9-2017

NATURAL LANGUAGE PROCESSING BASED GENERATOR OF TESTING INSTRUMENTS

Qianqian Wang

NATURAL LANGUAGE PROCESSING BASED GENERATOR OF TESTING INSTRUMENTS

A Project

Presented to the

Faculty of

California State University,

San Bernardino

In Partial Fulfillment

of the Requirements for the Degree

Master of Science

in

Computer Science

---

by

Qianqian Wang

September 2017

NATURAL LANGUAGE PROCESSING BASED GENERATOR OF TESTING INSTRUMENTS

A Project

Presented to the

Faculty of

California State University,

San Bernardino

---

by

Qianqian Wang

September 2017

Approved by:

Dr. Kerstin Voigt, Advisor, Computer Science and Engineering

Dr. Tong Lai Yu, Committee Member

Dr. George M. Georgiou, Committee Member

ABSTRACT

Natural Language Processing (NLP) is the field of study that focuses on the interactions between human language and computers. By “natural language” we mean a language that is used for everyday communication by humans. Different from programming languages, natural languages are hard to be defined with accurate rules. NLP is developing rapidly and it has been widely used in different industries. Technologies based on NLP are becoming increasingly widespread, for example, Siri or Alexa are intelligent personal assistants using NLP build in an algorithm to communicate with people. “Natural Language Processing Based Generator of Testing Instruments” is a stand-alone program that generates “plausible” multiple-choice selections by analyzing word sense disambiguation and calculating semantic similarity between two natural language entities. The core is Word Sense Disambiguation (WSD), WSD is identifying which sense of a word is used in a sentence when the word has multiple meanings. WSD is considered as an AI-hard problem. The project presents several algorithms to resolve WSD problem and compute semantic similarity, along with experimental results demonstrating their effectiveness.

ACKNOWLEDGEMENTS

I very gratefully acknowledge my advisor Dr. Kerstin Voigt for all her guidance, for meeting with me every week, and patiently listening to me for hours, for teaching me step by step how to program and test Python, for giving me suggestions, for imparting so much valuable knowledge and for all her encouragement and words of kindness. She is more than a professor for me, she is more like a close friend.

I am also very grateful to Dr. Tong Lai Yu and Dr. George M. Georgiou who agreed to serve on my master project committees. And Dr. Qingquan Sun who took part in my proposal and final presentation. Thank you for your very valuable time and suggestions

I would like to acknowledge the help of the Department of Computer Science at California State University, San Bernardino. Specifically, I would like to thank my graduate advisor Dr. Josephine Mendoza, who took the time from her busy schedule to gave me a lot of help, and thank all my class instructors for their help with infrastructure.

Finally, I would like to thank my parents and my fiancé, they gave me support all the time, I love them.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER ONE

STATEMENT OF PURPOSE

The objective of this study is to improve an existing software QAW.py, which is written by Dr. Voigt in Python programming language. This software is to generate multiple-choice selections from a study guide with terms and definitions or questions and answers. The old version of the program is functional, but several aspects can be improved, the biggest issue is that the relevance between multiple selections is not strong.

The input to the automated test generator is identical to the input to QAW or Easy Notes, namely a study guide or set of flashcards with questions associated with correct answers, or terms with matching definitions.

The automated system is to produce associations of questions with plausible answer alternatives fully automatically, without the intervention or help of a human judge. The core of accomplishing this goal is computer automated natural language processing, and in particular, the core challenge of natural language processing, automated word sense disambiguation.

CHAPTER TWO

INTRODUCTION

Background

Dr. Voigt developed a Python program, QAW.py, which takes as its input a text file containing study guide of the sort that US high school students are routinely asked to produced prior to their exams. Such study guides consist of long lists of either terms and their definitions, or questions and answers on some subjects' matter. Given study guide, the QAW software produces a comprehensive multiple choice test which can be taken online or in batch and paper-based mode. The original QAW program, the starting point of this project, is to be understood as a rapidly programmed, simple but functional prototype. As such it has multiple shortcomings, and this project aims at ensuring that the automatically generated test is appropriately challenging to the student who studies with the test.

