PHRASE SENSE DISAMBIGUATION

Submitted in partial fulfilment of the requirements
of the degree of

Bachelor of Engineering in Computer Engineering

By

Rushabh Mishra (14102A0070)
Suryateja Gudiguntla (14102A0063)
Vedant Mahabaleshwarkar (14101A0062)

Under the Guidance of

Prof. Pankaj Vanwari

Department of Computer Engineering

Vidyalankar Institute of Technology

Wadala (E), Mumbai-400037

University of Mumbai

2017-18

CERTIFICATE OF APPROVAL

This is to certify that the project entitled

“Phrase Sense Disambiguation”

is a bonafide work of

Rushabh Mishra (14102A0070)

Suryateja Gudiguntla (14102A0063)

Vedant Mahabaleshwarkar (14101A0062)

submitted to the University of Mumbai in partial fulfillment of the requirement for the award of the degree of

Undergraduate in **“Bachelor of Engineering in Computer Engineering”**.

Guide	Head of Department	Principal
(Prof. Pankaj Vanwari)	(Dr. Arun Chavan)	(Dr. S.A. Patekar)

Project Report Approval for B. E.

This project report entitled ***Phrase Sense Disambiguation*** by

1. ***Rushabh Mishra (14102A0070)***
2. ***Suryateja Gudiguntla (14102A0063)***
3. ***Vedant Mahabaleshwarkar (14101A0062)***

is approved for the degree of ***Bachelor of Engineering in Computer Engineering.***

Examiners

1.--
--

2.--
--

Date:

Place:

Declaration

I declare that this written submission represents my ideas in my own words and where others' ideas or words have been included, I have adequately cited and referenced the original sources. I also declare that I have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in my submission. I understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

	Name of Student	Roll No.	Signature
1.	Rushabh Mishra	(14102A0070)	
2.	Suryateja Gudiguntla	(14102A0063)	
3.	Vedant Mahabaleshwarkar	(14101A0062)	

Date:

Acknowledgements

We take this occasion to thank Vidyalankar Institute of Technology's Department of Computer Engineering for giving us an opportunity to work on this project as part of our B.E. Final year.

We extend our most sincere and heartfelt thanks to our guide, Prof. Pankaj Vanwari for providing us with the valuable guidance at the crucial junctures and advising us throughout the year. We extend our sincere thanks to our respected head of the department, Dr. Arun Chavan, for allowing us to use the facilities available.

We thank our parents for their continued support and patience throughout our engineering education which culminates into our final year project as part of the bachelor's degree.

Lastly, we thank our classmates and peer for the support and encouragement that they have provided us during our work.

Abstract

From the time the first computer was invented, the development of computers has always evolved in a way as to make the interaction between humans and computers feel natural. At each stage in the development process of computers, users have always had to follow a certain language syntax to make sure the machine understands what the user is trying to communicate. This is because the machine does not understand that a phrase in a sentence can mean various things depending on the context, which is known as the ambiguity problem. Our system aims to develop a knowledge base that smart systems can use to try to understand and resolve ambiguity in predicate phrases. This will help the system understand natural human language far easily than the current scenario.

There is a vast amount of knowledge already available on the internet in the text format. This is available via Wikipedia, DBpedia, online news articles, tweets etc. Our project plans to use this textual information to build a knowledge base. This knowledge base will store thousands of phrases and their meaning in various contexts. This will help enrich the vast plethora of knowledge available on the internet.

The knowledge base we develop will have most phrases and their contextual meaning stored in a format understandable to computers. Computers and Smart Systems like Google Assistant, Siri, Amazon Alexa can use this knowledge base to get a probabilistic meaning of any unknown phrases the user may say. This will help computers disambiguate the natural language input and make human machine interaction seem more natural.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

In this chapter, we define and explain the problem that we have taken up as our project. Our project falls under the domain of Natural Language Processing, so we have mentioned the ongoing problems in the field, some of which are easily or mostly solved, while some problems need extensive research and work. We have also detailed the objectives we aim to achieve from this project.

1.1. Phrase Sense Disambiguation

In computational linguistics, word-sense disambiguation (WSD) is an open problem of natural language processing and ontology. WSD is identifying which sense of a word (i.e. meaning) is used in a sentence, when the word has multiple meanings. The solution to this problem impacts other computer-related writing, such as discourse, improving relevance of search engines, anaphora resolution, coherence, inference, etc.

But, there is one persisting problem with these systems. Currently, no smart system can understand the ambiguity in phrases and try to resolve them. WSD fails when trying to understand the context of a discourse. The same phrase can be used in multiple sentences, and it can have a different meaning in those sentences. The meaning of phrases is often dependent on the type of entities it is used with, this is something that WSD fails to decipher.

1.2. Open problems in NLP

While surveying QA technologies for our project, we encountered the problem of making a machine understand not only the structure(syntax) of a natural language like English but also understand the meaning(semantics) of the discourse.

Following are some interesting open problems in NLP which need to be solved to achieve language understanding in machine. They are classified into three broad categories viz.

1. Easy or problems that are mostly solved,
2. Intermediate or problems in which we are making satisfactory progress,

3. Hard or still needs lot of work to gain significant success.

This is by no means a comprehensive list and we have included only the problems that are either relevant to our work or are currently undergoing significant research work.

1.2.1. Easy or mostly solved

In this subsection, we look at NLP problems that have been solved accurately. The solutions to these problems are vital to the more extensive problems in the field that are yet to be solved.

1.2.1.1. Spam detection

Spam emails are the emails that the receiver does not wish to receive [1]. A spam filter is a program that is used to detect unsolicited and unwanted email and prevent those messages from getting to a user's inbox.

1.2.1.2. Parts of Speech Tagging

Profits soared at Boeing Co., easily topping forecasts on Wall Street, as their CEO Alan Mulally announced first quarter results.

Profits/N soared/V at/P Boeing/N Co./N ,/, easily/ADV topping/V forecasts/N on/P Wall/N Street/N ,/, as/P their/POSS CEO/N Alan/N Mulally/N announced/V first/ADJ quarter/N results/N./.
KEY: N = Noun, V = Verb, P = Preposition, Adv = Adverb [2]

1.2.1.3. Named Entity Recognition

Profits soared at Boeing Co., easily topping forecasts on Wall Street, as their CEO Alan Mulally announced first quarter results.

Profits/NA soared/NA at/NA Boeing/SC Co./CC ,/NA easily/NA topping/NA forecasts/NA on/NA Wall/SL Street/CL ,/NA as/NA their/NA CEO/NA Alan/SP Mulally/CP announced/NA first/NA quarter/NA results/NA ./NA
KEY: NA = No entity, SC = Start Company, CC = Continue Company, SL = Start Location, CL = Continue Location [3]

1.2.2. Intermediate or making satisfactory progress

These are the problems which have a lot of research to support the fundamentals of getting to the solution of the problem, but a universally accepted solution for these problems is not yet agreed upon.

1.2.2.1. Sentiment Analysis

Best roast chicken in San Francisco! – Positive
The waiter ignored us for 20 minutes. – Negative [4]

1.2.2.2. Anaphora and Coreference resolution

"Roy eats dinner. He is hungry. Sam eats with him. She is hungry too." To resolve 'He' and 'him' to 'Roy' and 'She' to 'Sam'.
"Carter told Mubarak he shouldn't run again."
To resolve whether 'he' is related to 'Carter' or 'Mubarak'.
Paul McCartney performed at Madison Square Garden. The rock-n-roll singer was seen by thousands of devoted fans.
To resolve 'The rock-n-roll singer' to 'Paul McCartney'. [5]

1.2.2.3. Word sense disambiguation

I need new batteries for my mouse. – 'mouse' is ambiguous here. Can mean a computer mouse or a toy mouse.
Roy is a German teacher.
Can mean that Roy's nationality is German, or Roy teaches German language. Without further information it is ambiguous even for a human. [6]

1.2.2.4. Parsing

The basic problem of parsing sentences. [7]
My dog also likes eating sausage.
Parse:

```
(ROOT (S
(NP (PRP$ My) (NN dog)) (ADVP (RB also))
(VP (VBZ likes) (S
(VP (VBG eating)
(NP (NN sausage)))))
(. .)))
```

Dependency Parse:

```
nmod:poss(dog-2, My-1) nsubj(likes-4, dog-2) advmod(likes-4, also-3) root(ROOT-0, likes-4)
xcomp(likes-4, eating-5) dobj(eating-5, sausage-6)
```

1.2.2.5. Machine Translation

Translating sentences from one language to another.

English	=	French
How are you today?	=	*Comment vas-tu aujourd'hui?*

1.2.2.6. Information Translation

To take a text as input and represent it in a structured form like a database entry.

1.2.3. Hard or still needs lot of work

These are the problems which are still in their research phase and will take a lot of time and efforts being put into them to get to commercially viable solutions.

1.2.3.1. Text Summarization

To take input as text document(s) and try to condense them into a summary.

1.2.3.2. Machine dialog system

User - I need a flight from New York to London, arriving at 10 pm? System - What day are you leaving?
User - Tomorrow.
System detects the missing information in your sentences.

1.2.3.3. Information Retrieval

A sentence can be construed differently depending on how the relation between entities is expressed. Every relation, verb phrase or action can be expressed in many ways. These expressions are often ambiguous and evolved from historic usage. Traditional parsing and pos tagging may fail to interpret it correctly. [8]

For example, the phrase *'Natwarlal gets 10 years for fraud'* is synonymous to getting sentenced to 10 years in prison for fraud. However parsing and annotating pos tags to the sentence could interpret it as the person has gotten 10 years for committing fraud.

1.3. RDF Triples

We have come a long way from command line interfaces to various versions of graphical user interfaces and now to voice interactions between virtual assistants. This practice of following a syntax, or structuring human machine interaction, in a machine understandable manner is what makes the interaction feel unnatural.

The Resource Description Framework (RDF) [9] is a framework for representing information in the Web. The design of RDF is intended to meet the following goals:

- having a simple data model
- having formal semantics and provable inference
- using an extensible URI-based vocabulary [10]
- using an XML-based syntax
- supporting use of XML schema datatypes
- allowing anyone to make statements about any resource

The underlying structure of any expression in RDF is a collection of triples, each consisting of a subject, a predicate and an object. A set of such triples is called an RDF graph.

Triples are a format used by popular knowledge bases like Yago [11] and DBpedia [12] to define relationships between known entities.

A triple contains three fields.

Subject –> Predicate –> Object

The direction of the arc is significant: it always points toward the object. Here the Subject and Object are known entities/entity categories and the predicate defines the relationship between them.

1.4. Aim of Phrase Sense Disambiguation

The aim of our project is primarily to build a knowledge base. The purpose of this knowledge base will be to enrich the information already available on the internet in the form of text. Each sentence will be converted into a triple format as (subject, predicate, object). The knowledge base will contain a set of valid English phrases that can be used between a subject and object of a given category. Thus, there will be sets of predicates generated between every category of subject and object present in the Wikipedia database.

Objectives:

- To make triplets from sentences in Wikipedia articles that have a subject and object which are already linked to each other using a predicate in the DBpedia tree.
- To generate a set of valid phrases that map to the semantic meaning of a predicate that links a subject category and object category in the DBpedia tree.
- To assign weights to phrases in a predicate set, and thus assign weights to multiple predicates between the same subject and object category to build a probabilistic model.
- To take such input from the user that has one or two out of subject, object, predicate as ambiguous and try to resolve that ambiguity using the knowledge base and weights.

Chapter 2

Review of Literature

In this chapter we look at some existing systems that our project referred to get an understanding of the workings of an NLP system. Reading existing literature helped us understand NLP terminologies and the accepted standards of data representation in this field.

2.1. Existing Systems

We have written about two existing projects that work on RDF graph building and WSD.

2.1.1. Semantic Web Machine Reading with FRED

A machine reader is a tool able to transform natural language text to formal structured knowledge so as the latter can be interpreted by machines, according to a shared semantics. [13] FRED is a machine reader for the semantic web: its output is a RDF/OWL graph, whose design is based on frame semantics. Nevertheless, FRED's graph are domain and task independent making the tool suitable to be used as a semantic middleware for domain- or task- specific applications.

2.1.2. Word Sense Disambiguation using WordNet

WordNet is a commonly used English lexical database. [6] The algorithm is applied to the data scraped from Wikipedia articles. The representative context used in the algorithm is extracted from the Wikipedia pages of the words belonging to the category. The lexical category of the given word is determined using the words in its neighboring context. After finding the lexical category of the word, the correct sense is found using a modified version of Lesks Algorithm.

2.2. Preprocessing Data

Natural Language Processing projects need input data in a certain format to enable efficient processing. Readily available data from various sources is often not in the format needed and needs cleanup for it to become suitable for use. We look at some systems our project has used for the preprocessing of data to bring it into a form our project found suitable.

