (1-c)(\mathbf{I} - c\mathbf{A})^{-1}\mathbf{e}\\ &= (1-c)(\mathbf{I} - c\hat{\mathbf{U}}\hat{\mathbf{V}})^{-1}\mathbf{e}\\ &= (1-c)(\mathbf{I} + c\hat{\mathbf{U}}\hat{\mathbf{\Lambda}}\hat{\mathbf{V}})\mathbf{e}\\ &= (1-c)(\mathbf{e} + c\hat{\mathbf{U}}\hat{\mathbf{\Lambda}}\hat{\mathbf{V}}\mathbf{e}) \qquad (4)\end{aligned}$$

Notice that the second to the last step is due to the Sherman-Morrison lemma [15]. This completes the proof. □

The efficiency of the proposed Alg. 2 is summarized in Lemma 5.2. It can be seen that it is much more efficient by (1) avoiding the re-computation of the low-rank approximation of the new normalized adjacency matrix, which takes $O(m)$ time; (2) avoiding directly updating the new core matrix, which takes $O(k+l)^3$.

LEMMA 5.2. **Efficiency of Update-RWR.** *Let $\hat{m}$ be the non-zero elements in $\hat{\mathbf{G}}$, the time complexity of Alg. 2 is $O(\hat{m} + nkl)$, where $k$ and $l$ are the rank of $\mathbf{X}$ and $\mathbf{U}$, respectively.*

PROOF. Step 2 takes $O(\hat{m})$ time to get the low-rank approximation of the matrix $\hat{\mathbf{G}}$. Step 3 takes $O(nk)$ time. Step 5 takes $O(nk^2 + nkl + k^3)$ time. Step 6 takes $O(nkl + l^2k)$ time. Step 7 takes $O(l^2k)$ time. Finally, step 9 takes $O(n(k+l) + (k+l)^2)$ time. Notice that $k \ll l \ll n$, therefore, the overall time complexity of Alg. 2 is $O(\hat{m} + nkl)$, which completes the proof. □

# 6. EXPERIMENTS AND RESULTS

The main motivation behind our experiments was to answer the following two questions: (1) **Symptom Expansion:** How useful are symptom graphs for the symptom expansion task? (2) **Dynamic Update:** How does the performance(efficiency/accuracy) of the proposed *Update-RWR* algorithm compare to the baseline *Power* method? The following subsections present a detailed description of our dataset, experiment design, evaluation metrics and the obtained results.

## 6.1 Dataset Description

Our dataset consists of 1 Million encounters of 13K patients. Among them, approximately 500K encounters (case subset) are cases (for 4.6K patients diagnosed with CHF), while the rest (control subset) are controls (8.4K patients without CHF). The case patients are defined by the operational criteria that include two or more mentions of CHF related diagnosis in outpatient visits and one or more mentions of CHF related diagnosis in the problem lists. Ten control patients are matched to each case patient on the age, gender and the clinic. Control patients do not have any CHF diagnosis by the diagnosis date of their corresponding case patient.

Each encounter is associated with a date, a patient ID and the type of visit. After processing the dataset with our UIMA pipeline, nearly 11.8K unique symptoms were extracted with an average of 12.7 positive symptoms and 5.7 negative symptoms per encounter.

## 6.2 Symptom Expansion Analysis

To evaluate the utility of a symptom graph for the symptom expansion task, we constructed six symptom graphs using the case subset (500K encounters) of the data. Four of them are constructed by using the four sections of the clinical notes (i.e., subjective, objective, assessment, and plan), respectively (all with location-sensitive weighting). The fifth was built by using text data in all the sections and location-sensitive weighting (LocSenseRWR). The

sixth was also built by using all the sections, but with equal symptom weighing (EqRWR). That is, in the sixth graph, if two symptoms appeared together in a clinical note, their edge weight was increased by 1 instead of $1/d$, where $d$ is the location offset in words. The first five allowed us to analyze the utility of different sections for the task. The last one allowed us to analyze the utility of location sensitive heuristic itself.

