Natural Language Processing

A Survey

Jon Abdulloev
Earlham College
Richmond, Indiana
aabdul15@earlham.edu

ABSTRACT

This paper provides a brief overview of the current state of the field of Natural Language Processing (NLP), primarily focusing on Natural Language Understanding (NLU), Natural Language Interface to Database Systems (NLIDBS), and Information Extraction (IE).

KEYWORDS

Natural Language Processing, Natural Language Understanding, Natural Language Interface to Database System, Information Extraction, Knowledge-base, Natural Language Toolkit

1 INTRODUCTION

In the past couple of decades, there has been a significant growing amount of research on Natural Language Processing (NLP) which has been largely motivated by its enormous applications. Some of the well-known systems that use NLP techniques include Siri from Apple [7], IBM Watson [6] and Wolfram|Alpha [9]. NLP is a broad topic with various subtopics as shown by the Venn diagram[1] in Figure 1 which is a visualization of the categorization of the subtopics of NLP. In this survey paper, we will discuss relevant and interrelated research papers belonging to the field of NLP, specifically pertaining to Natural Language Understanding (NLU), Natural Language Interface to Database Systems (NLIDBS), and Information Extraction (IE). We will also discuss the Natural Language Toolkit (NLTK) [11], a leading platform for building Python programs to work with human language data which will be the primary tool we will use to work with Natural Language Processing.

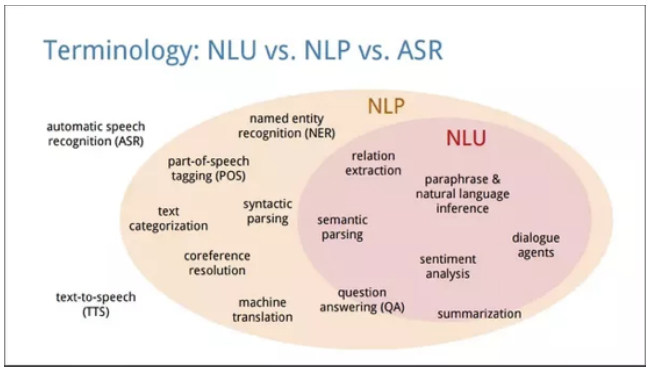


Figure 1: Visualization of the scope of NLP

[1]https://www.quora.com/What-is-the-difference-between-natural-language-processing-NLP-and-natural-language-understanding-NLU

2 NATURAL LANGUAGE INTERFACE TO DATABASE SYSTEMS

Much research has been conducted in recent years on building efficient NLIDBSs that allow people without SQL backgrounds to query a SQL database in natural language. For instance, a research team at Salesforce developed WikiSQL with the aim of democratizing SQL so databases can be queried in natural language. WikiSQL is a large crowd-sourced dataset for developing natural language interfaces for relational databases that generates structured queries from natural language using Reinforcement Learning. In this section, we analyze some of the techniques used in NLIDBSs.

2.1 Semantic Tractability

Popescu et al. discuss an approach to modern NLIDBS which involves composing statistical parsing with semantic tractability [15]. They use the term semantically tractable to describe "easy-to-understand" questions where the words or phrases correspond to database elements or constraints on join paths. They introduced the PRECISE algorithm which takes a lexicon and a parser as input. The authors define a lexicon as a 3-tuple (T, E, M), where T is the set of tokens, E is the set of database elements, wh-values[2] and join paths[3], and M is a subset of $T \times E$ - a binary relation between tokens and database elements. Given an English question, PRECISE maps it to one or more corresponding SQL queries using the attachment function $F_{L,q} : T \rightarrow T$, where L is the lexicon, q is a question, and T is the set of tokens in the lexicon. Tokens can be characterized as linguistic units such as words, punctuation, numbers or alphanumerics [12].

The authors discussed their observations about sentence structures and revealed that nouns, adjectives, and adverbs in semantically tractable questions refer to database relations, attributes or values. The attributes and values in a question tend to "pair up" naturally to indicate equality constraints in SQL. However, values may be paired with implicit attributes that do not appear in the question. For example, *cuisine* is an implicit attribute in *"What are the Chinese restaurants in Seattle?"*. However, implicitness does not apply to values as the question "What are restaurants with cuisine in Seattle?" does not make sense. Furthermore, Popescu et al. found that a preposition indicates a join between the relations corresponding to the arguments of the preposition. For example, the preposition 'from' in the question *"what airlines fly from Boston to Chicago?"* connects the value 'Boston' (in the relation 'cities') to the relation 'airlines'. Hence, 'from' indicates a join between 'airlines' and 'cities'.

[2]wh-values correspond to wh-words such as *what* and *where.*
[3]A join path is a set of equality constraints between a sequence of database relations.

