SortingHat: A Framework for Deep Matching Between Classes of Entities

Joydeep
Department of Data Science
International Institute of Information technology, Bangalore
Email: joy.deep@iiitb.org

***Abstract*—Sorting hat is a framework for deep matching between different entities from different domains, rather than just their visible attributes in a given dataset. An example of this is the automatic task assignment problem where several tasks have to be assigned to people with varied skill-sets and experiences. The dataset used here is JIRA, provided by EMC India. Various types of entities (like tasks and people) and concepts (extracted from free text) from the datasets are used as the basis for deep matching.**

***Keywords*—Entity, concept, closure, focus, generatability, semantic context.**

I. Introduction

In a small organisation(say with 10 members) it's very easy to allocate tasks but in a huge organisation with thousands of employees task allocation is tedious job. In general, in a work-flow organisation, task allocation is done manually. The person in-charge of this need to have a deep understanding of the skill sets of all employees and issues resolved in the past. Based on the above knowledge the person in-charge allocates a job to a particular employee.
Better the decisions taken, effective will be the company. For example suppose a task is allocated to an employee whose skill set doesn't match with the requirements of the task make the workflow inefficient. Similarly, if task is allocated to an employee who has required skill set but is already loaded with other tasks makes the workflow inefficient again.
From the above, we can deduce that there is a requirement of intelligence, deep knowledge in order to make the workflow more efficient. By automating it, burden on the manager will reduce even if it cant make decisions automatically. This type of problem is known as "Deep matching" problem.
This paper describes work in progress, of a deep matching framework called SortingHat. SortingHat looks at past datasets of task assignments and builds a co-occurrence graph as the basic data structure for supporting deep matching.
The building blocks of the deep matching infrastructure are entities and concepts.Entities represent categorical value. Entities can be easily extracted from a dataset and has specific. Entities refer to names of elements that participate in the work-flow. These include names of people, projects, modules, documents etc. that are explicitly identified (for example, in column names) in the work-flow dataset. Concepts refer to (unidentified) keywords occurring in free-text descriptions of the work flow. For example, 'Sachin', 'Cricket' and 'India' are concepts in the text "Sachin plays cricket for India". The concepts are extracted from the free text using noun-phrase extraction mechanisms.
This paper is organized as follows. Section II talks about the literature related to the current problem and the area of text semantics. In section III, we discuss about the dataset obtained from a JIRA issue tracking system of a large corporation, which is being used as the data for proof of concept. We will identify the entities and the fields used to extract concepts from the issue tracking data. Then we will discuss about the approach we take to build the semantic engine around the JIRA data in section IV. In the same section, we also propose the generic framework for deep matching. Finally, we conclude and present the future work in section V.

II. RELATED LITERATURE

There have been large amount of work done in the area of mining semantics from datasets. Semantic matching aims at finding similarities between two documents which are not lexically explicit.Graph based techniques for semantic matching are proposed in [4], [5], [6].
Rachakonda et al [10], [11] and Kulkarni et al [9] have proposed cognitive modelling based semantic association mining methods from unstructured user generated content.

III. Data

Currently we are working on JIRA dataset which is from an issue tracking system. The dataset is provided by a huge multinational organization.The data has 149 columns for each issue entry. These columns describe many aspects of the issues. After a detailed analysis of these columns, we identified 9 fields which we treat as the entities of the data giving semantically useful information. The table I lists these 9 entities and gives a brief description about each of them.

The entities, we abstracted from the table above are connected by the concept space. The concept space is created using the concepts in the free text in each issue. The table II lists the fields from where concepts are extracted using noun-phrase extraction mechanisms and a concept space is build.

IV. Implementation

The primary data structure is a co-occurrence graph. The co-occurrence graph records pair-wise co-occurrences among

entities and concepts in the entire dataset. The co-occurrence graph represents the concept space. Formally, a co-occurrence graph representing the concept space is described as:

$$G = (V, E, w, V_L, \gamma)$$

Here V is a set of concepts or keywords identified from the dataset. $E \subseteq [V]^2$ represents pairwise co-occurrences of concepts. The term V_L refers to a set of entity classes. Some concepts are identified as referring to real-world entities that are pertinent to the system. The function $\gamma : V \to V_L$ assigns entity classes to concept names. A special class called $Concept \in V_L$ refers to an unidentified keyword, that is assigned by default for any keyword which cannot be identified as a real-world entity. The function $w : E \to Z^+$ associates every edge with a co-occurrence count and $w(t, u)$, where t, $u \in V$, represents the edge weight of the edge between t and u .

Closure X^* Given a set of terms X, their is the set of all the terms which co-occur with at least one of the terms in X.

$$X^* = \{u \mid \exists x \in X, u, x \in C\}$$

The closure of nodes c, e is the set of nodes a, b, c, d, e, f , g, h as shown in Figure (ii).