2.2.1. S.T.A.R. – NL: Simplification of Text & Anaphora Resolution in Natural Language

It outputs an answer in form of a set of objects (a single noun or a collection). STAR performs four tasks viz. pronominal anaphora resolution, text simplification, document processing and question processing. [14] Text simplification attempts to decompose long sentences into smaller, simpler ones without loss of information. Pronominal anaphora resolution links the pronouns appearing in the text with their respective subjects, usually nouns. Document processing converts the English text, now free of pronouns, to a structured form and stores it into a knowledge-base.

2.2.2. Entity Extraction: From Unstructured Text to DBpedia RDF Triples

This system is based on a pipeline of text processing modules that includes a semantic parser and a coreference solver. [15] By using coreference chains, we group entity actions and properties described in different sentences and convert them into entity triples. Applied the system to over 114,000 Wikipedia articles and could extract more than 1,000,000 triples. Using an ontology-mapping system that was bootstrapped using existing DBpedia triples, we mapped 189,000 extracted triples onto the DBpedia namespace.

2.2.3. DBpedia

DBpedia is a crowd-sourced community effort to extract structured content from the information created in various Wikimedia projects. [12] This structured information resembles an open knowledge graph (OKG) which is available for everyone on the Web. A knowledge graph is a special kind of database which stores knowledge in a machine-readable form and provides a means for information to be collected, organized, shared, searched and utilized. Google uses a similar approach to create those knowledge cards during search. We hope that this work will make it easier for the huge amount of information in Wikimedia projects to be used in some new interesting ways. The English version of the DBpedia knowledge base describes 4.58 million things, out of which 4.22 million are classified in a consistent ontology, including 1,445,000 persons, 735,000 places (including 478,000 populated places), 411,000 creative works (including 123,000 music albums, 87,000 films and 19,000 video games), 241,000 organizations (including 58,000 companies and 49,000 educational institutions), 251,000 species and 6,000 diseases.

2.3. Input Sentence Processing

Just like source data needs clean up, even user given input needs to be processed to bring it into a format that the system understands. We have written about a system that we used for input sentence processing.

2.3.1. Tagme - Fast and Accurate Annotation of Short Texts with Wikipedia Pages

Several recent software systems have been designed to obtain novel annotation of cross-referencing text fragments and Wikipedia pages. [16] Tagme is state of the art in this setting and can accurately manage short textual fragments (such as snippets of search engine results, tweets, news, or blogs) on the fly.

Chapter 3

Design and Analysis

Every project needs a feasibility study to ensure that the project is feasible. For the smooth completion of the project, defining a process model and following it is essential. This chapter mentions in detail the process model used for our project and the feasibility study done by the project members.

3.1. Process Model

Process model defines the framework for work done in the project. Each project has a different process model depending on the size and characteristics of the project. In this subsection, we have written about the process model used in our project.

3.1.1. Spiral Model

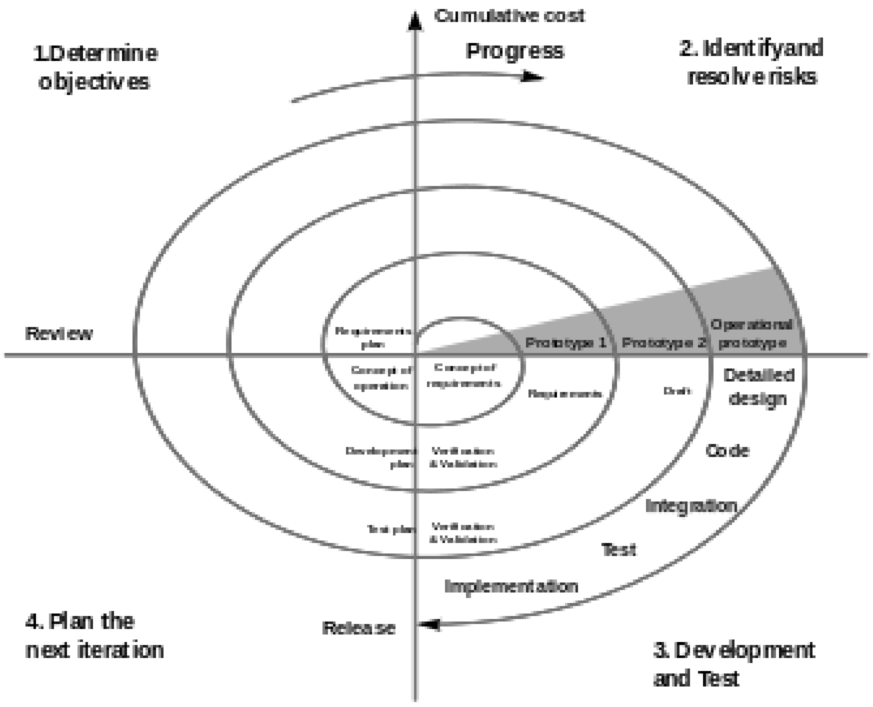


Fig 1: Spiral Model

The Spiral Life Cycle Model [17] is a type of iterative software development model which is generally implemented in elevated risk projects. In Spiral Model, we can arrange all activities in the form of a spiral.

Each loop has 4 sections:

1. Identify objectives-

We try to understand the project objective, alternatives in design and constraints imposed because of cost, technology, schedule, etc.

2. Risk Analysis –

We try to find which other approaches can be implemented in order to fulfil the identified constraints. Operational and technical issues are addressed. Risk mitigation is of focus in this phase, Evaluation of risk factors determine the future action.

3. Development & Testing –

Here we develop the planned product. Testing is carried out. Development can be carried out in waterfall approach.

4. Plan the next phase –

Here we review the progress and judge it considering all parameters. Issues which need to be resolved are identified and necessary steps are taken for next iteration.

Subsequent loops of Spiral Model involve similar phases. Analysis and engineering efforts are applied in this model. If at any point the risk involved in the project is lot more than anticipated, it can be aborted at that phase. Reviews at each phase of the product in the spiral can be done by an in-house expert or an external client.

Spiral Model is also called meta-model because in a way it comprises of other models of SDLC. Both waterfall and prototyping models are used in it. We do software development systematically over loops (adhering to waterfall approach) and at the same time we make prototype after completion of each spiral cycle (adhering to prototyping model). This way we reduce risks and follow a systematic development approach.

3.2. Feasibility Study

Feasibility study is about the viability of a system, how beneficial the development of such a system is for the organization. The proposed system has to be examined for its technical, cultural, schedule, legal, economical and operational feasibility. Many alternatives are found and the best among them, which suits our requirement in a better way, is chosen.

One should keep following points in mind to choose a better alternative.

- Greater speed of processing
- Platform independent
- Less memory occupancy
- Effective procedures eliminating errors
- Fast retrieval of data
- Efficient way to store data
- Ease of use

These alternatives should be considered, and a better system is to be designed.

1. Technical feasibility:

   The technical feasibility assessment is focused on gaining an understanding of the present technical resources of the organization and their applicability to the expected needs of the proposed system. It is an evaluation of the hardware and software and how it meets the need of the proposed system.

2. Economic feasibility:

   The purpose of the economic feasibility assessment is to determine the positive economic benefits to the organization that the proposed system will provide. It includes quantification and identification of all the benefits expected.

3. Legal feasibility:

   Determines whether the proposed system conflicts with legal requirements, e.g. a data processing system must comply with the local data protection regulations.

4. Operational feasibility:

   Operational feasibility is a measure of how well a proposed system solves the problems and takes advantage of the opportunities identified during scope definition and how it satisfies the requirements identified in the requirements analysis phase of system development.

5. Schedule feasibility:

A project will fail if it takes too long to be completed before it is useful. Typically, this means estimating how long the system will take to develop, and if it can be completed in a given time using some methods like payback period. Schedule feasibility is a measure of how reasonable the project timetable is.

3.2.1. Time Feasibility

Our project is a software project. Thus, time required is for following phases

- Designing the system.
- Coding.
- Debugging.
- Testing the system.

Expected time to complete the project is 6-7 months.

3.2.2. Feasibility Analysis Matrix

	Weight	Candidate 1	Candidate 2	Candidate 3
Description		Developing it by outsourcing	Developing inhouse	Developing inhouse with consultant
Operational Feasibility	10%	Supports user requirements with delayed response. Score:70	Supports user requirements with quick response. Score:100	Supports user requirements with quick response. Score:100
Technical Feasibility	10%	Requires hiring people from outside. Score:60	All experts are inhouse. Score:100	All experts are inhouse. Score:100
Cultural Feasibility	5%	feasible	feasible	feasible
Economic Feasibility	30%	Approx. Rs.1,50,000/-	Approx. Rs. 50,000/-	Approx. Rs.1,00,000/-
Payback Period	20%	Approx. 10 years	Approx. 5 years	Approx. 7 years
Schedule Feasibility	10%	6 months (90)	6-7 months (70)	6 months (70)
Legal Feasibility	10%	80	80	80
Weighted Score		77.5	88.7	75.2

Table 1: Feasibility Analysis Matrix

3.2.3. Cost Analysis

Cost benefit analysis is a term that refers both to helping, to appraise or assess the case for a project program or policy proposal.

Estimation

The functional user requirements of the software are identified and each one is categorized into one of the 5 types: outputs, inquiries, inputs, internal files, and external interfaces. Once the function is identified and categorized into a type, it is then assessed for complexity and assigned several function points.

Each of these functional user requirements maps to an end user business function, such as a data entry for an input of a user query for an inquiry. This distinction is important because it tends to make the functions measured in function points map easily into user-oriented requirements, but it also tends to hide internal functions (e.g. Algorithms), which also require resources to implement, however, there is no ISO recognized FSM method that includes algorithmic complexity to deal with this weakness, implemented in several commercial software products.

The function point method was developed by Albrecht for IBM. A function point is a rough estimate of a unit of delivered functionality of a software project. Counts are made for the following categories:

- Number of user inputs
- Number of user outputs
- Number of user inquiries
- Number of files
- Number of external interfaces

Each count is multiplied by its corresponding complexity weight and the results are summed to provide the UFC.

The adjusted function point calculation is calculated by multiplying the UFC by a technical complexity factor (TCF) also referred to as Value Adjustment Factor (VAF). Components of the TCF are listed in the following table:

Measure parameter	Weighting factor			
	Simple	Average	Complex	Values for our project(pi)
No. of user inputs	3	4	6	2
No. of user outputs	4	5	7	1
No. of user inquiries	3	4	6	2
No. of files	7	10	15	5
No. of external interfaces	5	7	10	2

Table 2: Components of TCF

Count = 85

The complexity weighing factor is assumed to be average.

Now the system complexity can be computed by answering following questions. These are complexity adjustment factors.

General System Characteristics		Brief Description	Rating
1.	Data Communication	Are data communications required?	4
2.	Distributed Data Processing	How are processing functions handled?	5
3.	Performance critical	Is performance of the system critical?	4
4.	Heavily used configuration	Will the system run in an existing, heavily utilized operational environment?	4
5.	Multiple sites	Is the system designed for multiple installations in different organizations?	5
6.	Online data entry	Does the system require online data entry?	4
7.	Reliable backup and recovery	Does the system need reliable backup and recovery?	5
8.	Online update	Are the master files updated online?	4
9.	Complex processing	Is internal processing complex?	4
10.	Reusability	Is the code which is designed being reusable?	5

11.	Installation	Are conversion and installation included in the design?	4
12.	Operational ease	How effective and/or automated are startup, backup, and recovery procedures?	4
13.	Complexity	Are the inputs, outputs, files or enquiries complex?	5
14.	Facilitate change	Is the application designed to facilitate change ease by the user?	5
Total			62

Table 3: Complexity Adjustment Factors

Sum (Fi) = **62**

Finally, FP estimation is derived as follows:

Function point (FP) = Count total * (0.65 + (0.01 * Sum (Fi))

= 85 * (0.65 + (0.01 * 62))