For evaluating each symptom graph, we started with the 15 Framingham symptoms as the initial vector and generated a ranked list of related symptoms using RWR. The accuracy of this ranked list was then evaluated in two ways - a) by comparing with expert judgements, b) by looking at the usefulness of the related symptoms for improving CHF prediction.

### 6.2.1 Expert Judgment Evaluation

*Test Set Construction*

The gold standard of related symptoms used for evaluating the ranked lists was generated via pooling [7], an approach used frequently in information retrieval evaluation. Each symptom graph provided a single ranked list of symptoms. These ranked lists along with an additional list obtained via a co-occurrence baseline which ranked related symptoms based on how frequently they co-occured with the Framingham symptoms in the clinical notes, were used for judgements. Top 50 symptoms from each ranked list, a total of 175 unique symptoms, were pooled together and judged by two medical experts. Finally, the 72 symptoms labeled as relevant by both experts were deemed as relevant and used for evaluation. The inter-annotation agreement was 81.8% (32 disagreements out of 175).

*Evaluation Metrics*

We used three standard information retrieval measures, i.e., Average Precision, Precision@10 and Recall@100, for quantitative evaluation, which are described below.

Average Precision intuitively captures the average of precision at every point when a new relevant symptom is retrieved. A higher $AP$ implies that a greater number of relevant symptoms are being assigned higher ranks in the ranked list. Suppose for some ranking $r_i \in R$ , there are $k_i$ relevant symptoms. Further let $rank(j)$ be the rank of the $j^{th}$ relevant symptom and $P(rank(j))$ be the precision at the rank of the $j^{th}$ relevant symptom. Then

$$P(rank(j)) = \frac{\#\ Relevant\ Symptoms\ Till\ rank(j)}{\#\ Symptoms\ Till\ rank(j)} = \frac{j}{rank(j)}$$

Average precision of $r_i$ is then given by: $AP(r_i) = \frac{\sum_{j=1}^{k_i} P(rank(j))}{k_i}$.

Precision@10 is defined as the percentage of relevant symptoms in the top 10 symptoms of the ranked list. $Recall@K$ is computed as the ratio of the number of relevant symptoms in the top $K$ retrieved list to the total number of all relevant symptoms.

*Comparison of SympGraph construction methods*

Figure 4 shows the performance comparison of SympGraph with location sensitive heuristic (labeled LocSenseRWR) over two different baselines - symptom co-occurrence and SympGraph with equal weighing. All graphs were constructed over the case subset and symptom ranked lists were obtained by using the 15 framingham symptoms as the initial vector for RWR. The location sensitive method (LocSenseRWR) outperforms the baselines in all three metrics. With an average precision of 0.46, it improves over the two baselines by about 31%. In particular, seven out of top ten symptoms were found to be relevant and nearly 62% of known relevant symptoms were recovered in top 100. The results suggest that not

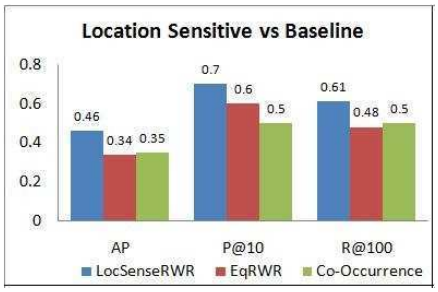


**Figure 4: Comparison of location sensitive weighing with other baselines (31% improvement in AP over co-occurrence)**

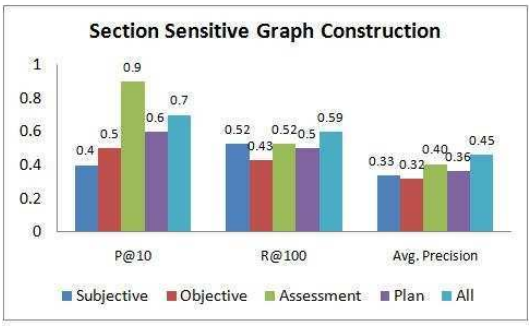


**Figure 5: Location sensitive graphs over different sections**

only are symptom graphs useful for the task, but the location sensitive heuristic indeed helps improve performance.