Their focus $X \perp$ is the set of all terms which co-occur with all the terms in X. Formally:

$$X_\perp = \{u \mid \forall x \in X, \{u, x\} \in C\}$$

The focus of nodes c, e is the set of nodes is c, e, f , g as shown in Figure (iii).

A set of terms X is said to be coherent if $X_\perp = \varphi$. Incoherent terms which do not share co-occurring terms are of little use in co- occurrence based semantics. In the example, terms $\{a, d\}$ do not share neighbours and are thus incoherent.In the example, terms a, d do not share neighbours and are thus incoherent (Figure (iv)).

Given a term t, its co-occurrence neighbour- hood $N(t)$ is a graph, consisting of the set of all terms co-occurring with t along with their co-occurrence counts as edge weights. In other words, it is the star like sub-graph in G originating from t. Formally:

$$N(t) = (T_N(t), C_N(t), w)$$

$$T_N(t) = \{t\} \cup \{u \mid u \in T, \{t, u\} \in C\}$$

where

$$C_N(t) = \{\{t, u\} \mid u \in T, \{t, u\} \in C\}$$

In the example, the neighbour-hood of e, N (e), is the highlighted sub graph shown in Figure (v).

The neighbourhood of a set of terms X can be defined in its two canonical forms: $N(X^*)$, the neighbourhood closure and $N(X_\perp)$ the neighbourhood focus. Formally:

$$N(X^*) = (\bigcup_{x \in X} T_{N(x)'}, \bigcup_{x \in X} C_{N(x)'}, w)$$

$$N(X^*) = (\bigcap_{x \in X} T_{N(x)'}, \bigcap_{x \in X} C_{N(x)'}, w)$$

In the example, Figure (vi) and Figure (vii) depict the neighbourhood of the closure and focus of c, e.

Given a co-occurrence graph G , the semantic context $\psi(t)$, of a term t, is the induced sub-graph of the vertices of the neighbourhood $T_N(t)$. An induced sub graph H of a graph G contains a subset of vertices of G and all edges of the form $\{v_1, v_2\}$ from G such that v_1 , $v_2 \in V(H)$, where $V(H)$ is the set of vertices in H. Formally:

$$\varphi(t) = (T_\varphi(t), C_\varphi(t), w)$$

where

$$T_\varphi(t) = T_N(t);$$

and

$$C_\varphi(t) = \{\{v_1, v_2\} \mid v_1, v_2 \in T_\varphi(t), \{v_1, v_2\} \in C\}$$

$$\varphi(X^*) = (T_\varphi(X^*), C_\varphi(X^*), w)$$

$$T_\varphi(x^*) = \bigcup_{x \in X} T_\varphi(x)$$

$$C_\varphi(x^*) = \{\{v_1, v_2\} \mid v_1, v_2 \in T_{\varphi(x^*)}, \{v_1, v_2\} \in C\}$$

$$\varphi(X_\perp) = (T_\varphi(X_\perp), C_\varphi(X_\perp), w)$$

$$T_\varphi(x_\perp) = \bigcap_{x \in X} T_\varphi(x)$$

$$C_\varphi(x_\perp) = \{\{v_1, v_2\} \mid v_1, v_2 \in T_{\varphi(x_\perp)}, \{v_1, v_2\} \in C\}$$

For example, in Figure (viii), edges $\{\{c, f\}, \{c, g\}, \{f, g\}, \{f, h\}\}$ are present in $\varphi(e)$ but not in $N(e)$.

The co-occurrence weight represents the joint co-occurrence of the pair of concepts t , u. However, it does not give information about the probability of occurrence of one term u in the context where t has occurred. We call such a probability measure as generatability of u with respect to t and represent it as $\Gamma t \to u$. Formally we calculate it as,

$$\Gamma_{t \to u} = \frac{w(t, u)}{\sum_{x \in T_{N(t)}} w(t, x)}, if u \neq t and u \in T_{N(t)}$$

In the example, the

generatability distribution of e, Γe is shown in Figure (xii).

A. *Work flow*

- From given dataset, we have considered each tuple as an issue and it is converted into an object with column names as its attributes.
- Data cleaning: keywords are retrieved using, keyword retrieval algorithm, removing stop words, removing irrelevant data.
- With these keywords co-occurrence graph is built. We are using RDBMS for storing co-occurrence graph. The graph is stored in degree table.

- Generatability is calculated for each co-occurring pair of keywords in co-occurrence graph. It is stored in generatability table.
- Random walk algorithm [2]
  Objective: identifying central topic for given set of words i.e. Topical anchor for example Given concepts in a new issue, we can predict its best suited projects, components, priorities and resolvers with the help of random walk algorithm.
  The set concepts from the new issues $C = c_1, c_2, ..., c_k$ is mapped on to the concept co-occurrence graph and a closure C^* is calculated. The C^* contains the representative entities as well. We run a random walk on C^* starting from the concepts in C and pick the highly visited entities $\{eout_1, eout_2, ..., eout_m\}$ of the mentioned output type. Each $eout_j$ belongs to a unique output entity type T_j . For each $eout_j$, we find a set of other similar entities OT_j . Finally we output a set $O = OT_1, OT_2, ..., OT_m$.