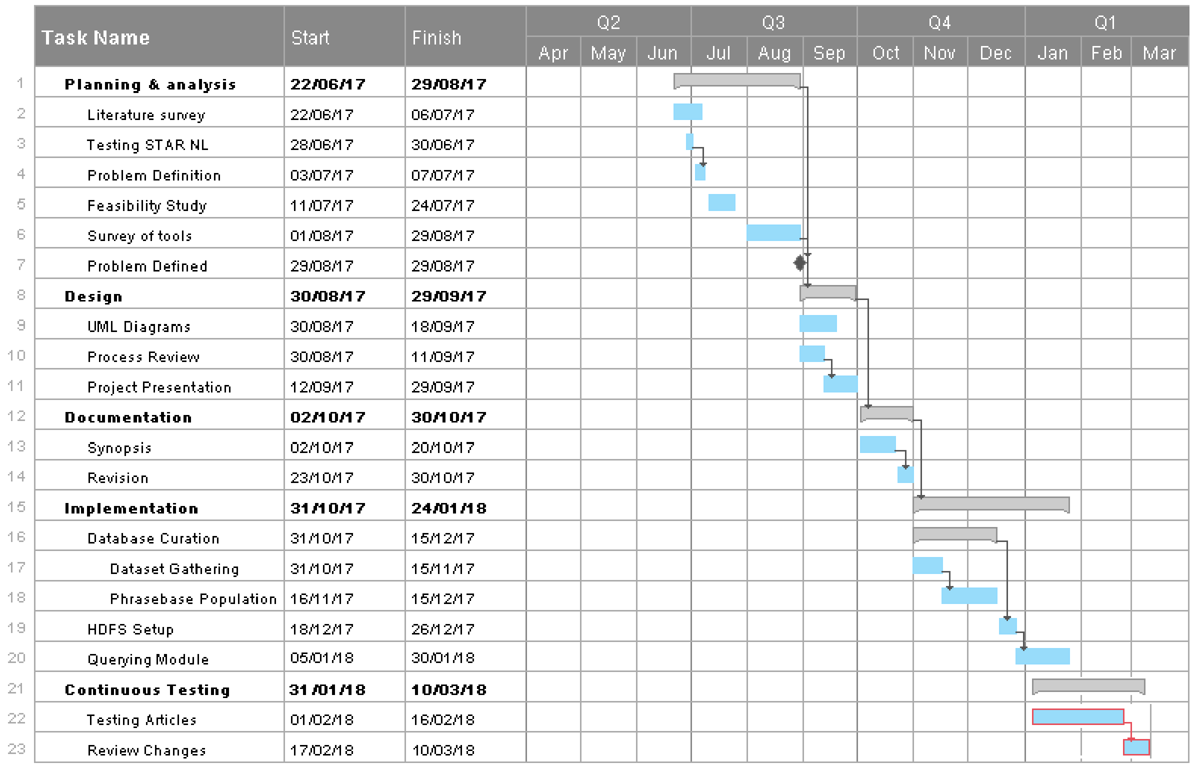
= **107.95**

Number of people working on the project = 3

Productivity = 107.95/3

= **35.98**

3.3. Project Timeline



	Task Name	Start	Finish
1	**Planning & analysis**	**22/06/17**	**29/08/17**
2	Literature survey	22/06/17	06/07/17
3	Testing STAR NL	28/06/17	30/06/17
4	Problem Definition	03/07/17	07/07/17
5	Feasibility Study	11/07/17	24/07/17
6	Survey of tools	01/08/17	29/08/17
7	Problem Defined	29/08/17	29/08/17
8	**Design**	**30/08/17**	**29/09/17**
9	UML Diagrams	30/08/17	18/09/17
10	Process Review	30/08/17	11/09/17
11	Project Presentation	12/09/17	29/09/17
12	**Documentation**	**02/10/17**	**30/10/17**
13	Synopsis	02/10/17	20/10/17
14	Revision	23/10/17	30/10/17
15	**Implementation**	**31/10/17**	**24/01/18**
16	Database Curation	31/10/17	15/12/17
17	Dataset Gathering	31/10/17	15/11/17
18	Phrasebase Population	16/11/17	15/12/17
19	HDFS Setup	18/12/17	26/12/17
20	Querying Module	05/01/18	30/01/18
21	**Continuous Testing**	**31/01/18**	**10/03/18**
22	Testing Articles	01/02/18	16/02/18
23	Review Changes	17/02/18	10/03/18

Fig 2: Project Timeline

Chapter 4

System Diagrams of Phrase Sense Disambiguation

4.1. Sequence Diagram

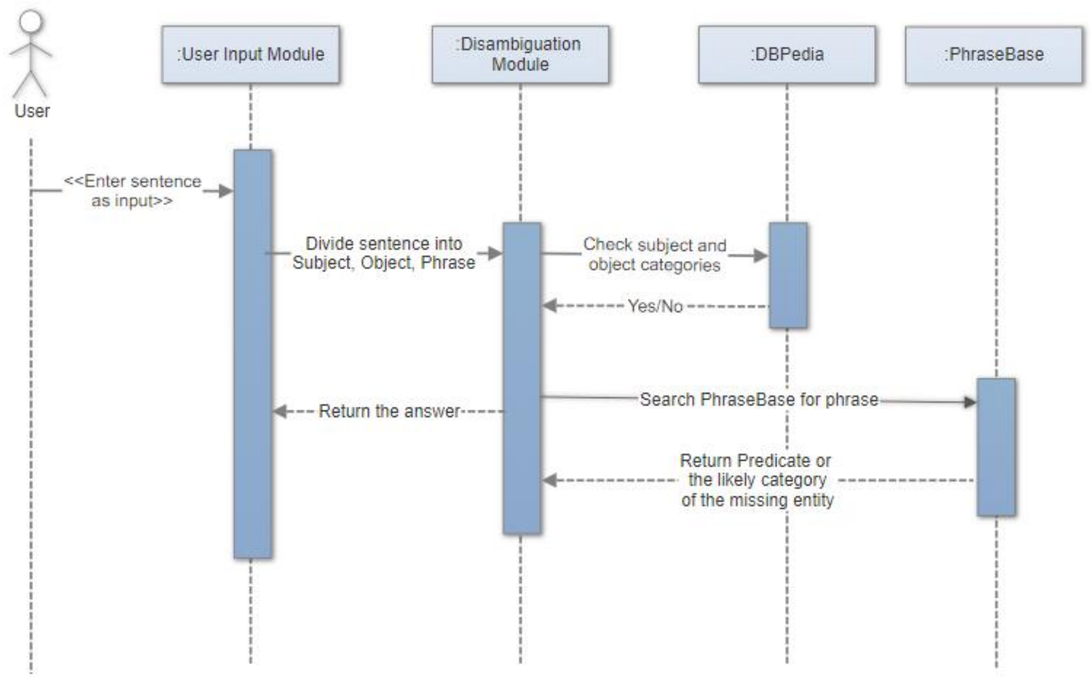
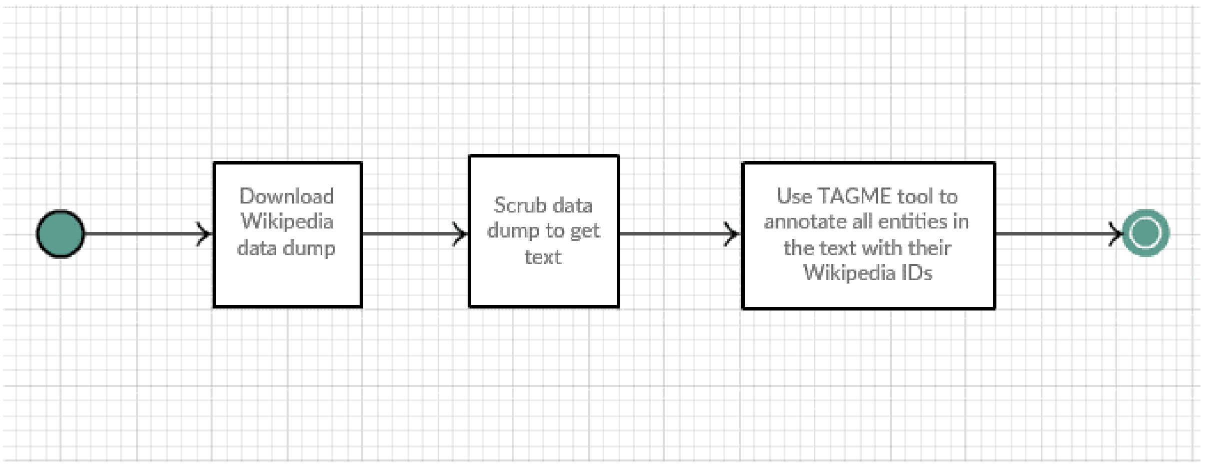

:User Input Module
:Disambiguation Module
:DBPedia
:PhraseBase
User
<<Enter sentence as input>>
Divide sentence into Subject, Object, Phrase
Check subject and object categories
Yes/No
Search PhraseBase for phrase
Return the answer
Return Predicate or the likely category of the missing entity

Fig 3: Sequence Diagram

4.2. Activity Diagrams

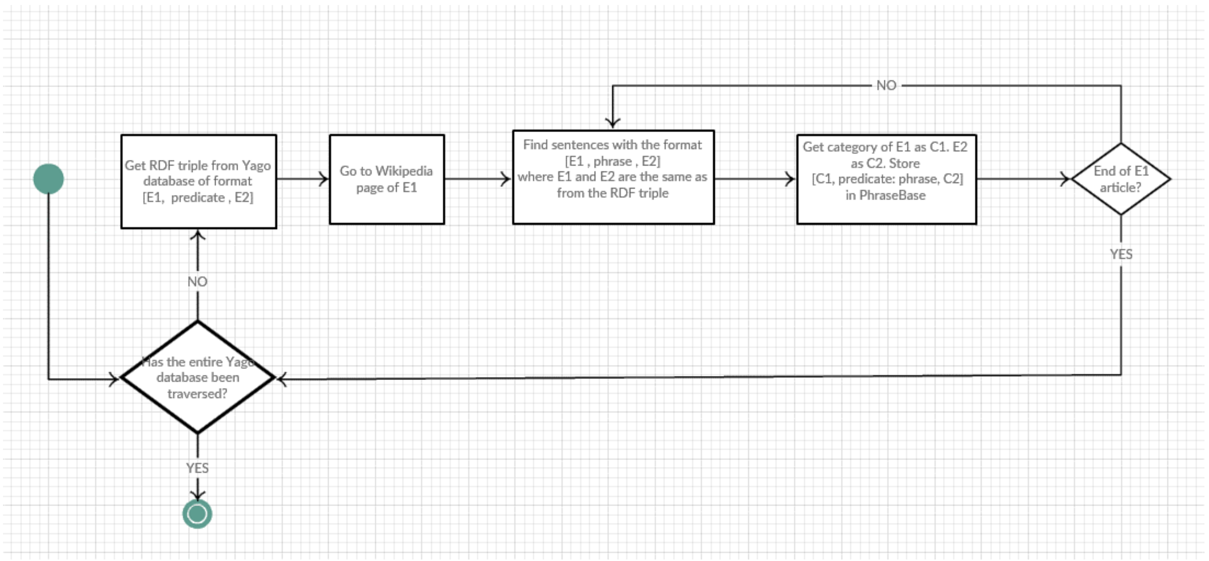
1. Enrich Text:



Download Wikipedia data dump
Scrub data dump to get text
Use TAGME tool to annotate all entities in the text with their Wikipedia IDs

Fig 4: Activity Diagram 1

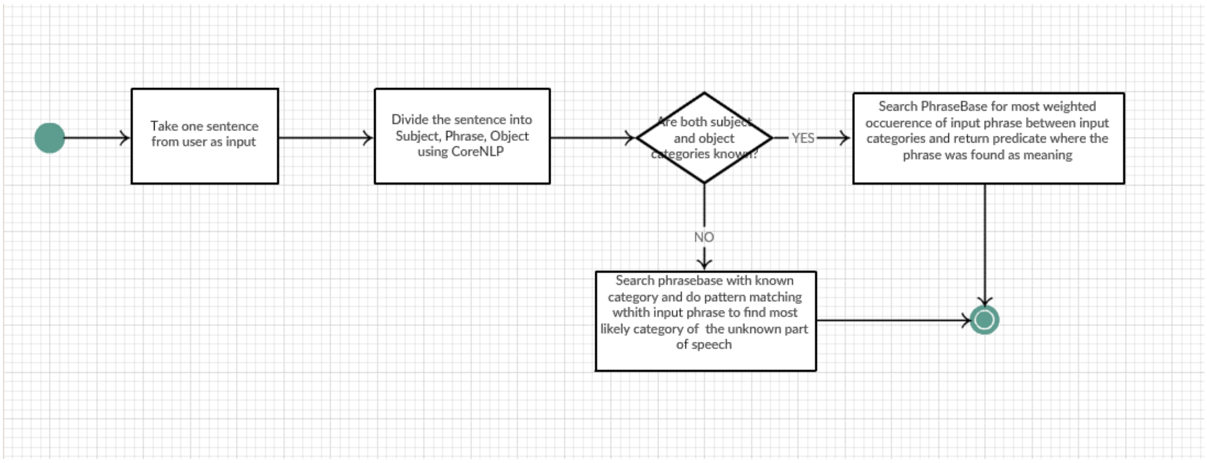
2. Phrasebase:



NO
Get RDF triple from Yago database of format [E1, predicate , E2]
Go to Wikipedia page of E1
Find sentences with the format [E1 , phrase , E2] where E1 and E2 are the same as from the RDF triple
Get category of E1 as C1. E2 as C2. Store [C1, predicate: phrase, C2] in PhraseBase
End of E1 article?
YES
NO
Has the entire Yago database been traversed?
YES

Fig 5: Activity Diagram 2

3. Input:



Take one sentence from user as input
Divide the sentence into Subject, Phrase, Object using CoreNLP
Are both subject and object categories known?
YES
Search PhraseBase for most weighted occuerence of input phrase between input categories and return predicate where the phrase was found as meaning
NO
Search phrasebase with known category and do pattern matching wthith input phrase to find most likely category of the unknown part of speech

Fig 6: Activity Diagram 3

Chapter 5

Modules in Phrase Sense Disambiguation

In this chapter we describe in detail the entire working of our project. Our project has three main modules i.e. data curation, Phrasebase building, and ambiguity resolution. This is a linear process and the output of each stage acts as an input to the next one. In the results sub section we have also showcased the prowess of our project by showing how our method is successful in disambiguating a sentence that two accepted solutions fail on.