Figure 5 shows the performance comparison of SympGraphs constructed over different SOAP sections using the location sensitive heuristic. Performance varies significantly across sections.Graph based on Assessment section performed the best. This makes sense as the Assessment section primarily contains a concise summary of asserted symptoms and potential diagnoses. On the other hand, as one would expect, Subjective and Objective graphs did not do so well presumably since these sections are usually noisy with a lot of additional details. In practice, well formed section boundaries may not always be available. Therefore inspite of its slightly lower performance, we are still most interested in the graph built using the entire notes, and will continue to use it for subsequent experiments.

### 6.2.2 CHF Prediction Evaluation

In order to further evaluate the utility of related symptoms, we defined the CHF prediction task. The goal was to analyze a sequence of patient medical records and predict if a patient was likely to develop CHF in future, using only the list of related symptoms as features. For this task we used the entire dataset of 1M encounters.

For positive examples (i.e., case patients with CHF), we only used the data from clinical notes before a patient was diagnosed with CHF for constructing the feature vector. For negative examples (i.e., control patients without CHF), we used all available data. Each selected symptom became one feature, and its feature value was the number of occurrences of that symptom in the patient's records.

We compared the following methods for selecting features:

*Framingham*: Uses 15 Framingham symptoms + 35 randomly selected symptoms as features

*Co-occurrence*: Uses 15 Framingham symptoms + 35 additional symptoms that most commonly co-occurred with Framingham symptoms in the encounters.

| | Fram.+Random | Fram+Co-Occ. | LocSenseRWR |
|---|---|---|---|
| AUC | 0.590(14.24%) | 0.649(3.85%) | **0.674** |

**Table 1: CHF prediction performance.Parenthesis show improvement achieved by LocSenseRWR over the method**

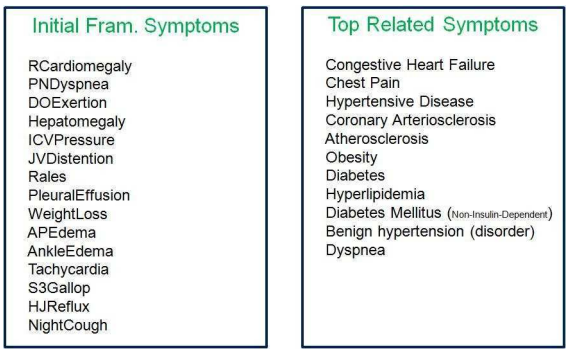


**Figure 6: Symptom expansion results**

*LocSenseRWR*: Uses the top 50 ranked symptoms obtained from a symptom graph constructed with location sensitive heuristic.

Using the selected features, the model was trained on 90% patients and tested on the rest with 10 fold cross validation. Evaluation metric used was Area Under the ROC-Curve (AUC). The classification model was an SVM classifier with Gaussian kernel:

$$\mathbf{K}_{ij} = e^{-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2(a\bar{r}^2)}}, \tag{5}$$

where $\mathbf{x}_i$ and $\mathbf{x}_j$ are data samples and $\bar{r}$ is the average of pairwise distances among all the data samples. $a$ is chosen from $[2^{-3}, 2^{-2}, 2^{-1}, 1, 2^1, 2^2, 2^3]$. The SVM trade off parameter $C$ is chosen from $[0.01, 0.1, 1, 10, 100]$. We report the best results among all the combinations of $a$ and $C$.

The results from CHF prediction task are shown in Table 1, which corroborate our observations from human judgement evaluation. The location sensitive graph (LocSenseRWR) outperforms the baselines, suggesting that the expanded symptoms are useful for the task. Figure 6 shows some of these top expanded symptoms. In particular, coronary arteriosclerosis, hypertension, diabetes etc. are all very related co-morbidity and risk factors to HF and have been confirmed by clinical experts.

## 6.3 Dynamic Update Analysis

In order to evaluate the efficiency of the proposed *Update-RWR* method we defined two real world update tasks: (1) **Symptom Extractor Update**: This task assumes that we have just updated our UIMA pipeline to extract a new symptoms from clinical notes. As a result the existing graph must be updated with a new symptom node and all its edges. (2) **Patient Update**: This task assumes that data for a new patient (a small patient subgraph) has joined and needs to be incorporated into an existing symptom graph.

