Identification of Karaka relations in an English sentence

Sai Kiran Gorthi, Ashish Palakurthi, Radhika Mamidi, Dipti Misra Sharma
International Institute of Information Technology - Hyderabad
{saikiran.gorthi, ashish.palakurthi}@research.iiit.ac.in
{radhika.mamidi, dipti}@iiit.ac.in

Abstract

In this paper we explain the identification of karaka relations in an English sentence. We explain the genesis of the problem and present two different approaches, rule based and statistical. We briefly describe about rule based and focus more on statistical approach. We process a sentence through various stages and extract features at each stage. We train our data and identify Karaka relations using Support Vector Machines (SVM). We also explain the impact of our work on Natural Language Interfaces for Database systems.

1 Introduction

In any sentence, it is important that the relations between words are expressed properly in order to understand it. There are many widely used relations between words like subject and object relations, agent-patient of thematic relations. These are commonly used relations that we come across. The Computational Paninian Grammar (CPG) framework uses Karakas to express relations between words in a sentence. Karakas are classes with which relations between words in a sentence can be expressed. One of the key features of the karaka relations is that they are verb centric. All the relations are dependent on the main verb of the sentence. The advantage of using Karakas is that they provide proper mapping between verb and its relations using a very small set of tags. By providing a good understanding of the syntactic as well as the semantic relations, they help in resolving ambiguities posed by a sentence.

There are around 24 Karakas in total, but we generally take into consideration only 6 Karakas which are frequently occurring (75%) in most of the sentences. They are karta (k1), karma (k2), karana (k3),sampradaan (k4), apadaan(k5) and adhikarana (k7). Karta is the doer of the action. It is independent of the other Karakas. Karma is the object of the action or verb. Karana helps in completing action. It acts as an instrument in the completion of the action. The beneficiary of the action is known as sampradana and the source of the action is called apadaan. There are three types of k7 Karakas namely k7t, k7p and k7. k7t denotes the time of action, whereas k7p denotes location of the doer or patient at the time of action. k7 represents location in topic. In the example, John gave the book to Clarke, we have John, who is the doer of the action as karta(k1), bag, which is the object of the action as karma(k2), Clarke, the beneficiary or sampradaan(k4) of the action. Let us take another example, He played cricket yesterday. Here yesterday is k7t. In this way, we can get different relations expressed for different sentences based on the verb.

NLIDB[1] system converts an input natural language query into an SQL query and then gets the answer from respective database by running the query on it. The NLIDB system proposed by (Gupta et al., 2012) requires two stages in the processing of the input query. The first stage is the syntactic stage and the second stage is the semantic stage.

In the syntactic stage, they give the input sentence to Stanford dependency parser. Once they get Stanford dependencies, they map them to karaka relations. These identified karakas are then used in the semantic stage to build semantic frames. This is where our work comes into effect. By identifying karaka relations in an English sentence, we address the issue of identifying karaka relations in the syntactic stage more effectively.

[1] Natural Language Interface for Databases

2 Related work

Significant work happened in the identification of karaka relations in Indian languages. This is the first attempt to work on identifying karaka relations in English. (Bharati et al., 2008) demonstrated on constraint based parsing for free word order languages. (Vaidya et al., 2009) showed that it is possible to apply CPG to English. We propose a statistical method for identifying karaka relations from a given sentence using machine learning.

The remainder of this section is organized as follows. In section 3, we explain our different approaches. In section 4, we discuss about the experiments and results. In section 5, we discuss the impact of our system. In section 6, we conclude and discuss about future work.

3 Approach

3.1 Rule-based approach

The following are the steps we followed for identifying Karaka relations in English using rule based approach:

- As a first step, we analyzed the queries we made for the NLIDB system of our university courses portal.
- We generated the Stanford parse for each query and tried to map each dependency relation to a CPG relation.
- We then generated a rule set to to automate the mapping from Stanford dependencies to CPG relations.
- We developed a preference-based allotment system for the purpose of mapping.
- After parsing each query from the Stanford parse, we take the dependency relations from it and check them from the rules table in column2.
- If we find a relation, we map it to its corresponding CPG label. We continue this process for each Stanford dependency relation.