5.1. Data Curation

We decided to work on Wikipedia datasets, as sentences and data in Wikipedia are worked on the most by other NLP projects. This helped us in focusing on achieving our aim for Phrase Sense Disambiguation and not worry about other NLP problems such as Anaphora Resolution and Word Sense Disambiguation. We had to setup a Hadoop cluster for this process as the datasets were very large, some exceeding 200 Gigabytes in size. [27] Then we mapped Wikipedia text with the subject and object instances, using datasets provided by DBpedia. Finally, we extracted the sentence we require to process from the complete article, so that we don’t process any unnecessary sentences.

5.1.1. Setting up Hadoop Cluster

The datasets we procured from DBpedia were vast in size. Processing files as large as 200 GBs would require a lot of processing power which one machine could not provide. [20] [21] Hence, we decided to set up a Hadoop Cluster comprising of two 8 GB ram computers and one 4 GB ram computer.

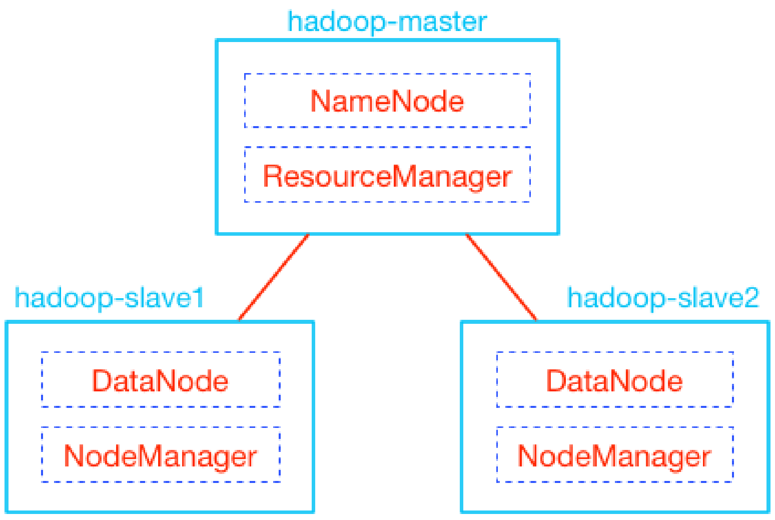


Fig 7: Hadoop Cluster

We majorly used two of the four main Hadoop components, i.e. HADOOP HDFS and HADOOP MAPREDUCE. [19]

5.1.1.1. HADOOP HDFS:

It is a distributed file system that stores files across a cluster with a user specified replication factor. This filesystem was very useful to us because of the need to store and process large datasets.

5.1.1.2. HADOOP MAPREDUCE:

It is a globally defined programming model built especially for parallel processing of large datasets.

Each MapReduce job in our project had two parts, mapper and reducer.

a) Mapper: The mapper code would fire queries on the datasets which was distributed across the cluster. This enabled the parallel firing of multiple queries across each machine in the cluster, giving better processing speeds.

b) Reducer: The reducer code would retrieve query results from each mapper and merge them to form a result.

Most of the pattern matching was done on the cluster setup here, as it provides faster speeds than a single machine, as well as helps in processing of the datasets mentioned further down without the need of any major preprocessing. This became the major reason for us not opting for a database and for the Hadoop cluster.

We also run our final pattern matching on a pseudo-distributed Hadoop cluster to give results as fast as possible by a single machine.

5.1.2. Mapping of DBpedia Datasets

DBpedia has many datasets, each file having various forms of metadata. Since our project needed information from multiple datasets, we took DBpedia provided datasets as input and processed and merged them according to our need.

5.1.2.1. Article Categories

This dataset contains tuples where each tuple has the following fields [23]

- Resource Name: Title of the Article in DBpedia/Wikipedia
- Resource Category: The category of the resource.

Example:

```
<dbr:A> <subject> <dbr:Category:Vowel_letters> <relative-
line=15&absolute-line=187> .
<dbr:Achilles> <subject> <dbr:Category:Characters_in_the_Iliad>
<relative-line=13&absolute-line=271> .
```

5.1.2.2. Mapping based objects

This dataset contains tuples where each tuple has the following fields [24]

- Subject name: Title of the Subject's Wikipedia article/DBpedia resource name
- Predicate: The predicate that defines the relation between the subject and the
object

Example:

```
<dbr:Anarchism> <predicate> <dbr:Education> <wiki:Anarchism?
relative-line=2&absolute-line=310> .
<dbr:Alabama> <dbo:country> <dbr:United_States> <wiki:Alabama?
absolute-line=5> .
```

5.1.2.3. NIF context

This dataset contains tuples where each tuple has the following fields [25]

- Resource name: Title of the Article in DBpedia/Wikipedia
- Resource context: Wikipedia article of the resource.

Example:

```
<dbr:Animalia_(book)> <type> <Context> <wiki:Animalia_(book)> .
<dbr:Animalia_(book)> <beginIndex> "0" <nonNegativeInteger>
<wiki:Animalia_(book)> .
<dbr:Animalia_(book)?dbpv=2016-10&nif=context> <endIndex>
"2294"^^<nonNegativeInteger> <wiki:Animalia_(book)> .
```

```
<dbr:Animalia_(book)> <sourceUrl> <wiki:Animalia_(book)>
<wiki:Animalia_(book)> .
<dbr:Animalia_(book)> <isString> "Animalia is an illustrated
children's book by Graeme Base. … A Learning Time activity guide for
Animalia created by The Little Big Book Club."
```

5.1.2.4. NIF context:

This dataset contains tuples where each tuple has the following fields [26]

- Subject name: Title of the Subject's Article in DBpedia/Wikipedia
- Character positions: Start and end character positions of the object's relevant occurrence in the Wikipedia article of the subject.

Example:

```
<dbr:Animalia_(book)&char=27,42> <beginIndex> "27"
<nonNegativeInteger> <wiki:Animalia_(book)> .
<dbr:Animalia_(book)&char=27,42> <endIndex> "42"
<nonNegativeInteger> <wiki:Animalia_(book)> .
```

Using these datasets, we made a dataset of our own by merging required information from the above datasets.

The merged dataset has tuples where each tuple has the following fields.

- Subject name: Title of the Subject's Article in DBpedia/Wikipedia
- Object name: Title of the Object's Article in DBpedia/Wikipedia
- Predicate: The predicate that defines the relation between the subject and the object
- Resource context: Wikipedia article of the resource.
- Character positions: Start and end character positions of the object's relevant occurrence.

Example:

```
<dbr:!!!_(album)> <dbr:!!!> <dbo:producer> <!!!> <53> <56>
<dbr:!!!_(album)&char=0,112> <Word> <!!! is the eponymous debut
studio album by rock band !!! …>
```

5.1.3. Sentence Extraction

This dataset is used to extract the sentence which contains the relevant occurrence of the object as referenced in the tuple. The Wikipedia page content in the tuple in the input dataset is then replaced by this extracted sentence. This is done because the rest of the content is not relevant to the tuple, and the extracted sentence is the only information needed to proceed. Doing this reduces the file size and leads to faster processing in further stages.

5.2. Phrasebase Generation

The Data curation module gives us data in the format that we need. This sub section focuses on using that data to build a Phrasebase. We extract the phrase using CoreNLP, then work on trimming down the dataset due to our limitations and focusing only on simple sentences, then the ontologies of DBpedia were mapped and merged to form our Phrasebase.

5.2.1. Extraction of phrase using CoreNLP

The output of the previous stage gives us a sentence that contains the subject and object according to RDF triple being processed. Hence, the phrase used in this extracted sentence maps directly to the predicate mentioned in the RDF triple.
This phrase was extracted using Stanford CoreNLP parser. [18] CoreNLP is a deep natural language processor that takes a sentence as input and can give carious forms of metadata about the sentence as output. We have used CoreNLP to do tag the parts of speech in the input sentence. The POS tags returned by CoreNLP are then used to extract the Verb Phrase (VP) which is the phrase in the sentence.

Sample CoreNLP output for
"Lionel Messi played for Argentina"

POS tag:
```
Lionel/NNP Messi/NNP played/VBN for/IN Argentina/NNP
```

Parse Tree:
```
(ROOT
  (SINV
    (NP (NNP Lionel) (NNP Messi))
    (VP (VBN played)
      (PP (IN for)
        (NP (NNP Argentina))))))
```

The parse tree is used to look at the POS tag of each node in the tree and extract the sub tree that has a VP as root. Hence, the extracted phrase from the above parse tree is "played for".

5.2.2. Simple Sentences

CoreNLP gives a perfect extraction of the phrase if the sentence is a simple sentence. However, if the sentence is not simple, i.e. cannot be broken down into the subject-predicate-object format, the sentence must be dropped because the phrase extracted using CoreNLP has Noun Phrases (NP) in it. Due to our inability to process such sentences, these tuples are dropped. This is a limitation in the project and can be improved in the future.

5.2.3. Subject Object Ontologies

The output of the previous stage gives us the subject-predicate-format, with each predicate with the phrase extracted from the sentence. The Phrasebase, however does not require the actual resource names of the subject and the object but rather their categories. We have used DBpedia ontologies as the categories for each entity. In this step, we extract the DBpedia (dbo) ontology of each resource in the file from a DBpedia dataset. [28] The subject and object in the file is then replaced with its ontology.

5.2.4. Merge Phrases with Ontologies

In the previous stage, each tuple from the original dataset is converted into a (subject_ontology - predicate - object_ontology) tuple. However, multiple entities from the original had the same ontologies. Therefore, all tuples having the same (subject_ontology - predicate - object_ontology) pattern are combined. The predicate for all these combined tuples were the same, but the phrases might not have been. Therefore, the combined tuple will have all the occurring phrases listed as a linked list to the predicate in the pattern.

e.g. Consider the tuples

1) Soccer player represents soccer team

2) Soccer player was a part of a soccer team

Here, the subject is <dbo:soccerPlayer>, object is <dbo:soccerTeam>, and the predicate is "plays for"

Hence, the merged output will look like

<dbo:soccerPlayer> - plays for (represents, is a part of, ...) - <dbo:soccerTeam>

Sample:

```
<dbr:'Azza> <dbo:Settlement> "17901024"
<dbr:'Blue_Blazes'_Rawden> <dbo:Film> "44964056"
<dbr:'Bout_Changes_'n'_Things>  <dbo:Album> "24895480"
```