The original program generates choices randomly, which allows test takers easy to use exclusion method to rule out the wrong answers and get the correct answer, rather than understanding what they learned. Several quiz generators online have the same function. For example, "Easy Notecards" [1] is a website, which helps users in reading novels by generating multiple-choice quizzes from notecards, but this website has the same problem as our old version, it generates choices randomly, alternative selections are not closely related resulting in a no challenging assessment. We focused on improving the

difficulty of questions by picking choices with smaller semantic distances, instead of getting them randomly.

Here is an example which is generated by QAW.py program. In this example, the question is obviously a name of a person, only answer 3 shows a definition of a person, the other three random definitions have nothing to do with the correct answer. In this situation, the test taker, even without having any knowledge of the subject of the test, will be able to eliminate items that are very obviously not plausible answers:

(1 of 5) Marie Currie/radioactivity:

[1] This was a style of realistic art that was being developed in the Soviet Union and it was becoming a dominant style in other various socialist countries. This was characterized by the glorified depiction of communist values, such as the emancipation of the proletariat in a realistic manner.

[2] This is a group of entities that share has been motivated by at least one common problem. They were working together so that they could achieve a common object. These are different from cooperatives and they are not really focused when it comes to economic benefit.

[3] She was a chemist who conducted the pioneering research on radioactivity. She was the first person who won a Nobel Prize. She even won twice in a row. Marie became a professor at the University of Paris.

[4] This was the Nazi propaganda term for annexing of Austria into Nazi Germany in Mach 1938. It was also known as the Anschluss Osterreichs.

This stands in contrast to the Anschluss movement when the Republic of German-Austria attempted the union with Germany.

Enter number: 3

... CORRECT!!!

SHE WAS A CHEMIST WHO CONDUCTED THE PIONEERING RESEARCH ON RADIOACTIVITY. SHE WAS THE FIRST PERSON WHO WON A NOBEL PRIZE. SHE EVEN WON TWICE IN A ROW. MARIE BECAME A PROFESSOR AT THE UNIVERSITY OF PARIS.

Natural Language Processing is used everywhere [2]. Natural Language Toolkit is a good tool would be used in Natural Language Processing. Since the original program was written in Python Language, this project would be generated in the same programming language. In this project, we used natural language processing to analyze the semantic distances between choices, thereby picking the other choices that are more plausible.

In the context of this project, “plausible choices” for the answer to a question are text selections (extracted verbatim from the study guide) whose meanings are or at least seem closely related to the posted question (or term to be defined). The test taker should not be able to easily rule out any of the potential answers based on their obvious lack of relatedness to the question. Instead, the test taker should have to apply true knowledge of the subject matter in order to distinguish the correct answer from a set of other seemly similar but incorrect ones.

Project Overview

Given: term and correct definition, choose “plausible” alternative definitions as follows:

1. Extract keywords:
    - Tokenize the text of term and correct definition.
    - Remove stop words.
    - Tag and extract nouns using filter_insignificant () function, then untag all words.
    - Change plural nouns to singular nouns.
2. Get correct senses of words:
    - Use the JIGSAW algorithm to extract correct synset for each word.
3. For all definitions of other terms in the study guide, we do the following:
    - Determine semantic similarity between keywords of a term and the keywords in all definitions.
4. Select alternative definitions, use the 3 to 4 with largest semantic similarity to keywords of correct definition.

CHAPTER THREE

NATURAL LANGUAGE TOOLKIT

Natural Language Toolkit, which is always be called as NLTK is a leading platform for building Python programs to work with human language data [3]. In our program, NLTK is the most significant package. It was developed by Steven Bird and Edward Loper in the Department of Computer and Information Science at the University of Pennsylvania [3]. It provides easy-to-use interfaces to several corpora and lexical resources [4]. It includes different libraries such as the one we used to resolve semantic similarity and word sense disambiguation problem, which is called Wordnet, and along with libraries for tokenization, stemming, tagging, and so on.