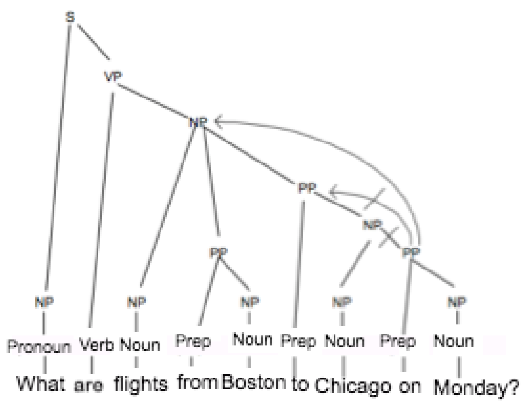


Figure 2: Example of an erroneous parse tree corrected by PRECISE's semantic over-rides [15]

2.2 Semantic Matching

Singh et al. developed an intelligent interface using a semantic matching technique that translates Natural Language Query (NLQ) into SQL using a set of production rules and a data dictionary [16]. The data dictionary consists of semantic sets for relations and attributes. The authors use a MySQL database management system as their backend and explain the purpose of the set of NLP Rules and some predefined structures to train the system accordingly. In order to convert NLQ into SQL Query, the NLQ goes through a series of steps. The system consists of the following components:

(1) User Interface: The user interacts with the system via a GUI and types the NLQ.
(2) Lowercase Conversion: It checks all the words in the user question and converts it into lower case
(3) Tokenization: It scans the whole question and then splits the question string into its constituent tokens and gives an order number to each token identified.
(4) Escape word removal: The extra/stop words are removed which are not needed in the analysis of the query and stores the number of identified SQL elements in arrays.
(5) Part Of Speech Tagger: The tokens are then classified into one or more of a set of lexical or part-of-speech categories such as nouns, pronouns, verb and string/integer variables. For example, given the sentence "The ball is red", the output of the POS tagger would be `The/AT ball/NN is/VB red/JJ`. The tokens are then further classified into SQL elements in arrays.
(6) Relations-Attributes-Clauses Identifier: Now the system classifies the tokens into relations, attributes and clauses on the basis of tagged elements and also separates the Integer and String values to form clauses.
(7) Ambiguity Removal: It removes all the ambiguous attributes that exists in multiple relation with the same attribute name and maps it with the correct relation.
(8) Query Formation: After the relations, attributes and clauses are extracted, the final query is constructed.
(9) Query Execution and Data Fetching: The query is then executed and data is fetched from the database.
(10) Results: The final query result is displayed to the user on the GUI.

Singh et al. explain their use of semantic sets of NLP rules for semantic matching and training of the system:

- The escape word set (E_w) which contains the list of stop words that occur in the NLQ.
- The Expression mapping set (E_{map}) which contains the list of conditional clauses which might occur in NLQ.
- A Noun set (N) that contains all elements which are nouns.
- A Verb set (V) that contains all elements which are verbs and act as criteria for forming clauses.
- The semantic set (S) that contains the list of all possible semantics related to table names and fields in the database.
- A Variable set (Va) that consists of all String and Integer variables used in forming clauses.
- A Relation set (R) that consists of relation names that are encountered in user query or are added by analyzing the attribute names present in the NLQ.
- An Attribute set (A) that contains all attributes present in the user query.
- An Ambiguity check set (A_c) that contains all attribute fields whose name are used in multiple relation as a field name excluding keys.
- The Conjunction training set (C_T) which consists of the list of Conjunctive clauses which occur in the NLQ and this set is generated at run-time - whenever a new conjunctive clause is encountered, it is appended to the existing Conjunction training set.

Singh et al. argue that the primary advantage of these sets is that they can be expanded whenever some new knowledge is discovered.

2.3 Using Clarification Dialogs to Resolve Ambiguities

Another approach to NLIDBs simply asks the user for clarification instead of using complex semantics based techniques. Kaufmann et al. present Querix, a domain-independent natural language interface for the Semantic Web that resolves ambiguities in natural language to query ontologies by using clarification dialogs [10].

The system consists of seven main parts: a user interface, an ontology manager, a query analyzer, a matching center, a query generator, a dialog component, and an ontology access layer.

The user interface allows the user to enter full NLQs and choose the ontology to be queried. When an ontology is chosen and loaded into Querix, the ontology manager enhances the resources' labels by obtaining synonyms from WordNet. The query analyzer uses the Stanford Parser [Klein and Manning, 2002] to generate a syntax tree for the NL query from which the sequence of the main word categories Noun (N), Verb (V), Preposition (P), Wh-Word (Q), and Conjunction (C) are extracted and used to build a query skeleton. For example, the query skeleton for the query "What are the population sizes of cities that are located in California?" is Q-V-N-P-N-Q-V-P-N. The query analyzer also uses WordNet which provides synonyms for all the nouns and verbs in the query's parse tree. The authors implement a cost function in order to obtain the most appropriate synonyms as WordNet usually suggests too many. Querix then matches the query skeleton with the synonym-enhanced triples in the ontology and for each query it does the following:

(1) It tries to match the extracted query skeleton with a small set of heuristic patterns, such as Q-V-N representing "what are the population sizes" or N-P-N representing "population sizes of cities". The subject-property-object patterns of a query is identified and valid patterns in the query skeleton have to overlap with regard to their first or last word category to enable the joining of the triples in step 3.
(2) It searches for all matches between the synonym-enhanced nouns and verbs of the input query with the resources and their synonyms in the ontology.
(3) It matches the triple patterns identified by step 1 and the resources found by step 2. This matching is enabled by storing each word category of the query skeleton together with its NL word form in step 1 and each noun as well as verb of the NL query with its synonyms from WordNet by the query analyzer. Additionally, the matching is controlled by the domain and range information of the ontology. After identifying all possible triples in the sentence skeleton and combining them to the ontology's resources, the query generator composes SPARQL queries from the joined triples. As each matching step comprises a cost function, the query generator produces a ranked list of SPARQL queries.