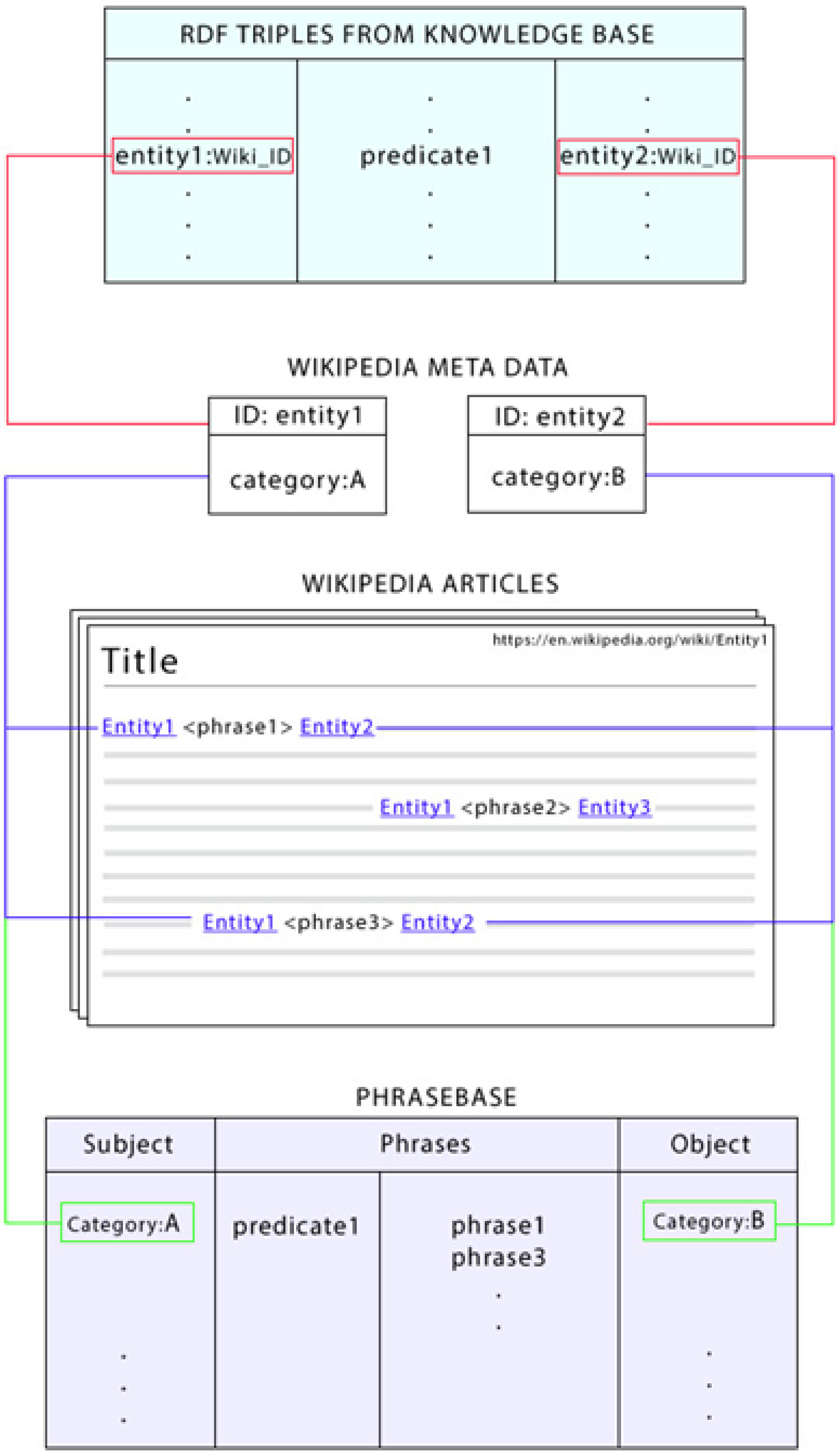


Fig 8: Phrasebase Generation

Every tuple of the Phrasebase thus generated has the following fields:

```
<phrase> <relation> <dbo:sub> <dbo:obj> <count c> <rank r>
```

Where:

<phrase> - The phrase of the pattern

<relation> - The relation that is signified by the phrase

<dbo:sub> - Ontology of the subject

<dbo:obj> - Ontology of the object

<count c> - Count of the pattern

<rank r> - Rank of the pattern

Example: Phrase “born in”

```
born in dbo:birthPlace dbo:Swimmer owl#Thing count 299 rank 9

born in dbo:birthPlace dbo:Swimmer owl#Thing count 6 rank 1

born in dbo:birthPlace dbo:Scientist dbo:Country count 475 rank 1

born in dbo:birthPlace dbo:Scientist dbo:City count 1449 rank 1

born in dbo:birthPlace dbo:Royalty dbo:Town count 143 rank 1

born in dbo:birthPlace dbo:Royalty dbo:Settlement count 500 rank 1

born in dbo:birthPlace dbo:PrimeMinister owl#Thing count 335 rank 9

born in dbo:birthPlace dbo:PrimeMinister owl#Thing count 9 rank 1

born in dbo:birthPlace dbo:PrimeMinister dbo:WorldHeritageSite count
1 rank 1

born in dbo:birthPlace dbo:PrimeMinister dbo:Village count 5 rank 1
```

5.3. Ambiguity Resolution

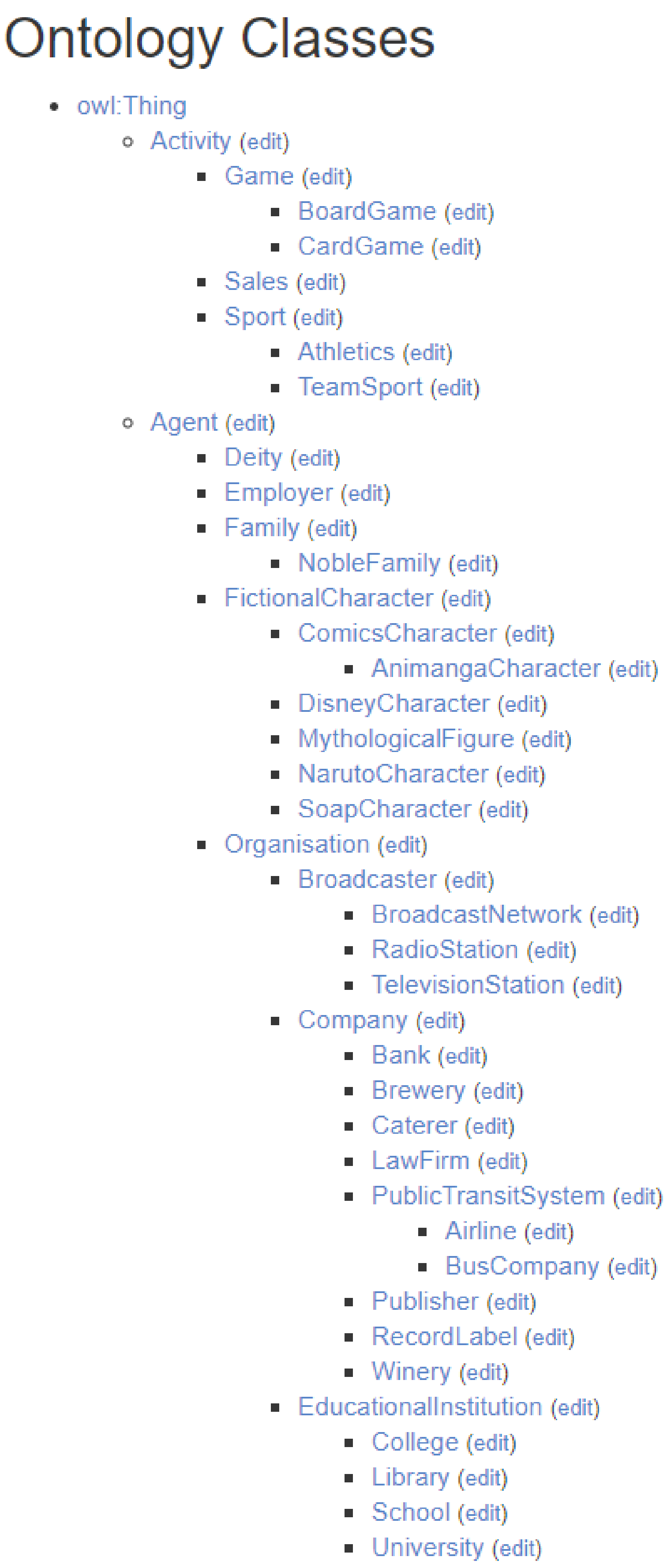
In this subsection the tuples in the phrases are enriched by adding metadata like rank and count. The process to take user input and disambiguate it is described.

5.3.1. Generating Rank and Count

The output of the previous stage had aggregated tuples which were combined using recurring patterns. The number of tuples that were used to generate an aggregated tuple was recorded. This number is the count for that aggregated tuple. This count is an indicator of the popularity and frequency of the pattern. These counts are significant during the disambiguation of new input. If in the input either the subject or object ontology is unknown, pattern matching with the Phrasebase is not enough for disambiguation as pattern matching may have multiple results, but the pattern with the higher count is the most likely answer. Patterns that were directly generated by the process so far were a part of the original dataset, and hence are directly found in the source of the data. Such directly found patterns are given a rank of one. In this step, if multiple patterns have a subject or object that have a common ancestor in the ontology tree, a new pattern is generated by replacing the subject/object in those patterns by their common ancestor. [28]

e.g. Suppose two patterns "Soccer player was born in location" and "Cricket player was born in location" are found in the dataset. These patterns will have a rank of 1. Since "soccer player" and "cricket player" both have a common ancestor "Athlete", a new pattern "Athlete was born in location" is generated. This pattern will have a rank of 2.

Similarly, any pattern with rank x can be combined with a pattern with rank y if they have a common ancestor, and the new pattern will have a rank of x + y.



Ontology Classes

- owl:Thing
  - Activity (edit)
    - Game (edit)
      - BoardGame (edit)
      - CardGame (edit)
    - Sales (edit)
    - Sport (edit)
      - Athletics (edit)
      - TeamSport (edit)
  - Agent (edit)
    - Deity (edit)
    - Employer (edit)
    - Family (edit)
      - NobleFamily (edit)
    - FictionalCharacter (edit)
      - ComicsCharacter (edit)
        - AnimangaCharacter (edit)
      - DisneyCharacter (edit)
      - MythologicalFigure (edit)
      - NarutoCharacter (edit)
      - SoapCharacter (edit)
    - Organisation (edit)
      - Broadcaster (edit)
        - BroadcastNetwork (edit)
        - RadioStation (edit)
        - TelevisionStation (edit)
      - Company (edit)
        - Bank (edit)
        - Brewery (edit)
        - Caterer (edit)
        - LawFirm (edit)
        - PublicTransitSystem (edit)
          - Airline (edit)
          - BusCompany (edit)
        - Publisher (edit)
        - RecordLabel (edit)
        - Winery (edit)
      - EducationalInstitution (edit)
        - College (edit)
        - Library (edit)
        - School (edit)
        - University (edit)

Fig 9: DBpedia Ontology

5.3.2. Tagme

TAGME is an on the fly annotator that tags entities in input sentences with their Wikipedia pages. [16]

When a new sentence is given to the system as input, it needs to be tagged first. Hence, we use the TAGME API to get the tagged result of the sentence. These tags are necessary since they help us map the entities in the sentence to their Wikipedia pages, which can then be used to find the ontology for the tagged entity.

Samples of TAGME output for sentence "Lionel Messi played for Argentina":

a) Web interface

b) JSON file returned by the TAGME API which is used in our code

```
{"timestamp":"2018-04-20T04:32:28",
"time":1,
"test":"5",
"api":"tag",
"annotations":[{
"id":2150841,
"title":"Lionel Messi",
"start":0,
"link_probability":1,
"rho":0.8658769130706787,
"end":12,
"spot":"lionel messi"},
{"id":454699,
"title":"Argentina national football team",
"start":24,
"link_probability":0.6492162942886353,
"rho":0.6904850006103516,
"end":33,
"spot":"argentina"}],
"lang":"en"}"
```

`"id"`: Wikipedia ID

`"title"`: title of the Wikipedia page

`"start"`: character number of the start of the tag

`"end"`: character number of the end of the tag

`"spot"`: string in the input sentence that was tagged

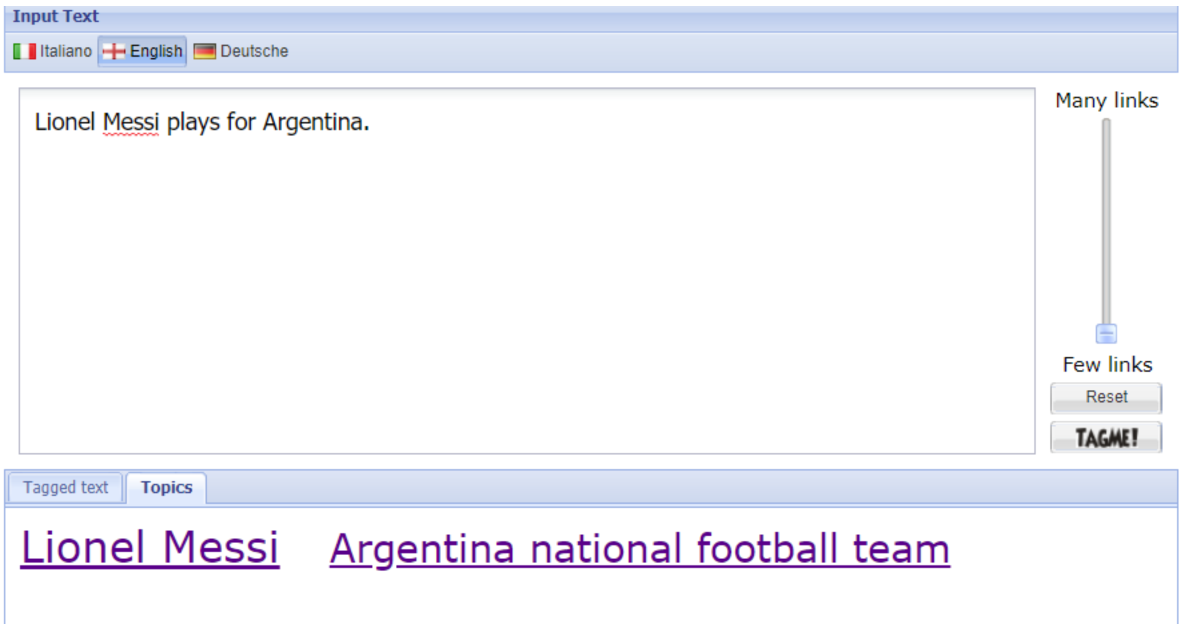
`"link_probability"`: Probability of the accuracy of the tag

5.3.3. Existing System Outputs

The current method for entity disambiguation does not consider the relation between the subject and object in the sentence, that is the context of the phrase. These current systems use a probabilistic model to find the relation between the subject and the object, ignoring the phrase in between and its significance. The probabilistic model uses the count of the occurrences of the subject and object together and categorizes the ambiguous entities based on this count. Hence different systems using different probabilistic models in the backend give different results, which aren’t always accurate.