Both tasks require incorporating a smaller subgraph (representing new information) into an existing symptom graph. We want to evaluate efficiency and effectiveness of Update-RWR algorithm compared to recomputation using the power method.

All dynamic update experiments were carried out on a 32-bit Windows Vista machine with 3Gb RAM and intel core-2 Duo 2.5 GHz processor. The initial symptom graph $\mathbf{G}$ consisted of $11.8K$ symptom nodes and 1.4 million edges. For both patient update and symptom update tasks, individual patient subgraphs usually contained less than 1000 edges, while for symptom update task, the symptom subgraphs contained up to 2500 edges.

### 6.3.1 Experimental Design

**Accuracy Analysis:** For accuracy evaluation, the top $N$ ranked symptoms generated by the Update-RWR and the baseline methods (Power method) are compared by using the baseline symptoms as the gold standard. Evaluation metrics used are: Average Precision (AP), Precision at 50 ($P$@50), and Recall at $N$.

Intuitively precision at 50 tells us how many of the top-50 symptoms generated by Update-RWR, were found in the gold standard. Recall at $N$ gives the percentage of gold standard symptoms that were recovered by Update-RWR. Finally AP takes a high value if both precision and recall are high.

**Efficiency Analysis:** Recall that both methods start with an initial normalized matrix $\mathbf{G}$ as their starting point. For the baseline, adding a new subgraph requires the modified graph matrix to be renormalized. For Update-RWR we need to calculate the $\mathbf{U}$,$\mathbf{V}$,and $\mathbf{\Lambda}$ once on the initial normalized matrix and update them whenever a new subgraph is added (refer to algorithms 1 and 2). As a result the speedup is evaluated by looking at three different times:

*Initial one time matrix construction*: In case of Update-RWR, this is the time for constructing the matrices $\mathbf{U}$,$\mathbf{V}$ and $\mathbf{\Lambda}$ on the large initial matrix as in algorithm 1. For the baseline this time is 0. This will happen only once.

*Matrix update time*: For the baseline, this is the time required to add the new subgraph matrix into the original symptom graph matrix and re-normalize the resulting graph. This is the time for computing the updated $\hat{\mathbf{U}}$, $\hat{\mathbf{V}}$ and $\hat{\mathbf{\Lambda}}$ as in algorithm 2. This will happen every time a new subgraph is added.

*Online Query time*: This is the time to perform random walk restart. In case of baseline, it is the time taken for applying the power method. For Update-RWR it is the time for performing the step $NB\_LIN\_OQ()$ in algorithm 1.

### 6.3.2 Results

Figure 7 presents the results for Symptom Update task for the symptom 'PNDyspnea'. Results for other symptoms were similar. The task was simulated by constructing a symptom graph without one of the Framingham symptom nodes and later updating it with all of missing symptom's edges. RWR was then run on the updated matrix using the new symptom as the initial vector. The accuracy was compared for top $N = 200$ symptoms (results for other values of $N$ were similar). The low rank approximation parameter $l$ for the initial normalized matrix $\mathbf{G}$ serves to control the trade-off between efficiency and accuracy of the method. The low rank approximation parameter $k$ for the update matrix $\hat{\mathbf{G}}$ did not affect the results as much and was empirically set to 2. The low rank approximations were obtained using the signular value decomposition (SVD [3]) method. Increasing $l$ results in improved average precision and recall@200, which both tend to 1 as $l$ goes beyond 400. On the other hand even for small values eg. $l = 50$ the performance is still reasonably good. In particular, we observed a $P$@50 of 1 for all data points in Figure 7. This implies at all of the top 50 ranked symptoms by Update-RWR were found in the top 200 baseline symptoms. In addition, 89% symptoms were common in the two lists. Finally AP of 0.88 implies that most of the symptoms ranked high by Update-RWR were common to both lists.