In order to resolve ambiguities whereby there are several different solutions to a query with the same cost score, the system's dialog component consults the user by showing a menu from which the user can choose the meaning she/he intended. The meanings offered are based on the possible triples identified by the matching center.

3 INFORMATION EXTRACTION

Information extraction is defined as the extraction of information from a text in the form of text strings and processed text strings which are placed into slots labeled to indicate the kind of information that can fill them [4]. Wikipedia gives a formal description by describing it as the task of automatically extracting structured information from unstructured and/or semi-structured machine-readable documents.

A wide range of approaches exist for parsing natural language sentences including simple part-of-speech tagging and context-free grammars, and more advanced techniques such as Lexical Functional Grammars, Head-Driven Phrase Structure Grammars, Link Grammars, and stochastic approaches. In this section we will discuss two simple methods involving generic patterns and one advanced approach which uses Link Grammars.

3.1 Generic Patterns

Pantel et al. proposed a weakly-supervised, general purpose, and accurate algorithm called Espresso which exploits generic patterns by filtering incorrect instances using the Web [14]. Generic patterns are broad coverage noisy patterns that have high recall and low precision. The system can extract *is-a* and *part-of* binary relations. For instance, the pattern "X of Y" can ambiguously refer to part-of, is-a and possession relations. However, "X of Y" can be used to extract relation instances such as "wheel of the car" correctly but it incorrectly extracts one such as "house of representatives". Given seed instances of a particular binary relation, the program iterates between the following three phases:

- Pattern induction - in this phase, a set of surface patterns P is inferred which then connects as many of the seed instances as possible in a given corpus. The authors explain that they chose the algorithm described in (Ravichandran and Hovy 2002) but modified it so that for each input instance x and y, they first retrieve all sentences containing x and y, and then generalize the sentences into a set of new sentences $S_{x,y}$ by replacing all terminological expressions by a terminological label, *TR*.
- Pattern ranking/selection - this is where all patterns in P are ranked according to reliability r_π and all but the top-k are discarded, where k is set to the number of patterns from the previous iteration plus one.
- Instance extraction - in the final phase the set of instances I that match any of the patterns in P are retrieved from the corpus.

After being able to extract several standard and specific semantic relations such as is-a, part-of, succession, reaction, and production, the authors hope to make use of the relations in applications like question answering for future work. They also hope to improve the system's performance by investigating the use of WordNet to automatically learn selectional constraints[4] on generic patterns.

Etzioni et al. introduced KNOWITALL, a system that aims to automate the tedious process of extracting large collections of facts from the web in an autonomous, domain-independent, and scalable manner [5]. KNOWITALL associates a probability with each fact enabling it to trade off precision and recall. Each KNOWITALL module runs as a thread and communication between modules is accomplished by asynchronous message passing. Its main modules are:

- Extractor: The Extractor is the main module of the system. Whenever a new class or relation is added to KNOWITALL's ontology, the Extractor uses generic, domain-independent rule templates to instantiate a set of information extraction rules for that class or relation. For example, the generic template "NP1 such as NPList2" indicates that the head of each simple noun phrase (NP) in NPList2 is an instance of the class named in NP1. This template can be instantiated to find city names from such sentences as "We provide tours to cities such as Paris, Nice, and Monte Carlo" from which three instances of the class City would be extracted.
- Search Engine Interface: This involves automatically formulating queries based on the extraction rules, whereby each rule has an associated search query composed of the keywords in the rule. For example, the above rule would issue the query "cities such as" to a search engine, download each of the pages named in the engine's results in parallel, and apply the Extractor to the appropriate sentences on each downloaded page.

[4]Selectional constraints are constraints on the range of concepts with which a concept can have a particular relationship

- Assessor: The Assessor uses a form of point-wise mutual information (PMI) between words and phrases that is estimated from web search engine hit counts in a manner similar to Turney's PMI-IR algorithm [18]. For example, if the Extractor has proposed "Liege" as the name of a city and the PMI between "Liege" and a phrase like "city of Liege" is high, then this is evidence that "Liege" is a valid instance of the class City. The Assessor computes the PMI between each extracted instance and multiple phrases associated with cities. These mutual information statistics are combined via a Naive Bayes Classifier. Hence in order to improve the precision of the information that is extracted from the web, statistics computed by querying search engines are used to assess the probability of the correctness of the Extractor's conjectures. The Assessor measures co-occurrence statistics of candidate extractions with a set of discriminator phrases. For example, if Cuba Gooding is an Actor, then we would expect the phrase "Cuba Gooding starred in" to be more prevalent on the web than if he has never acted. Thus, "X starred in" is a pattern for a discriminator phrase.
- Database: Etzioni et al. store information (including metadata such as the rationale for and the confidence in individual assertions) in a commercial RDBMS. They argue that the advantage of this is that the database is persistent and scalable, supporting rapid-fire updates and queries.