5.3.3.1. Tagme

If given an input sentence:
“Lionel Messi plays for Argentina.”
Tagme tags Lionel Messi as the soccer player accurately, as well as tags Argentina as the soccer team and not as the country.
But when given an input:
“Lionel Messi was born in Argentina.”
Tagme tags Messi accurately but still tags Argentina as the soccer team and not the physical country.

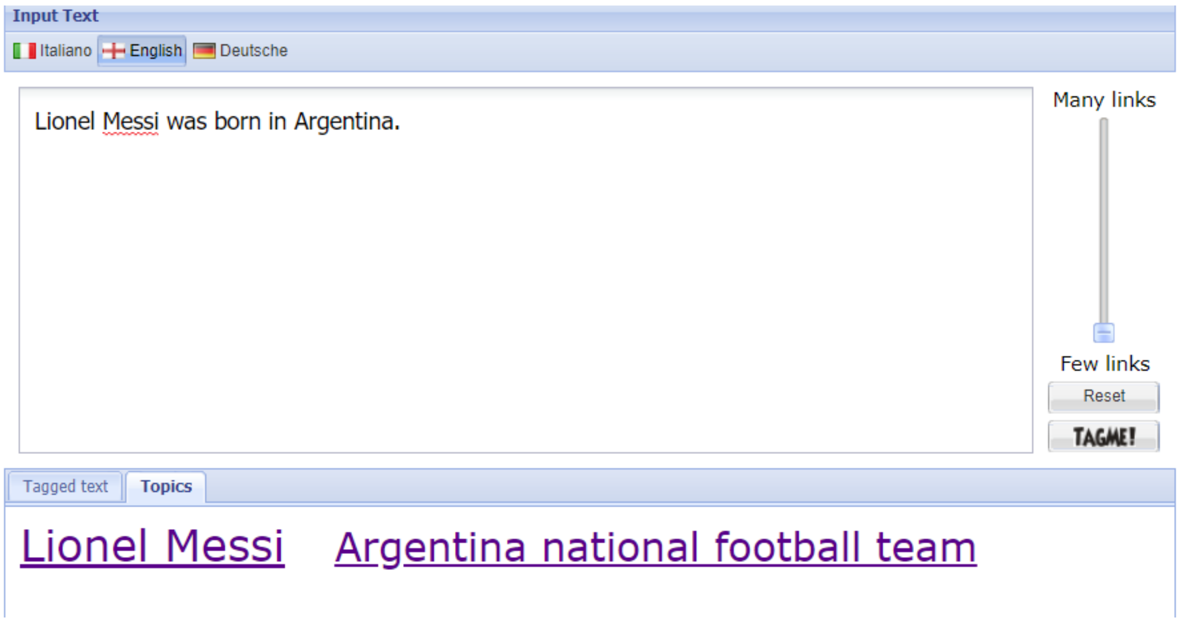

Input Text

Lionel Messi plays for Argentina.

Tagged text | Topics

Lionel Messi   Argentina national football team

Fig 10: Tagme output for sentence 1


Input Text

Lionel Messi was born in Argentina.

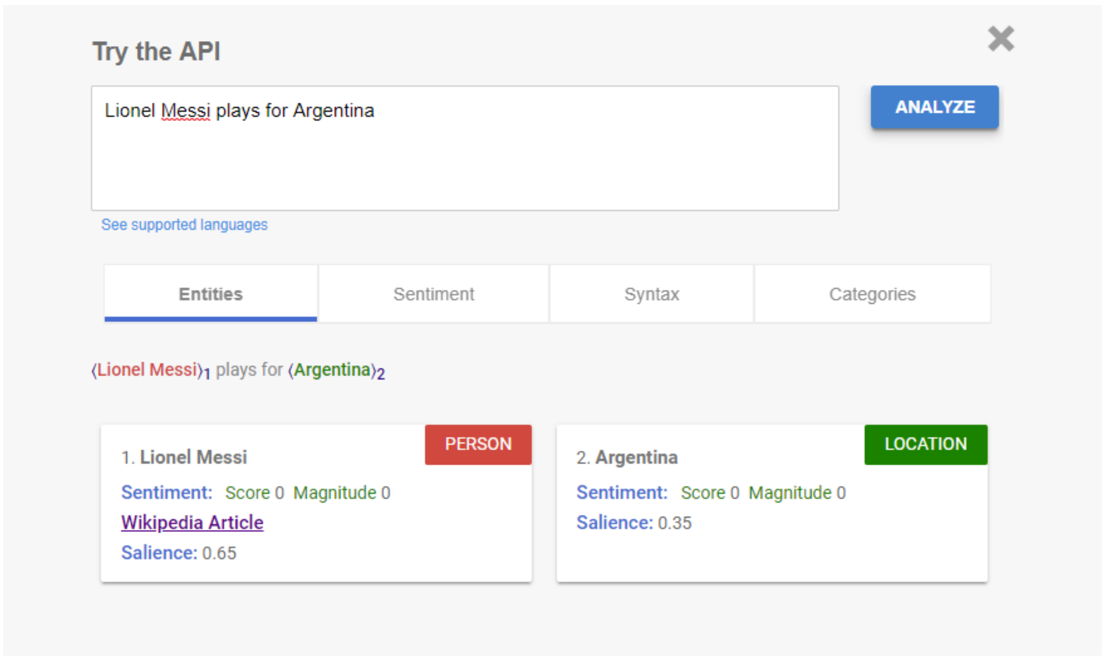
Tagged text | Topics

Lionel Messi   Argentina national football team

Fig 11: Tagme output for sentence 2

5.3.3.2. Google Cloud Natural Language

Google Cloud Natural Language is also a powerful tagger that helps to gain insights from unstructured text. We focused on the entity tagging capability of this tool. When given the same input sentence of "Lionel Messi plays for Argentina.", Google tagged Lionel Messi as the soccer player accurately, but tagged Argentina as the country and not the team. When given an input of "Lionel Messi was born in Argentina." Google tags Messi accurately and accurately tags Argentina as the physical country.


Try the API

Lionel Messi plays for Argentina

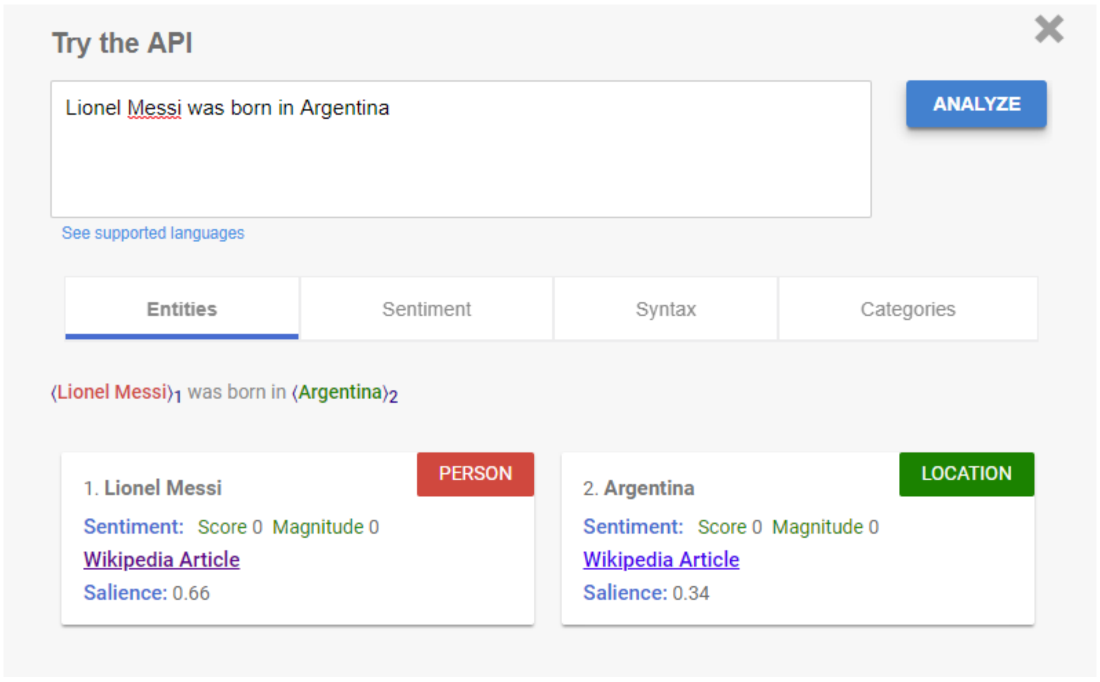
See supported languages

⟨Lionel Messi⟩$_1$ plays for ⟨Argentina⟩$_2$

1. Lionel Messi PERSON
Sentiment: Score 0 Magnitude 0
Wikipedia Article
Salience: 0.65

2. Argentina LOCATION
Sentiment: Score 0 Magnitude 0
Salience: 0.35

Fig 12: Google output for sentence 1


Try the API

Lionel Messi was born in Argentina

See supported languages

⟨Lionel Messi⟩$_1$ was born in ⟨Argentina⟩$_2$

1. Lionel Messi PERSON
Sentiment: Score 0 Magnitude 0
Wikipedia Article
Salience: 0.66

2. Argentina LOCATION
Sentiment: Score 0 Magnitude 0
Wikipedia Article
Salience: 0.34

Fig 13: Google output for sentence 2

We thus observe that the phrase between the subject and object was inconsequential to these taggers, and they provide results as per the output provided by their probabilistic model. Our project aims to solve this by disambiguating the meaning of the phrase in between and thus tagging the entities accurately.

5.3.4. Disambiguation and results

The input sentence is tagged using Tagme as stated previously. But here, we also factor in the probability value provided by Tagme, which indicates its confidence of how accurately it has tagged the sentence. We define a threshold for this probability and drop any tags that are below this threshold. This is done to avoid wrong tags and resolve the issues as demonstrated in Section 5.3.3.

Code:

```
     String s1 = new String
("https://tagme.d4science.org/tagme/tag?lang=en&gcube-token=17bf7977-
7ffb-4051-a480-7edfeac6045f-843339462&text=");
        String temp = sentence; // Input Sentence
        String s2 = temp.replace(" ","%20");
        s1=s1.concat(s2);
        URL oracle = new URL(s1);
        URLConnection yc = oracle.openConnection();
        BufferedReader in = new BufferedReader(new
InputStreamReader(yc.getInputStream()));
        String inputLine;
        inputLine = in.readLine();
        Map annot;
        String[] tag_id;
        String[] tag_title;
        String[] tag_prob;
        JSONObject jsonObject = new JSONObject(inputLine);
     JSONArray friends = jsonObject.getJSONArray("annotations");
     tag_id = new String[friends.length()];
     tag_title = new String[friends.length()];
     tag_prob = new String[friends.length()];
     int index;
     for (index=0; index<friends.length(); ++index){
           JSONObject currentFriend = friends.getJSONObject(index);
           tag_id[index] = currentFriend.getString("id");
           tag_title[index] = currentFriend.getString("spot");
           tag_prob[index] = currentFriend.getString("link_probability");
           System.out.println(tag_id[index]+" "+tag_title[index]+"
"+tag_prob[index]);
        }
```

The phrase in the input sentence is extracted using CoreNLP.

Code:

```
     // create an empty Annotation just with the given text
     Annotation document = new Annotation(sentence);
     // run all Annotators on this text
     pipeline.annotate(document);
     // these are all the sentences in this document
     // a CoreMap is essentially a Map that uses class objects as keys and
has values with custom types
     List<CoreMap> sentences =
document.get(CoreAnnotations.SentencesAnnotation.class);
     for(CoreMap sent: sentences) {
       Tree sentenceTree =
sent.get(TreeCoreAnnotations.TreeAnnotation.class);

    TregexPattern nounPhraseTregexPattern = TregexPattern.compile("VP");
    TregexMatcher nounPhraseTregexMatcher =
nounPhraseTregexPattern.matcher(sentenceTree);
    Double sub_prob=0.0;
    Double ob_prob=0.0;
    while (nounPhraseTregexMatcher.find()) {
    Tree tree = nounPhraseTregexMatcher.getMatch();
    VP = new String(tree.toString());

     TregexPattern nounPhraseTregexPattern1 = TregexPattern.compile("NP");
     TregexMatcher nounPhraseTregexMatcher1 =
nounPhraseTregexPattern1.matcher(tree);
        while (nounPhraseTregexMatcher1.find()) {
        NP = nounPhraseTregexMatcher1.getMatch().toString();
        //System.out.println(NP);
        }
    VP = VP.replace(NP,"");
    VP = VP.substring(4,VP.length()-1);
```

```
System.out.println(VP);
if(VP.isEmpty())
{
 continue;
}
else{
word="";
int k=0;
while(k < VP.length())
{
 if(VP.charAt(k) == ' ')
 {
 k++;
 int flag =0;
 if(!(k >= VP.length()))
       while(VP.charAt(k) != ')' && VP.charAt(k) != '(' )
       {
             flag = 1;
             word += VP.charAt(k++);
       }
       if(flag == 1)
             word += " ";
 }
 k++;
}
```

This phrase is matched with the existing phrases in our Phrasebase and the matched patterns are displayed in our results. If Tagme has tagged any one or two of the subject and object accurately as per our defined thresholds, we again match the ontologies of these subjects and objects with the matched phrases.