These are the packages from NLTK that we used in our project:

1. From nltk.tokenize import RegexpTokenizer: This package provides tokenizers to tokenize sentences into lists of words.
2. From nltk.corpus import stopwords: The nltk.corpus package defines a collection of corpus reader classes, which can be used to access the contents of a diverse set of corpora [5]. We used this package to remove stopwords in lists.
3. From nltk.tag import untag: Interface for tagging each token in a sentence with supplementary information, such as its part of speech [4].

4. From nltk.wsd import lesk: This package provides the Lesk algorithm to solve word sense disambiguation problem.
5. From nltk.corpus import wordnet: WordNet is the most frequently used package in our program. We imported several semantic similarity methods in our program from this package.
6. From nltk.stem import WordNetLemmatizer: Lemmatize using WordNet's built-in morphy function [4]. We used this method to change plural nouns to singular nouns.
7. From nltk.corpus import wordnet_ic: This package loads an information content file from the wordnet_ic corpus. For example, we used Brown Corpus as follows, brown_ic = wordnet_ic.ic('ic-brown.dat').

CHAPTER FOUR

TOKENIZATION

In this project, a study guide consists of terms and definitions, all terms and definitions appear in form of sentences. After using tokenizer, we received a list of words for each sentence. This process was called tokenization, and a list of words would be treated as tokens of a sentence.

Choose a Tokenizer

NLTK provides different types of tokenizers, in this project, we need to choose a tokenizer to split sentences into lists of individual words. Five types of tokenizers and their basic functions are shown as follows:

- Word_tokenize provides very basic word tokenization, it is an instance of the TreebankWordTokenizer class. It separates words using spaces and punctuation, and it keeps the punctuation.
- PunktWordTokenizer splits words on punctuation but keeps the punctuation with the word instead of creating separate tokens.
- WordPunctTokenizer is similar to PunktWordTokenizer, the only difference is that it splits all punctuation into separate tokens [2].
- RegexpTokenizer uses regular expressions to complete control over how to tokenize text. It can be used based on how we construct the regular expression.

- Whitespace Tokenizer uses RegexpTokenizer to tokenize on whitespace.

Here is an example to show the differences of five tokenizers. This is a sentence, “I’m a student.”, and after tokenized, the results are shown as follows:

- *word_tokenize: [' I ', " 'm ", ' a ', ' student ']*
- *PunktWordTokenizer: [' I ', " 'm ", ' a ', ' student. ']*
- *WordPunctTokenizer: [' I ', " ' ", ' m ', ' a ', ' student ', ‘. ']*
- *RegexpTokenizer: [" I'm ", ' a ', ' student ']*
- *Whitespace Tokenizer: [" I'm ", ' a ', ' student.']*