A sample of the syntactic patterns that underlie KNOWITALL's rule templates is shown in Figure 3, where the "," in the patterns indicates an optional comma after NP1.

NP1 {","} "such as" NPList2
NP1 {","}"and other" NP2
NP1 {","}"including" NPList2
NP1 "is a" NP2
NP1 "is the" NP2 "of" NP3
"the" NP1 "of" NP2 "is" NP3

Figure 3: A rule Template [5].

The authors credit Marti Hearst's hyponym patterns [8] for some of their rule templates. They then demonstrate how these patterns can be used as extraction rules: if NP1 in the first pattern is bound to the name of a class in the ontology, then each simple noun phrase in NPList2 is likely to be an instance of that class. When this pattern is used for the class Country it would match a sentence that includes the phrase "countries such as X, Y, and Z" where X, Y, and Z are names of countries. The same pattern is used to generate rules to find instances of the class Actor, where the rule looks for "actors such as X, Y, and Z".

Etzioni et al. use these patterns as the basis for extraction rule templates and add syntactic constraints that look for simple noun phrases. Using such noun phrase analysis, the system can recognize China as an instance of the class Country in the sentence "China is a country in Asia" since the word "country" is the head of a simple noun phrase. On the other hand the system can also detect that "country" is not the head of a noun phrase in the sentence "Garth Brooks is a country singer" so Garth Brooks won't be extracted as the name of a country.

The authors demonstrate how an example of a rule template (shown in the Figure 4) is instantiated for a particular class. The Extractor generates a rule for Country from this rule template by substituting "Country" for "Class1", plugging in the plural "countries" as a constraint on the head of NP1. This produces the rule shown in the Figure 5.

```
Rule Template:
  NP1 "such as" NPList2
  & head(NP1)= plural(name(Class1))
  & properNoun(head(each(NPList2)))
  =>
  instanceOf(Class1,head(each(NPList2)))
```

Figure 4: This generic rule template is instantiated for a particular class in the ontology to create an extraction rule that looks for instances of that class [5].

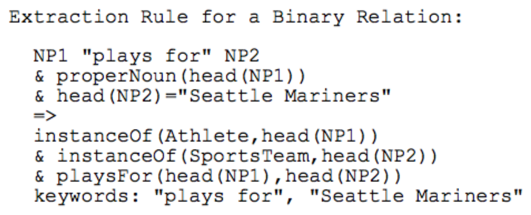
```
Extraction Rule:

  NP1 "such as" NPList2
  & head(NP1)="countries"
  & properNoun(head(each(NPList2)))
  =>
  instanceOf(Country,head(each(NPList2)))
  keywords: "countries such as"
```

Figure 5: This extraction rule looks for web pages containing the phrase "countries such as". It extracts any proper nouns immediately after that phrase as instances of Country [5].

The Extractor then takes the literals of the rule as the "keywords" of the rule, which KNOWITALL sends to a search engine as a query, in this case the search query is the phrase "countries such as". It also uses the Brill tagger [2] to assign part-of-speech tags and identifies noun phrases with regular expressions based on the part-of-speech tags. The Extractor then matches the rule in Figure 4 to each tagged sentence. NP1 matches a simple noun phrase; it must be immediately followed by the string "such as"; following that must be a list of simple NPs. If the match is successful, the Extractor applies constraints from the rule. The head of NP1 must match the string "countries". The Extractor checks that the head of each NP in the list NPList2 has the capitalization pattern of a proper noun. Any NPs that do not pass this test are ignored. If all constraints are met, the Extractor creates one or more extractions: an instance of the class Country for each proper noun in NPList2.

The Extractor can also utilize rules for binary or n-ary relations. Figure 6 shows a rule that finds instances of the relation `playsFor(Athlete, SportsTeam)`. This particular rule has the second argument bound to an instance of SportsTeam, "Seattle Mariners", which KNOWITALL has previously added to its Database.

As future work, Etzioni et al. plan to extend KNOWITALL in several important ways to further investigate to what extent the goal of developing a completely open-ended, extensible, information-extraction system can be achieved. They also want to automatically



```
Extraction Rule for a Binary Relation:

  NP1 "plays for" NP2
  & properNoun(head(NP1))
  & head(NP2)="Seattle Mariners"
  =>
  instanceOf(Athlete,head(NP1))
  & instanceOf(SportsTeam,head(NP2))
  & playsFor(head(NP1),head(NP2))
  keywords: "plays for", "Seattle Mariners"
```

Figure 6: This extraction rule finds instances of athletes that play for a sports team. The second argument is bound such that it looks for athletes that play for the Seattle Mariners [5].

extend the ontology by using domain-independent extraction rules to identify new classes in a given domain.