Code:

```
StringTokenizer itr = new StringTokenizer(sentence,word);
subject = sentence.substring(0,sentence.indexOf(word.trim())).trim();
object = (sentence.replace(subject,"")).replace(word,"").trim();
System.out.println(subject+"+++"+object);
sub_tag="";
ob_tag="";
sub_res="";
ob_res="";
double threshold = 0.75;
for(int z=0;z<index;z++){
     if((subject).contains(tag_title[z].trim())&&
Double.parseDouble(tag_prob[z]) > threshold)
           {
                 sub_tag = tag_id[z];
                 sub_prob = Double.parseDouble(tag_prob[z]);
     }
     if((object).contains(tag_title[z].trim())&&
Double.parseDouble(tag_prob[z]) > threshold)
           {
                 ob_tag = tag_id[z];
                 ob_prob = Double.parseDouble(tag_prob[z]);
     }
}
```

This gives us a list of patterns with the same subject, object or both. Out of these patterns, the pattern with the highest count and least rank, as per a defined threshold are displayed in our result.

Chapter 6

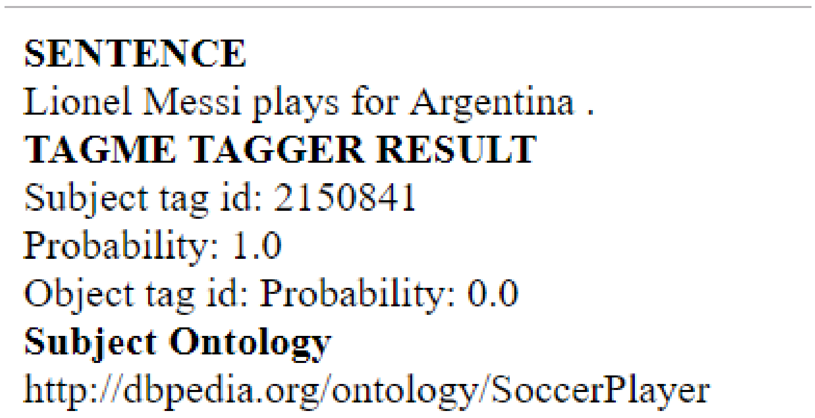
Results

Using the system thus developed, we were able to extract many patterns and phrases. We successfully extracted approximately 390,000 patterns and 69,000 phrases. The number of phrases can be easily increased by incorporating a system that can extract phrases from complex sentences. This substantial number of patterns would have been impossible to match with the input pattern in a fixed time duration on a single computer, hence we deployed a MapReduce function on a pseudo-distributed Hadoop cluster.

The input file consisted of multiple sentences, and for each of these sentences, the mapper and the reducer had to be initiated. To decrease the time taken, we initiated the CoreNLP Stanford Pipeline only once per input file. This pipeline was loaded in approximately 35 seconds. After this, the MapReduce function ran for approximately 1 minute per sentence in the input file.

This included all the steps including tagging from Tagme, then CoreNLP extracting the subject, object and the phrase of the sentence, then the extracted subject and object was matched with the Tagme output, and if the tagged result passed the probability threshold, it was stored else it was discarded. Then all the patterns that matched only the phrase were populated and stored in the temporary output file. If the subject and object categories previously found were the same as present in a pattern, then the pattern was written into the output file. However, if the subject and object category were not matched, the ontology hierarchy of the subject/object was matched, and the result was stored in the output file. This time of 1 minute can be greatly reduced if run on a distributed Hadoop cluster with more than 2 data nodes and a dedicated name node server machine.

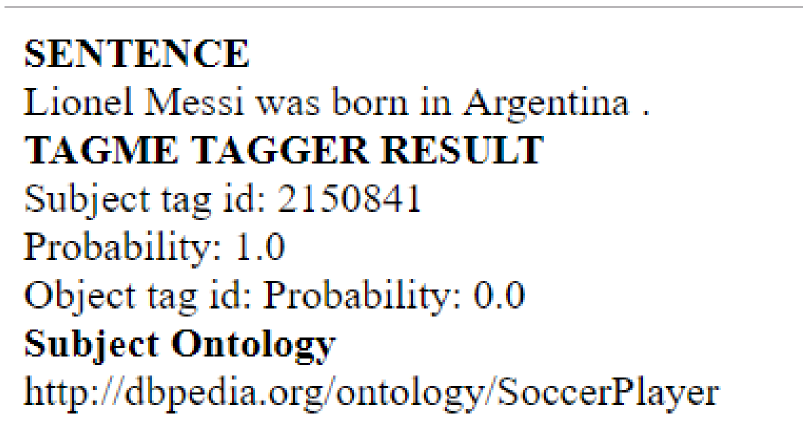
Fig 14 shows the Tagme result and the probability threshold for the sentence “Lionel Messi plays for Argentina.”



```
SENTENCE
Lionel Messi plays for Argentina .
TAGME TAGGER RESULT
Subject tag id: 2150841
Probability: 1.0
Object tag id: Probability: 0.0
Subject Ontology
http://dbpedia.org/ontology/SoccerPlayer
```

Fig 14: Sentence 1 Tagme Result

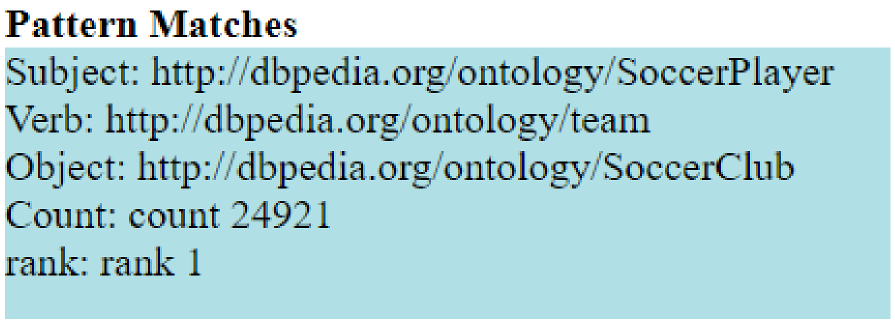
Fig 15 shows the same for “Lionel Messi was born in Argentina.”



```
SENTENCE
Lionel Messi was born in Argentina .
TAGME TAGGER RESULT
Subject tag id: 2150841
Probability: 1.0
Object tag id: Probability: 0.0
Subject Ontology
http://dbpedia.org/ontology/SoccerPlayer
```

Fig 15: Sentence 2 Tagme Result

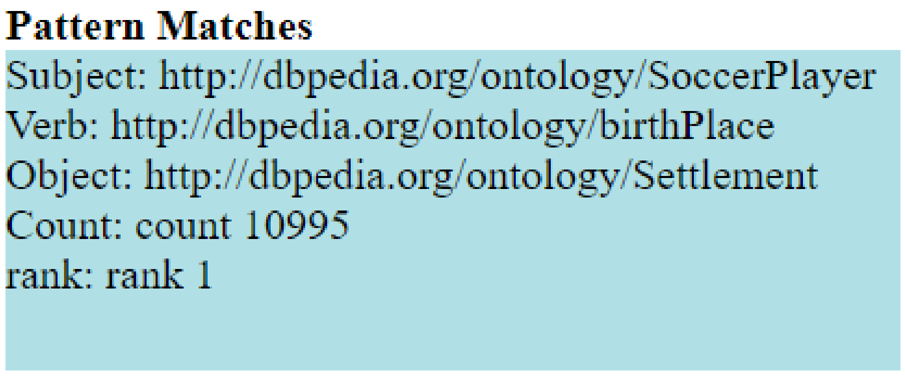
Fig 16 shows the result for the first sentence, and as seen, it has accurately tagged Argentina as Soccer Club, and since the rank is 1, the result we give is highly unambiguous.



```
Pattern Matches
Subject: http://dbpedia.org/ontology/SoccerPlayer
Verb: http://dbpedia.org/ontology/team
Object: http://dbpedia.org/ontology/SoccerClub
Count: count 24921
rank: rank 1
```

Fig 16: Sentence 1 Final Result

Fig 17 shows the result for the second sentence, and similarly, our project successfully shows the output where Argentina isn't tagged as the Soccer Club but as a settlement, which is the default ontology of any physical location in DBpedia.

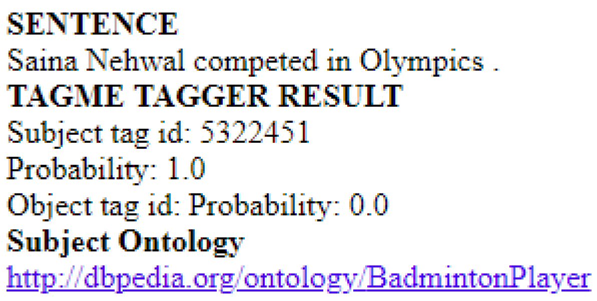


```
Pattern Matches
Subject: http://dbpedia.org/ontology/SoccerPlayer
Verb: http://dbpedia.org/ontology/birthPlace
Object: http://dbpedia.org/ontology/Settlement
Count: count 10995
rank: rank 1
```

Fig 17: Sentence 2 Final Result

However, if no patterns are matched in the pattern matching step, we attach the entire ontology hierarchy to the subject or object ontology which is known. A common ancestor is found for the pattern given in the input. Since the pattern is made during runtime, specific to this result, the count will be 0, and the answer becomes slightly ambiguous, but this result is not wrong.

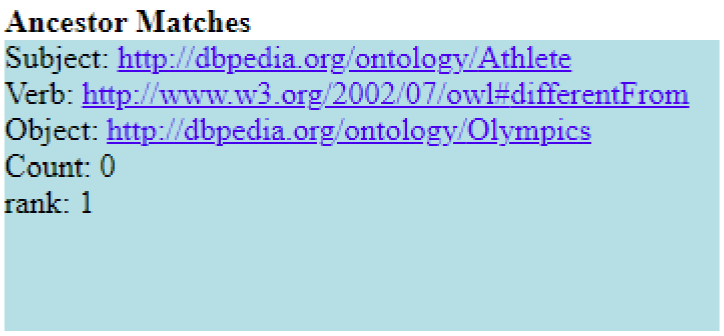
For example, in the sentence "Saina Nehwal competed in Olympics.", Saina Nehwal is accurately tagged as a Badminton Player as shown in Fig 18.



SENTENCE
Saina Nehwal competed in Olympics .
TAGME TAGGER RESULT
Subject tag id: 5322451
Probability: 1.0
Object tag id: Probability: 0.0
Subject Ontology
http://dbpedia.org/ontology/BadmintonPlayer

Fig 18: Sentence 3 Tagme Result

However, there are no patterns in our Phrasebase that match the pattern present here with the subject ontology as Badminton Player. Hence the ancestors are matched, and the result is shown as given in the Fig 19.



Ancestor Matches
Subject: http://dbpedia.org/ontology/Athlete
Verb: http://www.w3.org/2002/07/owl#differentFrom
Object: http://dbpedia.org/ontology/Olympics
Count: 0
rank: 1

Fig 19: Sentence 3 Final Result

Chapter 7

Conclusion

Phrase sense disambiguation proposed a method to disambiguate sentences by recognizing the meaning of the phrases in the sentence, and hence understanding the context. Existing systems use a probabilistic model for entity disambiguation. We have seen the shortcomings of such a model. In this project, we tried entity disambiguation by recognizing the context using phrases, and we can see that it gave better results than the existing systems.

Our aim with PSD was to build a backend system that front-end systems such as search engines and other forms of natural language query processors could use to understand the user's input better, and hence generate better results. Using the example in our results, we notice that the phrase “was born in” is tagged as birthplace. This is highly useful for Information Retrieval systems, as these tagged phrases would not require any extra processing as our system can confidently provide the meaning of the phrase by means of an ontology. One functional drawback of such a system is the in-memory storage of the Phrasebase. The Phrasebase is a huge data structure which cannot be stored in user's device memory always, and due to the vast nature of the data structure, query processing is slow.

Overall, we can see that a model based on phrase disambiguation can be used by NLP systems worldwide to generate better results.