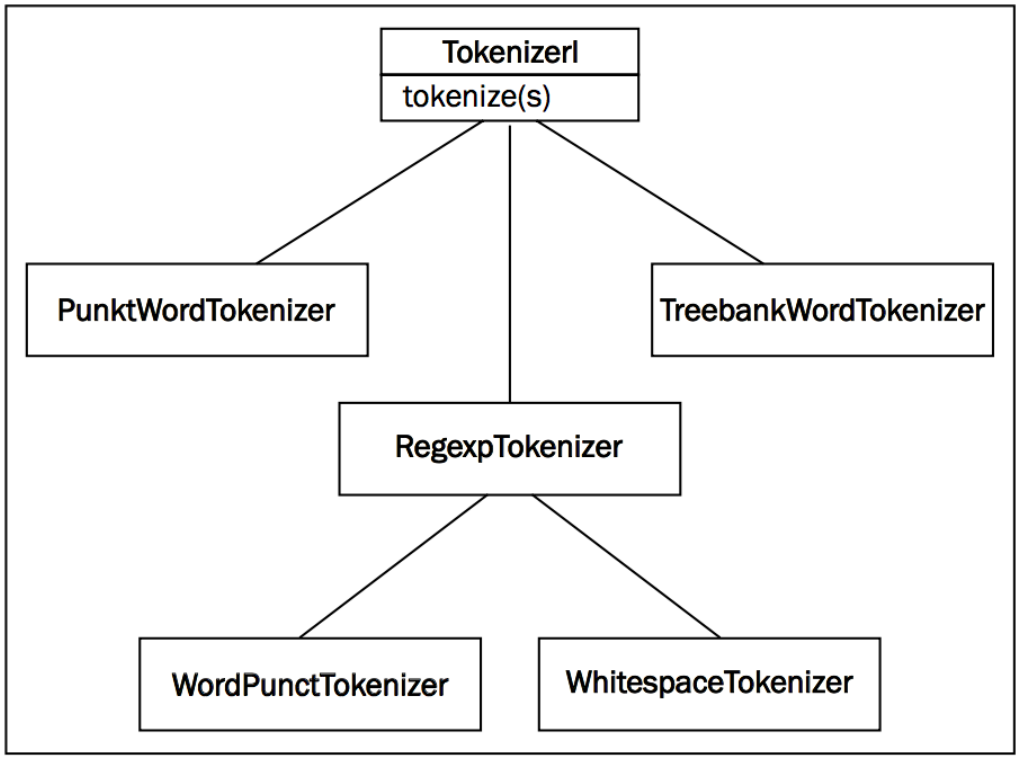

Figure 1. Different Types of Tokenizers

To choose an appropriate tokenizer, we need to decide how we want to tokenize a piece of text. In this project, what we need are lists of keywords, we don't need to keep punctuation, and we need to keep the essential words that can present the basic meaning of sentences. We chose RegexpTokenizer, which can match our exception. This tokenizer matches alphanumeric tokens plus single quotes so that we didn't split up contractions.

CHAPTER FIVE

FILTER STOP WORDS

After tokenizing all sentences, we got lists of words, the core objective we would like to do at last step was to calculate semantic similarity between each word in different terms and definitions, too many unrelated words in one list would influence the final result, so we only wanted to keep the keywords, and these keywords have special effects on the meaning of sentences. In this project, we only kept nouns for analyzing.

To get the keywords, we should remove stop words. Stop words are the “extremely common words” [6]. These words usually do not contain important information, even these words are removed, the main meaning will not be influenced. Such as “the” and “a”, they make no contribution to the meaning of a sentence.

In NLTK stopwords corpus, words () method provide lists of stop words for 14 different languages, in this project only English list were used. It is worth noting that before removing stop words, all words in lists need to be converted to lowercase, the reason is that uppercase words will not be considered as stop words, even they are in stop words list with lowercase form.

```
>>> from nltk.corpus import stopwords
>>> stopwords.words('english')
['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you', 'your', 'yours',
'yourself', 'yourselves', 'he', 'him', 'his', 'himself', 'she', 'her', 'hers',
'herself', 'it', 'its', 'itself', 'they', 'them', 'their', 'theirs', 'themselves',
'what', 'which', 'who', 'whom', 'this', 'that', 'these', 'those', 'am', 'is', 'are',
'was', 'were', 'be', 'been', 'being', 'have', 'has', 'had', 'having', 'do', 'does',
'did', 'doing', 'a', 'an', 'the', 'and', 'but', 'if', 'or', 'because', 'as', 'until',
'while', 'of', 'at', 'by', 'for', 'with', 'about', 'against', 'between', 'into',
'through', 'during', 'before', 'after', 'above', 'below', 'to', 'from', 'up', 'down',
'in', 'out', 'on', 'off', 'over', 'under', 'again', 'further', 'then', 'once', 'here',
'there', 'when', 'where', 'why', 'how', 'all', 'any', 'both', 'each', 'few', 'more',
'most', 'other', 'some', 'such', 'no', 'nor', 'not', 'only', 'own', 'same', 'so',
'than', 'too', 'very', 's', 't', 'can', 'will', 'just', 'don', 'should', 'now']
```

Figure 2. Stop Words

CHAPTER SIX

PART-OF-SPEECH TAGGING

The process of classifying words into their parts of speech and labeling them accordingly is known as part-of-speech tagging, or POS-tagging [6]. NLTK provides the method to attach a tag to each word. We used the tag () method, the input should be a list of words, and after using this method, program would return a list of tagged words as output. Figure2 shows tags we used in our project.

We used these tags to extract nouns only, the nouns are tagged as “NN”, “NNS”, “NNP”, or “NNPS.” After all nouns were extracted, we used the nltt.tag.untag() function to untag all sentences.