3.2 Deep Linguistic Structures

Suchanek et al. use a Link Grammar Parser that is based on context-free grammar [17]. A linkage is a connected planar undirected graph whose nodes are the words of a sentence and the edges represent connectors. For example, in Figure 8, the connector *subj* marks the link between the subject and the verb of the sentence.

A link grammar specifies which word may be linked by which connector to the preceding and following words. This ensures that the linkage fulfills certain linguistic constraints. The parser also assigns part-of-speech tags. For example, in Figure 7, for the sentence "Chopin was great among the composers of this time", the suffix ".n" identifies "composers" as a noun.

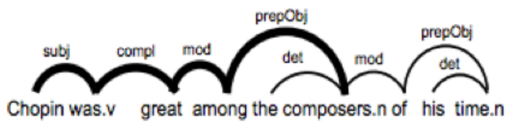


Figure 7: A simple linkage [17].

A pattern is a linkage in which two words have been replaced by placeholders. Figure 8 shows a sample pattern with the placeholders "X" and "Y" where the unique shortest path from one placeholder to the other, called the bridge, is marked in bold. The most important component of a pattern is its bridge because if the bridge of the pattern appears in the linkage regardless of the nouns and adjectives, then a pattern matches a linkage. Hence, for instance, the pattern in Figure 8 matches the linkage in Figure 7 since the bridge of the pattern occurs in the linkage.

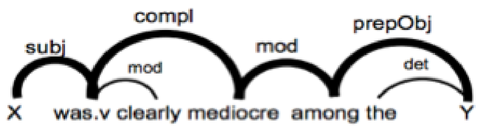


Figure 8: A pattern in a linkage [17].

The authors also implement a function to categorize a pair of words for the target relation as one of the following:

(1) an example (e.g. "Chopin" / "1810")
(2) a counterexample (e.g. if "Chopin" / "1810" is an example, then "Chopin" / "2000")
(3) a candidate (e.g. "Mozart" / "2000" if there are no examples or counterexamples)
(4) none of the above.

The core algorithm has three phases:

- Discovery Phase - where it seeks linkages in which an example pair appears. It replaces the two words by placeholders, thus determining a pattern that expresses the target relation. This is where the authors apply their two machine learning techniques, the adaptive kNN classifier and an SVM classifier, to collect positive patterns and then produce a counterexample by analyzing the corpus again to find all linkages that match a positive pattern.
- Training Phase - where statistical learning is applied to learn the concept of positive patterns, the result of which is a classifier for patterns.
- Testing Phase - where the algorithm runs the corpus again and finds all possible patterns by replacing two words by placeholders.

In their conclusion, the authors explain that the linkages allow for more sophisticated ways of resolving anaphoras or matching patterns. However, their own implemented system acquired and exploited new corpora on its own (e.g. newspapers) and used its knowledge to acquire and structure its new knowledge more efficiently.

4 KNOWLEDGE BASES

Mahesh et al. explain that a knowledge base is an integral part of a NLP system due to its use in the extraction and acquisition of meanings from corpora. Therefore, an NLP system must have a substantial amount of knowledge about the world and the domain of discourse in order for it to make sense of given texts.

The knowledge-based approach to NLP involves acquiring and representing world and domain-specific knowledge using linguistic knowledge, and consequently applying that knowledge to solve well-known problems in NLP such as ambiguity resolution and making inferences. In this section we discuss the NLP problems and present some knowledge-based algorithms that can be used to solve them.

4.1 Knowledge-Based NLP

Mahesh et al. show how general knowledge of the world together with specific knowledge can be used to resolve syntactic and semantic ambiguities and make necessary inferences [13]. They illustrate some of the well-known problems and describe methods using knowledge-based selection to solve the problems. They also introduce KB-NLP systems that use those methods to address the problems in NLP. Some of the problems and the respective KB-NLP systems that address them are described below.

(1) Syntactic Ambiguity which can be of two kinds:
    - Category (or part of speech) ambiguity which is the simple case that can be resolved using syntactic linguistic knowledge.

- Attachment ambiguity which often requires semantic and world knowledge to resolve.

(2) Word Sense Ambiguity (Lexical semantic ambiguity) - To resolve this type of ambiguity, the lexical meaning that best fulfills the constraints on the composition of the text's possible meanings is selected. World knowledge is used to derive selectional constraints on how meanings can be composed with one another. In lieu of world knowledge, statistical methods can also be used to derive the constraints in narrow domains where sufficient training data on word senses is available. The process of checking selectional constraints is implemented as a search in a semantic network or conceptual ontology and can be expensive if large networks are involved. Therefore, constraints are only checked for those pairs of words in the text that are syntactically related to one another. For example, in the sentence "The bar was frequented by gangsters" the word "bar" has a word sense ambiguity: it can mean either an oblong piece of a rigid material or a counter at which liquors or light meals are served. Using knowledge of selectional constraints, a KB-NLP system can correctly resolve the word sense ambiguity and determine that "bar" in this case is a liquor-serving place in order for it to be frequented by people.