Chapter 8

Future Scope

A new feature can be added, which is best described as "Generation of missing links". Consider an input sentence "Saina Nehwal competed in Delhi".

For this sentence, the pattern "Badminton player competed in location" is not likely to be found, as players generally compete in tournaments/matches and not locations.

The current system will know that the most popular partial match for this input is "player competed in location". A module can be built that then looks for patterns with "location" as the object and "tournament"/"match" as the subject and possibly find a pattern like "tournament held in location".
Thus, the system finds the missing link and recognizes the context as "Badminton player compete in tournament held in location."

The sentence processing using CoreNLP can be improved so that the system can process complex and compound sentences. This will help in generation of a richer Phrasebase which can be used to generate even better results.
The Phrasebase can be optimized by using a better data structure, and a faster way to process queries can be designed to reduce response time.

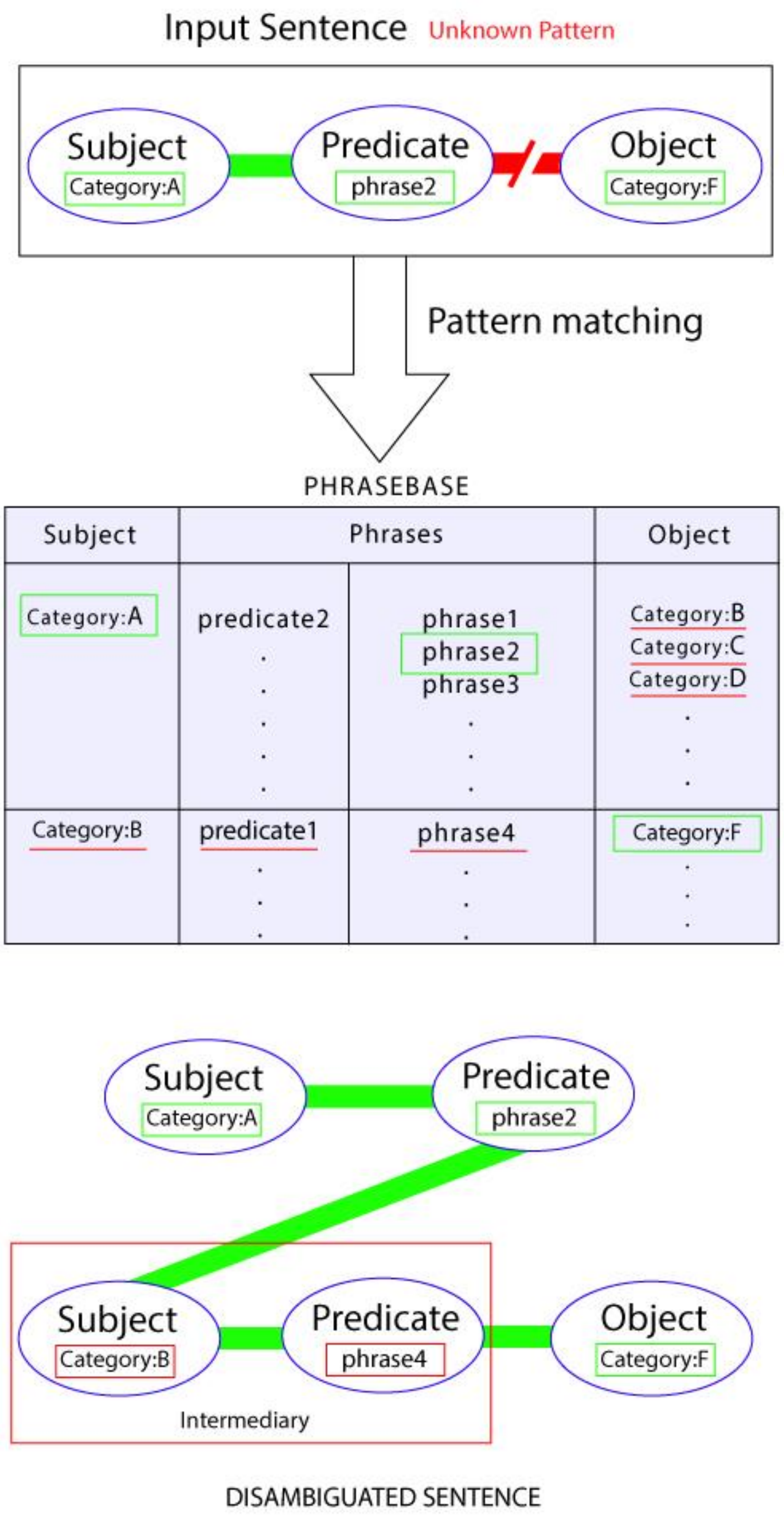


Fig 20: Missing Links Module

References

[1] Rohit Giyanani, Mukti Desai, "Spam Detection using Natural Language Processing", IOSR Journal of Computer Engineering (IOSR-JCE), Volume 16, Issue 5, PP 116-119.

[2] https://nlp.stanford.edu/software/tagger.html

[3] https://nlp.stanford.edu/software/CRF-NER.html

[4] Peter D. Turney, "Thumbs Up or Thumbs Down? Semantic Orientation Applied to Unsupervised Classification of Reviews", Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL), Philadelphia, July 2002, pp. 417-424.

[5] https://stanfordnlp.github.io/CoreNLP/coref.html

[6] Samhith. K, Arun Tilak., Prof G. Panda, "Word Sense Disambiguation using WordNet Lexical Categories", 2016 International Conference on Signal Processing, Communication, Power and Embedded System (SCOPES)

[7] https://nlp.stanford.edu/software/lex-parser.html

[8] Thorsten Brants, "Natural Language Processing in Information Retrieval", Google Inc.

[9] K. Selçuk Candan, Huan Liu, Reshma Suvarna, "Resource description framework: metadata and its applications", ACM SIGKDD Explorations Newsletter Homepage archive, Volume 3 Issue 1, July 2001, Pages 6-19

[10] http://dtd.nlm.nih.gov/publishing/tag-library/n-2a60.html

[11] Fabian M. Suchanek, Gjergji Kasneci, Gerhard Weikum, "Yago: a core of semantic knowledge", WWW '07 Proceedings of the 16th international conference on World Wide Web, Pages 697-706

[12] Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, Zachary Ives, "DBpedia: a nucleus for a web of open data", ISWC'07/ASWC'07 Proceedings of the 6th international the semantic web and 2nd Asian conference on Asian semantic web conference, Pages 722-735

[13] Aldo Gangemi, Valentina Presutti, Diego Reforgiato Recupero, Andrea Giovanni Nuzzolese, Francesco Draicchio, Misael Mongiovì, “Semantic Web Machine Reading with FRED”, Semantic Web Journal 8(6):873-893, 2017

[14] Rutuja Gurav, Nikhil Gore, Sadhana Desai, “S.T.A.R. – NL: Simplification of Text & Anaphora Resolution in Natural Language. A Discourse-based Question Answering System.”, Report submitted in partial fulfilment of the requirements of the degree of B.E. in Computer Engineering, at Computer Department of Vidyalankar Institute of Technology, 2017

[15] Peter Exner and Pierre Nugues, “Entity Extraction: From Unstructured Text to DBpedia RDF Triples”, Department of Computer Science, Lund University, Conference: Web of Linked Entities Workshop in conjunction with the 11th International Semantic Web Conference (ISWC 2012) Boston, USA, November 11, 2012.

[16] Paolo Ferragina, Ugo Scaiella, “TAGME: on-the-fly annotation of short text fragments (by Wikipedia entities)”, CIKM '10 Proceedings of the 19th ACM international conference on Information and knowledge management, pages 1625-1628

[17] B. W. Boehm, “A spiral model of software development and enhancement”, Computer (Volume: 21, Issue: 5, May 1988)

[18] Manning, Christopher D., Surdeanu, Mihai, Bauer, John, Finkel, Jenny, Bethard, Steven J., and McClosky, David, “The Stanford CoreNLP Natural Language Processing Toolkit”, Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations, 2014, pp. 55-60

[19] Tom White, “Hadoop: The Definitive Guide”, O'Reilly Media, Inc. ©2015

[20] https://github.com/kiwenlau/hadoop-cluster-docker

[21] www.michael-noll.com/tutorials/

[22] Ndapandula Nakashole, Gerhard Weikum and Fabian Suchanek, “PATTY: A Taxonomy of Relational Patterns with Semantic Types”, International Conference on Empirical Methods in Natural Language Processing (EMNLP 2012)

[23] http://wiki.dbpedia.org/services-resources/documentation/datasets#ArticleCategories

[24] http://wiki.dbpedia.org/services_resources/documentation/datasets#mappingbasedobjectsuncleaned

[25] http://wiki.dbpedia.org/services-resources/documentation/datasets#NIFContext

[26] http://wiki.dbpedia.org/services-resources/documentation/datasets#NIFTextLinks

[27] http://wiki.dbpedia.org/dbpedia-version-2016-04

[28] http://mappings.dbpedia.org/server/ontology/classes/

[29] https://developers.google.com/knowledge-graph/

[30] Sean Bechhofer, University of Manchester, Frank van Harmelen, Free University Amsterdam, Jim Hendler, University of Maryland, Ian Horrocks, University of Manchester, Deborah L. McGuinness, Stanford University, Peter F. Patel-Schneider, Bell Labs Research, Lucent Technologies, Lynn Andrea Stein, Franklin W. Olin College of Engineering, "OWL Web Ontology Language Reference", W3C Recommendation 10 February 2004

Appendix A: Glossary

1. **RDF:** Resource Description Framework is a standard model for data interchange on the Web. RDF has features that facilitate data merging even if the underlying schemas differ, and it specifically supports the evolution of schemas over time without requiring all the data consumers to be changed. RDF extends the linking structure of the Web to use URIs to name the relationship between things as well as the two ends of the link (this is usually referred to as a "triple").

2. **URI:** In information technology, a Uniform Resource Identifier (URI) is a string of characters used to identify a resource. Such identification enables interaction with representations of the resource over a network, typically the World Wide Web, using specific protocols.

3. **CoreNLP:** Stanford CoreNLP provides a set of human language technology tools. It can give the base forms of words, their parts of speech, whether they are names of companies, people, etc., normalize dates, times, and numeric quantities, mark up the structure of sentences in terms of phrases and syntactic dependencies, indicate which noun phrases refer to the same entities, indicate sentiment, extract particular or open-class relations between entity mentions, get the quotes people said, etc.

4. **Anaphora:** In linguistics, anaphora is the use of an expression whose interpretation depends upon another expression in context.

5. **OWL:** The W3C Web Ontology Language (OWL) is a Semantic Web language designed to represent rich and complex knowledge about things, groups of things, and relations between things. OWL is a computational logic-based language such that knowledge expressed in OWL can be exploited by computer programs, e.g., to verify the consistency of that knowledge or to make implicit knowledge explicit.

6. **Named Entity Recognition:** Named-entity recognition (NER) is a subtask of information extraction that seeks to locate and classify named entities in text into pre-defined categories such as the names of persons, organizations, locations, expressions of times, quantities, monetary values, percentages, etc.

7. **Google Knowledge Graph:** The Knowledge Graph is a knowledge base used by Google and its services to enhance its search engine's results with information gathered from a variety of sources. This information is presented to users in a box to the right of search results.

8. **YAGO:** YAGO (Yet Another Great Ontology) is an open source knowledge base developed at the Max Planck Institute for Computer Science in Saarbrücken. It is automatically extracted from Wikipedia and other sources.

9. **Turtle:** Turtle (Terse RDF Triple Language) is a format for expressing data in the Resource Description Framework (RDF) data model with a syntax like SPARQL. RDF, in turn, represents information using "triples", each of which consists of a subject, a predicate, and an object. Each of those items is expressed as a Web URI.

10. **Freebase:** Freebase was a large collaborative knowledge base consisting of data composed mainly by its community members. It was an online collection of structured data harvested from many sources, including individual, user-submitted wiki contributions.

11. **Parser:** A parser is a software component that takes input data (frequently text) and builds a data structure – often some kind of parse tree, abstract syntax tree or other hierarchical structure, giving a structural representation of the input while checking for correct syntax.

12. **Language Model:** A statistical language model is a probability distribution over sequences of words. Having a way to estimate the relative likelihood of different phrases is useful in many natural language processing applications, especially ones that generate text as an output.

13. **Ambiguity:** Ambiguity is a type of uncertainty of meaning in which several interpretations are plausible. It is thus an attribute of any idea or statement whose intended meaning cannot be definitively resolved according to a rule or process with a finite number of steps.

Appendix B: Awards and Competitions

1. Secured 1st Prize in Computer Engineering for the project, at the Tantravihar 2018, Annual Intracollegiate BE Project Competition, organized on 2nd and 3rd April 2018, at Vidyalankar Institute of Technology.
2. Presented a poster in RACEM (Recent Advances and Challenges in Engineering and Management) 2017 at Vidyalankar Institute of Technology on 22nd and 23rd December 2017.