In terms of efficiency, we observe that the speedup reduces as $l$ increases. This is because a higher $l$ implies a matrix multiplication between dense higher dimensional matrices. Assuming $l$ as 50 we observe that while the difference in matrix update times is not that much, Update-RWR outperforms the baseline significantly in online query performance. We do not compare the performance of Update-RWR with the original $NB\_Lin$ in algorithm 1 as it

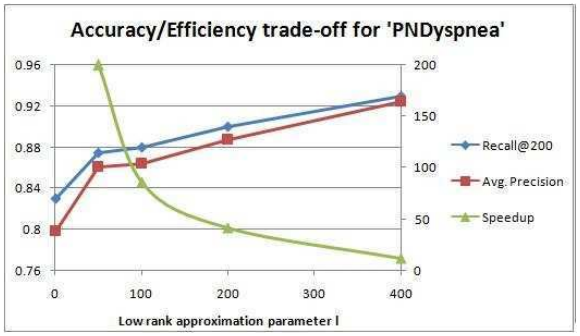


**Figure 7: Symptom update results (speed up values are for online query performance)**

| Method | Initial Matrix Construction Time | Avg. Matrix Update Time | Avg. Online RWR Query Time |
|---|---|---|---|
| Update-RWR | 5.28 | 3.067(+/-.046) | 0.017(+/-.005) |
| Baseline(Power) | - | 6.705(+/-.094) | 1.336(+/-.028) |
| Times Speedup | - | 2.186 | 78.59 |

**Table 2: Time comparisons for patient update task. All values in secs. Averages are over 50 patient additions. Parenthesis provide standard deviations.**

would require us to reconstruct the three $\mathbf{U}$,$\mathbf{V}$,$\mathbf{\Lambda}$ matrices on every update, and as a result the method would perform far worse than the other two, making the comparison not interesting.

Table 2 presents the different time metrics for Update-RWR and Power method for adding a new patient subgraph and running RWR. The time values are averaged over 50 different patient additions. The value of low rank approximation parameters $l$ and $k$ were fixed at 50 and 2, respectively. As expected, our proposed method is over 78 times faster than the baseline. This improves efficiency dramatically when multiple RWR queries need to be performed. Figure 8 illustrates this point.

Finally, Figure 9 compares the accuracy. Here we update $\mathbf{G}$ by repeatedly adding new patients using Update-RWR i.e. for adding $x$ patients, Update-RWR gets applied $x$ times consecutively. The accuracy of the updated matrix after say $x$ patient additions, is evaluated by comparing against the ideal normalized matrix for those additions, i.e. the one we get after adding all $x$ patient subgraphs to $\mathbf{G}$ and renormalizing. We observe that the accuracy continues to remain high inspite of over 400 consecutive new patient additions.

## 7. RELATED WORK

Information extraction from clinical records and medical text in general has been studied in the existing work (see, e.g., [4, 12, 23, 18, 16]). In most cases, standard information extraction techniques have been applied or adapted to the medical domain for recognizing various entities such as diseases [22] and treatments [2]. We however, go beyond symptom extraction to further construct symptom graphs and study how to leverage such graphs to convert raw symptom mentions into more useful knowledge.

Graph representations are frequently used to facilitate data mining. In most cases, a graph is naturally constructed based on observed data where the edges are directly available in an application. For example, Web graphs are often constructed with web pages as nodes and hyperlinks as edges [13, 8]. Also, information network