(3) Syntactic and Semantic Analysis - To resolve this type of ambiguity, Mahesh et al. list and describe possible KB-NLP systems as solutions. These systems use methods for combining linguistic knowledge with world knowledge either in the representations or during processing. Some of the systems include:
   - Jurafsky's SAL (1992) integrated representations of all the different kinds of knowledge in a monolithic knowledge base of integrated constructs called grammatical constructions.
   - McRoy and Hirst (1990) is a race-based architecture which simulates syntax and semantics running in parallel by associating time costs with each operation. The model is able to resolve a variety of ambiguities by simply selecting whichever alternative that minimizes the time cost.
   - Mikrokosmos is a knowledge-based machine translation[5] (KBMT) system that has focused particularly on lexical disambiguation (Onyshkevych and Nirenburg, 1995; Mahesh and Nirenburg, 1995b; Beale, Nirenburg, and Mahesh, 1996).

Mahesh et al. propose a detailed solution to the common PP-attachment problem, an attachment ambiguity involving prepositional phrases (PP). They propose applying selectional constraints on the selection of one of the possible attachments. The selectional constraints which are derived from semantic and world knowledge are used to compose the meanings of the two child units being attached with meanings of each of the possible parent syntactic units. They explain that such selectional constraints are typically represented in the form of a permissible range of fillers for slots in frames representing the meanings. The potential filler, which refers to the meaning of the child unit, is compared against the selectional constraint by a fuzzy match function which computes a weighted distance between the two meanings in a semantic or ontological network of concepts. The closer the two meanings are in the network, the higher the score the function assigns to the choice. The algorithm then combines the scores from different constraints for the same choice by applying a mathematical function. The resulting combined scores are used to select the best choice according to the knowledge of selectional constraints.

In Figure 9, for example, the PP "with the horse" can be attached in any one of the three ways shown in the tree. Knowledge of selectional constraints can be used to infer that an instrument of seeing must be an optical instrument and since a horse is not an optical instrument, the attachment to the verb phrase (VP) "saw" can be ruled out, resulting in an interpretation where the horse is an accompanier to "the man".

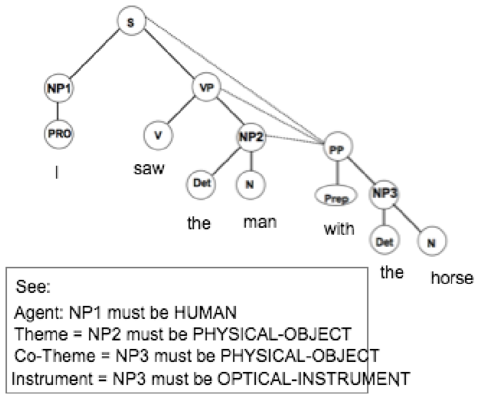

S
NP1
VP
PRO
V
NP2
PP
I
saw
Det
N
Prep
NP3
the
man
with
Det
N
the
horse
See:
Agent: NP1 must be HUMAN
Theme = NP2 must be PHYSICAL-OBJECT
Co-Theme = NP3 must be PHYSICAL-OBJECT
Instrument = NP3 must be OPTICAL-INSTRUMENT

Figure 9: A PP Attachment Ambiguity showing an ambiguous parse tree and selectional constraints on attachments [13].

The authors further explain that knowledge-based solutions are also applicable to a variety of other problems in NLP such as thematic analysis, topic identification, discourse and context tracking, and temporal reasoning. Knowledge-based methods also play a crucial role in natural language generation for problems such as lexical choice and text planning.

4.2 Universal Truths

Chaudhri et al. [3] curate a knowledge base for a biology textbook using a knowledge engineering technique that

(1) re-formulates sentences as Universal Truths so that the surface form of knowledge is closer to the knowledge to be represented
(2) extracts the knowledge into a linguistically motivated ontology
(3) defines how various conceptual distinctions are expressed in natural language using a set of syntactic and semantic guidelines.

[5] In machine translation, the goal is to have the computer translate the given text in one natural language to fluent text in another language without any human in the loop. This is one of the most difficult tasks in NLP and has been tackled in a lot of different ways over the years. Almost all MT approaches use POS tagging[6] and parsing as preliminary steps [12].

The technique used to convert sentences into universal truths involves identifying the structure of a Universal Truth as pair containing X (a noun phrase denoting a concept) and Y (a clause or verb phrase denoting information that is true about the concept) in one of the following forms: (a) Every X Y (b) In X, Y (c) During X, Y. The Universal Truths are then converted into Knowledge Representation Plans which are sets of literals that would appear in the consequent of an existential rule[7].

The authors claim that the plans for a Knowledge Base are similar to the design specification or the pseudo code for a program - writing the plans first helps an encoder to think through the overall design of the representation before entering it into the Knowledge Base. The final stages entails entering the plans into the Knowledge Base using a graphical interface to create the knowledge representation and the partial concept graph for signal reception.