The overall performance of the system with the method above is 52%. We adopted a statistical approach for better results.

Karaka relation	Stanford dependency relation
k1	nsubj (subject)
k2	dobj (object)
k3	prep_with
k4	prep_to
k5	prep_from
r6	poss
k7	prep_at

Table 1: Basic rules for mapping

4 Statistical approach

4.1 Stage 1 $DataExtraction$

English Dependency Tree bank data is in SSF[2] format. We processed each sentence and extracted all of its relations(child node, parent node, karaka relation).We got 2557 such relations.

4.1.1 Example

4.1.1.1 SSF Sentence id='2' $1((NPfsaf =' 'drel =' k1 : VG'name =' NP' 1.1BipashaNNPfsname =' Bipasha'))2((VGfsaf =' 'name =' VG' 2.1stormsVMfsaf =' storm, n, m, p, 3, 0 'name =' storms' \|fsaf =' storm, v, m, s, 3, 0 'tense =' PRES'))3((PRTfsname =' PRT'drel =' pof_idiom : VG' 3.1outRPfsaf =' out, p, m, s, 3, 0 'name =' out' \|fsaf =' out, n, m, s, 3, 0 ' \|fsaf =' out, adj, m, s, 3, 0 ' \|fsaf =' out, D, m, s, 3, 0 ' \|fsaf =' out, v, m, s, 3, 0 ' \|fsaf =' out, p, m, s, 3, 0 '))4((PPfsname =' PP'drel =' k5 : VG' 4.1ofINfsaf =' of, p, m, s, 3, 0 'name =' of' 4.2((NPfsname =' NP2' 4.2.1filmNNfsaf =' film, n, m, s, 3, 0 'name =' film' \|fsaf =' film, v, m, s, 3, 0 ' 4.2.2festivalNNfsaf =' festival, n, m, s, 3, 0 'name =' festival' 4.2.3GoaNNfsaf =' goa, n, m, s, 3, 0 'name =' Goa'))))5.SYMfsaf =' ., punc, n, s, 3, 0 'name =' .' /Sentence$

4.1.1.2 Extracted data

4.1.1.2.1 Sentence Bipasha storms out of film festival Goa.

[2]Shakti Standard Format

4.1.1.2.2 Relations Bipasha, storms, k1
out of... , storms, pof_idiom
Goa, storms, k5

4.2 Stage 2 $DependencyParse$

Having extracted English sentence for each relation, we got the Stanford Dependency parse structure for the sentence. We then run a parallel check for the same sentence in the karaka relation we extracted in the first step. For a matching relation(child node, parent node), we add the corresponding verb of the relation(from Stanford Dependency Tree) and Stanford Dependency label to the relation and update as (verb of the relation, child node, parent node, Stanford dependency relation, karaka relation). Lets call it feature set.

4.2.1 Examples

storms, Bipasha, storms, nsubj, k1
storms, out of... , storms, prep_out_of, pof_idiom

4.3 Stage 3 $RootVerbExtraction$

We also extract the verb with respect to a relation in the format aforementioned. We take this verb from the feature set and replace with it's root verb using Morphadoner(Burns, 2013).

4.3.1 Example

declared - declare
said - say

4.4 Stage 4 $VerbClassification$

Beth Levin defined classes for verbs(Levin, 2011) based on their action. Each verb could fall into different categories depending on their varied action. Two verbs having the same set(list of different classes they belong to) could be considered as similar verbs. Using this classification, we built a dictionary (key : verb, value :class_number). Using this dictionary, we replace the root verb in the feature set with its class number.

4.4.1 Example

abandon - 1
abate - 3
desert - 1

4.5 Stage 5 $POSTagging$

The first two features we considered are the two words in the relation. The data is so sparse that the frequency of any noun in the entire dataset is frequently 1, very less, when we're dealing with a machine learning application which needs good density in the data point distribution. Since the data is very sparse and considerably less, we replaced the words in the relation with their POS tags. This step ensure that the features are more specific and suitable for training. We anyway get the stanford parse, so it's POS tags could also be used.