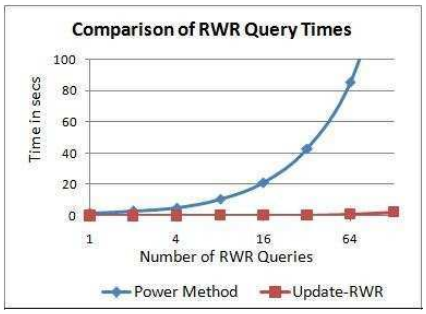


**Figure 8: Online RWR query execution times. The scale on x-axis is logarithmic.**

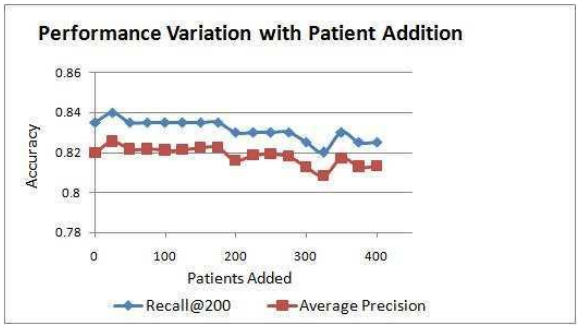


**Figure 9: AP and Recall@200 for patient update task**

mining has recently been studied (see e.g., [19]), especially bibliographical network and social networks. A main difference between the proposed SympGraph and these graphs lies in that both the nodes and edges in a SympGraph are *extracted* computationally from text data. An immediate consequence of this difference is that we need to address the issue of uncertainty in the edges of a SympGraph, which raises new challenges in designing graph mining algorithms.

Random Walk with Restart (RWR) [20] and related ideas have been applied to many problems such as Personalized PageRank [5] for Web search and image tagging [1]. Thanks to our graph viewpoint for the extracted symptoms, we are able to leverage this powerful tool to discover related patient symptoms. In terms of computation, most of the existing fast algorithms apply to the static graphs. In [21], the authors proposed an efficient on-line algorithm for *skewed bipartite* graphs, which does not apply to the symptom graph. Other remotely related works include dynamic PageRank [14, 17], Fast SimRank [10, 9], etc. Our work can be regarded as adding to this line of work a new application of RWR for finding related symptoms in graphs of patient symptoms.

Prediction of patient diseases based on patient records has been studied mostly by using the structured fields in the patient records (see e.g., [6]). Our work aims at tapping into the unstructured text data in clinical notes to extract symptom information, which can be used as extra features for prediction in a statistical prediction model, thus can be regarded as complementary with the study of prediction methods. Our experiment results show that the symptom features generated using the proposed SympGraph can be leveraged to improve the accuracy of prediction of CHF disease.

# 8. CONCLUSIONS

In this paper, we present SympGraph, a general mining framework for extracting and analyzing symptoms in longitudinal clinical notes. SympGraph constructs symptom graphs based on co-location relations among detected symptom mentions. Within the general formulation of symptom graphs, we present the important clinical mining application of symptom expansion. We also present an efficient dynamic update method to incorporate updates to the underlying symptom graphs.

Our evaluation on a large-scale real world clinical database with 1 million clinical notes of over 13K patients revealed an improvement of over 31% for the symptom expansion task and that the expanded symptoms were indeed useful for predicting the onset of Congestive Heart Failure. In terms of the scalability we show that the proposed dynamic symptom expansion approach is significantly faster than the baseline Power method when multiple online queries are involved.

It is worth noting that although we mainly explored the use of the proposed SympGraph for symptom expansion at the population level. SympGraph can easily support many other mining applications, particularly due to the possibility of constructing SympGraph in multiple resolutions. Indeed, we can construct SympGraphs corresponding to any meaningful ways of aggregating patient records (e.g., at the level of individual patients, patient groups, or aggregation over different time periods or geographic locations). Moreover, in addition to symptom expansion, SympGraph can also naturally support many other tasks such as patient stratification and clustering, comparative analysis of patient groups or mine temporal symptom patterns. Given the importance and promise of applying text mining to the health domain, we believe that the proposed SympGraph can serve as a general representation framework for developing sophisticated text mining systems in health domain.

# 9. ACKNOWLEDGEMENTS

This paper is based upon work supported in part by an IBM faculty award. Research was also partly sponsored by the Army Research Laboratory and was accomplished under Cooperative Agreement Number W911NF-09-2-0053.

# 10. REFERENCES

[1] T. Bailloeul, C. Zhu, and Y. Xu. Automatic image tagging as a random walk with priors on the canonical correlation subspace. In *MIR '08*, pages 75–82, New York, NY, USA, 2008. ACM.

[2] S. Doan and H. Xu. Recognizing medication related entities in hospital discharge summaries using support vector machine. In *COLING (Posters)*, pages 259–266, 2010.

[3] G.H. Golub and C.F.V. Loan. *Matrix Computation*. Johns Hopkins, 1996.