5 NATURAL LANGUAGE TOOLKIT (NLTK)

The Natural Language Toolkit [11] is a suite of Python modules providing many NLP data types (including tokens, tags, chunks, trees, and feature structures), processing tasks, corpus samples and readers, together with animated algorithms, tutorials, and problem sets. The toolkit was created by Steven Bird, Edward Loper, and Ewan Klein, and supports research and teaching in computational linguistics and natural language processing. It is distributed under the GPL open source license.

One of the advantages of using NLTK is that it is entirely self-contained. It provides (i) convenient functions and wrappers that can be used as building blocks for common NLP tasks as well as (ii) raw and pre-processed versions of standard corpora used in NLP literature and courses [12].

Madnani provides an overview of the NLTK corpora that are used widely in the NLP research community in [12]. Some of these include:

- The Brown Corpus of Standard American English - consists of one million words of American English texts printed in 1961. For the corpus to represent as general a sample of the English language as possible, 15 different genres were sampled such as Fiction, News and Religious text. Subsequently, a POS-tagged[8] version of the corpus was also created with substantial manual effort.
- Gutenberg Corpus - a selection of 14 texts chosen from Project Gutenberg and it contains a total of 1.7 million words.
- Stopwords Corpus - list of 2400 stop words across 11 different languages (including English).

The NLTK comes shipped with corpora and corpus readers that understand the file structure of the corpus and load the data into Python data structures. For example, the code in Figure 10 reads a part of the Brown Corpus and in the process of tokenization it prints a list of tuples, each of which consists of a word and its tag [1].

[7]An existential rule is a rule whose antecedent has one variable that is universally quantified, and whose consequent has one or more variables that are existentially quantified

[8]Part-of-Speech (POS) tagging involves associating a symbol representing a lexical category such as NN (Noun), VB(Verb), JJ (Adjective), AT(Article) with each word in a sentence.

```
>>> for sent in brown.tagged('a'):
...     print sent
[('The', 'at'), ('Fulton', 'np-tl'),
('County', 'nn-tl'), ('Grand', 'jj-tl'),
('Jury', 'nn-tl'), ('said', 'vbd'), ...]
```

Figure 10: This piece of code reads a part of the Brown Corpus and prints a list of tuples [1].

Bird described how the NLTK can be used to carry out tokenization, stemming, tagging, chunking and parsing [1]. For tagging, the regular expression tagger can be used to assign a tag to a token according to a series of string patterns. For example, the tagger in Figure 11 assigns `cd` to cardinal numbers, `nns` to words ending in the letter `s`, and `nn` to everything else.

```
>>> patterns = [
...     (r'\d+(.\d+)?$', 'cd'),
...     (r'\.*s$', 'nns'),
...     (r'.*', 'nn')]
>>> simple_tagger = tag.Regexp(patterns)
>>> list(simple_tagger.tag(tokens))
[('John', 'nn'), ('saw', 'nn'),
 ('3', 'cd'), ('polar', 'nn'),
 ('bears', 'nns'), ('.', 'nn')]
```

Figure 11: This piece of code assigns `cd` to cardinal numbers, `nns` to words ending in the letter `s`, and `nn` to everything else [1].

The `tag.Unigram` class provides a more advanced tagger which uses a statistical tagging algorithm to assign to a particular token the most likely tag. For example, it will assign the tag `jj` to any occurrence of the word *frequent*, since *frequent* is used as an adjective (e.g. a *frequent* word) more often than it is used as a verb (e.g. I *frequent* this cafe). Before a unigram tagger can be used, it must be trained on a corpus. During the tagging of a new tag, it will assign the default tag `None` to any token that was not encountered during training.

```
>>> text = "John saw the books on the table"
>>> tokens = list(tokenize.whitespace(text))
>>> list(unigram_tagger.tag(tokens))
[('John', 'np'), ('saw', 'vbd'),
 ('the', 'at'), ('books', None),
 ('on', 'in'), ('the', 'at'),
 ('table', None)]
```

Figure 12: The unigram tagger [1].

Some other taggers included in the NLTK are the Brill tagger (contributed by Christopher Maloof) and the HMM tagger (contributed by Trevor Cohn).

The NLTK also provides ways to chunk data. Chunking is a technique for shallow syntactic analysis of (tagged) text. Chunk data can be loaded from files that use the common bracket or IOB notations. We can define a regular-expression based chunk parser for use in chunking tagged text. NLTK also supports simple cascading of chunk parsers.

NLTK provides several parsers for context-free phrase-structure grammars. Grammars can be defined using a series of productions as shown in Figure 13.

```
>>> grammar = cfg.parse_grammar('''
...     S -> NP VP
...     VP -> V NP | V NP PP
...     V -> "saw" | "ate"
...     NP -> "John" | Det N | Det N PP
...     Det -> "a" | "an" | "the" | "my"
...     N -> "dog" | "cat" | "ball"
...     PP -> P NP
...     P -> "on" | "by" | "with"
...     ''')
```

Figure 13: Defining grammars using a series of productions [1].