Data: Co-occurrence graph G , coherent set of terms Q, a bound on the maximum number of iterations in the random walk maxiter.

Algorithm 1: Random Walk Algorithm

Result: Topical anchor t
$if Q_\perp = \varphi then$
// Initialise query terms with seed cash and set their cash flow history to 0. Set the cash and $historyvaluesofallothernodesto0$.
$foreachnodeiinQ, letc[i] \leftarrow 1 \mid Q \mid;$
$foreachnodeiin\varphi(Q^*)\ Q, letc[i] \leftarrow 0;$
$foreachnodeiin\varphi(Q^*), letH[i] \leftarrow 0;$
$letiter \leftarrow 0;$
$whileiter < maxiterdo$
$lethistorysum \leftarrow 0;$
$foreachnodeipickedatrandomfrom\varphi(Q^*)do$
$H[i] \leftarrow H[i]$+$c[i];$
// Distribute cash to all neighbours in the semantic context.
$foreachnodejinN(i)and\varphi(Q^*)do$
$c[j] \leftarrow c[j] + (\Gamma i \rightarrow j) * c[i];$
end
$historysum \leftarrow historysum + H[i];$
$c[i] \leftarrow 0;$
end
// Normalise cash histories.
$foreachnodeiin\varphi(Q^*)do$
$H[i] \leftarrow H[i]/historysum;$
end
$iter \leftarrow iter + 1;$
end
let $t \leftarrow argmax(H[i]);$
// The most central node is chosen; this can be modified to pick the k-most central nodes. return t; end

Here generatibility is used while cash is distributed at a node instead of co-occurrence edge weights since if we use co-occurrence edge weight the nodes in the sub-graph gets an unfair amount of cash. To make it fair generatability is used.

V. Conclusion

This is an ongoing report on the project SortingHat. The models and the data from workflow systems in terms of entities and concepts are under implementation phase.

Acknowledgement

I would like to thank Prof. S.Srinivasa, Mr. S. Kulkarni for providing me the opportunity to work on this project and for providing valuable suggesting while moving forward in this project.

References

[1] Sumant Kulkarni, Srinath Srinivasa, Jyotiska Khasnabish, Kartikay Nagal, Sandeep Kurdagi , *SortingHat: A Framework for Deep Matching Between Classes of Entities*

[2] Aditya Rachakonda, Srinath Srinivasa, *Mining Semantics From Unstructured Text Using A Cognitive Model On Lexical Co-Occurrences*

[3] http://www.wikipedia.com

[4] http://www.gopivotal.com

[5] https://archive.ics.uci.edu/ml/datasets/Legal+Case+Reports

TABLE I. THE DIFFERENT ENTITIES IN JIRA DATASET.

Entity	Description
Project	The name of the project.
Issue	The Issue Id of the issue.
Resolver	The person who resolved the issue.
Resolution	State of the issue after resolution (Not a Bug, Duplicate, Fixed etc.,).
Component	Name of the project component.
Time Taken	Time taken to fix issue.
Issue Priority	Importance of the issue.
Classification	Type of the fix (API, Code etc.,).
Root Cause	Reason for the issue to arise.

Fig. 1.

TABLE II. THE DIFFERENT FIELDS CONTRIBUTING TO CONCEPT SPACE IN JIRA DATASET.

Column Name	Description
Summary	Short description of the issue.
Description	Detailed description of the issue.
Customer Facing Description for Release Notes	The message passed to the customer to convey the existence of the issue. It describes how the customer can observe the issues.

Fig. 2.

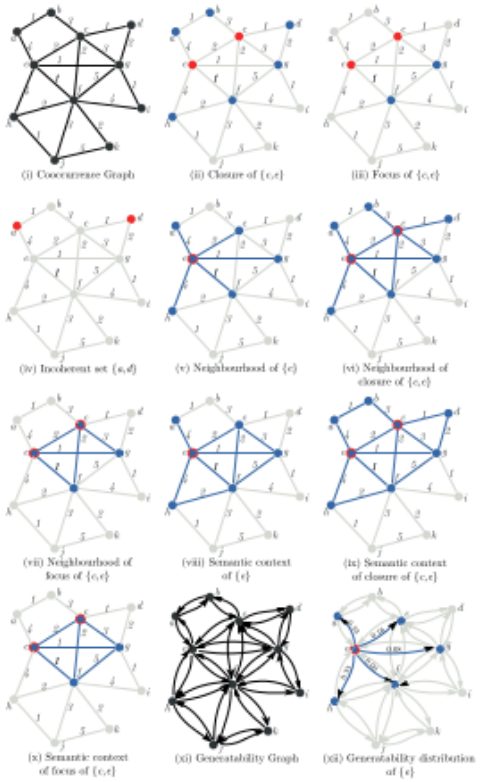

Fig. 3. primitives