[4] H. Harkema, I. Roberts, R. Gaizauskas, and M. Hepple. Information extraction from clinical records. In *Proceedings of the 4th UK e-Science All Hands Meeting*, 2005. Nottingham, UK.

[5] T.H. Haveliwala. Topic-sensitive pagerank. In *WWW*, pages 517–526, 2002.

[6] B.E. Himes, Y. Dai, I.S. Kohane, S.T. Weiss, and M.F. Ramoni. Prediction of chronic obstructive pulmonary disease (copd) in asthma patients using electronic medical records. *JAMIA*, 16(3):371–379, 2009.

[7] K. Jones and C.J.V. Rijsbergen. Report on the need for and provision of an "ideal" information retrieval test collection. Technical Report British Library Research and Development Report 5266, Computer Laboratory, University of Cambridge, 1975.

[8] J.M. Kleinberg. Authoritative sources in a hyperlinked environment. *Journal of the ACM*, 46(5):604–632, 1999.

[9] C. Li, J. Han, G. He, X. Jin, Y. Sun, Y. Yu, and T. Wu. Fast computation of simrank for static and dynamic information networks. In *EDBT*, pages 465–476, 2010.

[10] D. Lizorkin, P. Velikhov, M.N. Grinev, and D. Turdakov. Accuracy estimate and optimization techniques for simrank computation. *PVLDB*, 1(1):422–433, 2008.

[11] P.A. McKee, W.P. Castelli, P.M. McNamara, and W.B. Kannel. The natural history of congestive heart failure: The framingham study. *N Engl J Med.*, 285:1441–1446, 1971.

[12] S. Meystre, G. Savova, K.K. Schuler, and J. Hurdle. Extracting information from textual documents in the electronic health record: A review of recent research. *IMIA Yearbook of Medical Informatics Methods Inf Med 2008*, 2008. 47 Suppl 1:128–44.

[13] L. Page, S. Brin, R. Motwani, and T. Winograd. The pagerank citation ranking: Bringing order to the web. Technical report, Stanford Digital Library Technologies Project, 1998.

[14] A. Pathak, S. Chakrabarti, and M.S. Gupta. Index design for dynamic personalized pagerank. In *ICDE*, pages 1489–1491, 2008.

[15] W.W. Piegorsch and G.E. Casella. Inverting a sum of matrices. In *SIAM Review*, volume 32, pages 470–470, 1990.

[16] A.R. Post and J.H. Harrison Jr. Protempa: A method for specifying and identifying temporal sequences in retrospective data for patient selection. *JAMIA*, 14(5):674–683, 2007.

[17] A.D. Sarma, S. Gollapudi, and R. Panigrahy. Estimating pagerank on graph streams. *J. ACM*, 58(3):13, 2011.

[18] G.K. Savova, J.J. Masanz, P.V. Ogren, J. Zheng, S. Sohn, K.K. Schuler, and C.G. Chute. Mayo clinical text analysis and knowledge extraction system (ctakes): architecture, component evaluation and applications. *JAMIA*, 17(5):507–513, 2010.

[19] Y. Sun, Y. Yu, and J. Han. Ranking-based clustering of heterogeneous information networks with star network schema. In John F. Elder IV, Françoise Fogelman-Soulié, Peter A. Flach, and Mohammed Javeed Zaki, editors, *KDD*, pages 797–806. ACM, 2009.

[20] H. Tong, C. Faloutsos, and J.Y. Pan. Random walk with restart: fast solutions and applications. *Knowl. Inf. Syst.*, 14:327–346, March 2008.

[21] H. Tong, S. Papadimitriou, P.S. Yu, and C. Faloutsos. Proximity tracking on time-evolving bipartite graphs. In *SDM*, pages 704–715, 2008.

[22] Y. Wang. Annotating and recognising named entities in clinical notes. In *ACL/AFNLP (Student Research Workshop)*, pages 18–26. The Association for Computer Linguistics, 2009.

[23] H.Xu, S.P. Stenner, S. Doan, K.B. Johnson, L.R. Waitman, and J.C. Denny. Medex: a medication information extraction system for clinical narratives. *Journal of American Medical Informatics Association*, 17(1):19–24, Jan-Feb 2010.