After defining a grammer, we can tokenize and parse a sentence with a recursive descent parser, as shown in Figure 14.

```
>>> text = "John saw a cat with my ball"
>>> sent = list(tokenize.whitespace(text))
>>> rd = parse.RecursiveDescent(grammar)


>>> for p in rd.get_parse_list(sent):
...     print p
(S:
  (NP: 'John')
  (VP:
    (V: 'saw')
    (NP:
      (Det: 'a')
      (N: 'cat')
      (PP: (P: 'with')
        (NP: (Det: 'my') (N: 'ball'))))))
(S:
  (NP: 'John')
  (VP:
    (V: 'saw')
    (NP: (Det: 'a') (N: 'cat'))
    (PP: (P: 'with')
      (NP: (Det: 'my') (N: 'ball')))))
```

Figure 14: Iterate over all the parses that is generated [1].

6 CONCLUSION

Although there has been significant research in the field of NLP, we are still far from developing fully linguistically conscious AI systems that will be able to cope with the challenges of human languages. This survey paper presents the challenges that are associated with querying information through NLIDBSs, collecting information through Information Extraction, and building a knowledge base in order to better solve NLP problems.

This paper also discusses the NLTK used in NLP, as well as the several methods and algorithms that are solutions to NLP problems. In the future, it is likely that several of these approaches to solving NLP problems will be combined to resolve the challenges of natural language such as ambiguity.

REFERENCES

[1] Steven Bird. 2006. NLTK: the natural language toolkit. In *Proceedings of the COLING/ACL on Interactive presentation sessions*. Association for Computational Linguistics, 69–72.

[2] Eric Brill. 1994. Some advances in transformation-based part of speech tagging. *arXiv preprint cmp-lg/9406010* (1994).

[3] Vinay K Chaudhri, Nikhil Dinesh, Daniela Inclezan, and MIAMIOH EDU. 2013. Three lessons for creating a knowledge base to enable explanation, reasoning and dialog. In *Proceedings of the Second Annual Conference on Advances in Cognitive Systems ACS*, Vol. 187. 203.

[4] Nancy Chinchor and Elaine Marsh. 1998. Muc-7 information extraction task definition. In *Proceeding of the seventh message understanding conference (MUC-7), Appendices*. 359–367.

[5] Oren Etzioni, Michael Cafarella, Doug Downey, Stanley Kok, Ana-Maria Popescu, Tal Shaked, Stephen Soderland, Daniel S Weld, and Alexander Yates. 2004. Web-scale information extraction in knowitall:(preliminary results). In *Proceedings of the 13th international conference on World Wide Web*. ACM, 100–110.

[6] David A. Ferrucci. 2011. IBM's Watson/DeepQA. *SIGARCH Comput. Archit. News* 39, 3 (June 2011), –. https://doi.org/10.1145/2024723.2019525

[7] Michael Galeso. 2017. *Apple Siri for Mac: An Easy Guide to the Best Features*. CreateSpace Independent Publishing Platform, USA.

[8] Marti A Hearst. 1992. Automatic acquisition of hyponyms from large text corpora. In *Proceedings of the 14th conference on Computational linguistics-Volume 2*. Association for Computational Linguistics, 539–545.

[9] Wolfram Research, Inc. [n. d.]. SystemModeler, Version 5.0. ([n. d.]). Champaign, IL, 2017.

[10] Esther Kaufmann, Abraham Bernstein, and Renato Zumstein. 2006. Querix: A natural language interface to query ontologies based on clarification dialogs. In *5th International Semantic Web Conference (ISWC 2006)*. Springer, 980–981.

[11] Edward Loper and Steven Bird. 2002. NLTK: The Natural Language Toolkit. In *Proceedings of the ACL-02 Workshop on Effective Tools and Methodologies for Teaching Natural Language Processing and Computational Linguistics - Volume 1 (ETMTNLP '02)*. Association for Computational Linguistics, Stroudsburg, PA, USA, 63–70. https://doi.org/10.3115/1118108.1118117

[12] Nitin Madnani. 2007. Getting started on natural language processing with Python. *Crossroads* 13, 4 (2007), 5–5.

[13] Kavi Mahesh and Sergei Nirenburg. 1996. *Knowledge-based systems for natural language processing*. New Mexico State University, Computing Research Laboratory.

[14] Patrick Pantel and Marco Pennacchiotti. 2006. Espresso: Leveraging generic patterns for automatically harvesting semantic relations. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, 113–120.

[15] Ana-Maria Popescu, Alex Armanasu, Oren Etzioni, David Ko, and Alexander Yates. 2004. Modern natural language interfaces to databases: Composing statistical parsing with semantic tractability. In *Proceedings of the 20th international conference on Computational Linguistics*. Association for Computational Linguistics, 141.

[16] Garima Singh and Arun Solanki. 2016. An algorithm to transform natural language into SQL queries for relational databases. *Selforganizology* 3, 3 (2016), 110–126.

[17] Fabian M Suchanek, Georgiana Ifrim, and Gerhard Weikum. 2006. Combining linguistic and statistical analysis to extract relations from web documents. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 712–717.

[18] Peter Turney. 2001. Mining the web for synonyms: PMI-IR versus LSA on TOEFL. *Machine Learning: ECML 2001* (2001), 491–502.