4.5.1 Example

Bipasha, storms : nsubj, VG

4.6 Stage 6 $FeatureEnumeration$

The feature set is ready. Now we need to make an appropriate format of the features. Machine Learning methods usually take features as numbers(each feature is a dimension).
From the annotated data, we've the result that these four features map to. We enumerate that too.

4.6.1 Example

Fx is Feature #x, OP is Output for the relation
{F1 F2 F3 F4 OP} is feature set
{685 VBD PRP nsubj k1} correspond to {1 1 1 1 1}.
685 is the verb class. It's given #1 because it comes first in the data. All other instances of 685, VBD, PRP, nsubj, k1 in the data also get the same numbers as these.

4.7 Stage 7 $Cross - Validation$

Using Support Vector Machines(Chang and Lin, 2011) and cross-validation method, we trained and tested on the enumerated feature set.

We experimented by dividing the data into 1:4, 3:7 and picked the best accuracy.

5 Experiments and results

Our training set consists of 18,345 tokens. Our dataset consisted of 1479 sentences. We divided our training sets into 4 equal subsets. One of the subsets is kept as a test set and we use the remaining three subsets for training. We repeated this

process for 4 times to get an average result, which is called the 4-fold cross validation. After conducting many experiments and applying the grid-search method, we found that c=8.0 and g=0.125 were the values giving the best results. We experimented on the following kernels:

1. Polynomial Kernel.
2. Radial Basis Function kernel (RBF).
3. Linear Kernel.

Polynomial		RBF		Linear	
d	**A**	**g**	**A**	**c**	**A**
1	73.28%	0.125	74.14%	6.0	74.42%
2	74.26%	0.3	73.21%	8.0	74.14%
3	72.54%	0.6	73.23%	10.0	73.83%
4	67.10%	0.8	70.78%	12.0	73.75%

Table 2: Results

In the results table, d is the degree of the polynomial kernel.
g, c are the gamma and cost parameters respectively.

6 Impact

We used our module in the NLIDB system in the syntactic stage. We investigated our results over 213 natural language queries. Our module was found to be effective in improving the accuracy of the 'Stanford dependency to karaka mapping'. The error rate of the syntactic stage decreased by 4.79%, thus improving the overall performance of the NLIDB system.

7 Conclusions and Future work

This paper presents a statistical approach and a brief overview of rule based approach for identification of karaka relations in English. We would go further to explore more features and experiment with larger data-sets. We shall include many semantic features so that mapping becomes more compact.

Acknowledgements

We thank Yeka Jayendra Rakesh for his useful insights and Shalaka Vaidya for helping us with the extraction of data.

References

Akshar Bharati, Samar Husain, Dipti Misra Sharma, and Rajeev Sangal. 2008. A two-stage constraint based dependency parser for free word order languages. In *Proceedings of the COLIPS International Conference on Asian Language Processing 2008 (IALP)*.

Philip R Burns. 2013. Morphadorner v2: A java library for the morphological adornment of english language texts. *Northwestern University, Evanston, IL*.

Chih-Chung Chang and Chih-Jen Lin. 2011. Libsvm: a library for support vector machines. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 2(3):27.

Abhijeet Gupta, Arjun Akula, Deepak Malladi, Puneeth Kukkadapu, Vinay Ainavolu, and Rajeev Sangal. 2012. A novel approach towards building a portable nlidb system using the computational paninian grammar framework. In *Asian Language Processing (IALP), 2012 International Conference on*, pages 93–96. IEEE.

Beth Levin. 2011. Verb classes within and across languages. *Handout, Leipzig, April*.

Ashwini Vaidya, Samar Husain, Prashanth Mannem, and Dipti Misra Sharma. 2009. A karaka based annotation scheme for english. In *Computational Linguistics and Intelligent Text Processing*, pages 41–52. Springer.