Number	Tag	Description
1.	CC	Coordinating conjunction
2.	CD	Cardinal number
3.	DT	Determiner
4.	EX	Existential *there*
5.	FW	Foreign word
6.	IN	Preposition or subordinating conjunction
7.	JJ	Adjective
8.	JJR	Adjective, comparative
9.	JJS	Adjective, superlative
10.	LS	List item marker
11.	MD	Modal
12.	NN	Noun, singular or mass
13.	NNS	Noun, plural
14.	NNP	Proper noun, singular
15.	NNPS	Proper noun, plural
16.	PDT	Predeterminer
17.	POS	Possessive ending
18.	PRP	Personal pronoun
19.	PRP$	Possessive pronoun
20.	RB	Adverb
21.	RBR	Adverb, comparative
22.	RBS	Adverb, superlative
23.	RP	Particle
24.	SYM	Symbol
25.	TO	*to*
26.	UH	Interjection
27.	VB	Verb, base form
28.	VBD	Verb, past tense
29.	VBG	Verb, gerund or present participle
30.	VBN	Verb, past participle
31.	VBP	Verb, non-3rd person singular present
32.	VBZ	Verb, 3rd person singular present
33.	WDT	Wh-determiner
34.	WP	Wh-pronoun
35.	WP$	Possessive wh-pronoun
36.	WRB	Wh-adverb

Figure 3. Part-of-Speech Tag Set

CHAPTER SEVEN

WORD SENSE DISAMBIGUATION

Background

After we got all keywords successfully, we used WordNet to get the synsets of each word in a list. The WordNet groups English words into sets of synonyms, these synonyms are called synsets. And then we chose one synset randomly. And then we used Wu-Palmer similarity model to calculate the semantic similarities between terms and definitions. And we also tested the results by using an online tool named “WS4J Demo” semantic similarity calculator [7] and analyzing the definitions of each word, we found that the result we got was far lower than we expected. Here is an example to show the semantic similarity we got and the semantic similarity “WS4J Demo” calculator got:

Term: ‘Nationalism’

Definition: ‘The belief that the interests of the nation as a whole are more important than regional interests or the interests of other countries.’

Keywords of term: ['nationalism']

Keywords of definition: ['belief', 'interests', 'nation', 'whole', 'countries']

Table 1. Semantic Similarities from Our Program and “WS4J Demo”

	From our Program	From “WS4J Demo” Calculator
Defs/terms	‘nationalism’	‘nationalism’
‘belief’	0.3077	0.8751
‘interests’	0.3750	0.6667
‘nation’	0.2667	0.4286
‘whole’	0.2667	0.6667
‘countries’	0.2667	0.4286

From table1, we can see that the result from our program is much lower than “WS4J Demo” Calculator. We analyzed semantic similarity between one word from term ‘nationalism’ and one word from its correct definition ‘belief’ to figure out the reason.

Synsets and definitions of ‘nationalism’:

0) *Synset ('patriotism.n.01'), 'love of country and willingness to sacrifice for it'*
1) *Synset ('nationalism.n.02'), 'the doctrine that your national culture and interests are superior to any other'*
2) *Synset ('nationalism.n.03'), 'the aspiration for national independence felt by people under foreign domination'*
3) *Synset ('nationalism.n.04'), 'the doctrine that nations should act independently (rather than collectively) to attain their goals'*

Synsets and definitions of 'belief':

0) *Synset ('belief.n.01'), 'any cognitive content held as true'*

1) *Synset ('impression.n.01'), 'a vague idea in which some confidence is placed'*

Semantic similarity between nationalism with synset 0 and belief with synset 1 is 0.3077:

```
In [69]: s1 = wordnet.synsets('nationalism')[0]

In [70]: s2 = wordnet.synsets('belief')[0]

In [71]: s1.wup_similarity(s2)
Out[71]: 0.3076923076923077
```

Figure 4. Similarity Between "Nationalism0" and "Belief0"

Semantic similarity between nationalism with Synset 0 and belief with Synset 1 is 0.8571:

```
In [72]: s1 = wordnet.synsets('nationalism')[1]

In [73]: s2 = wordnet.synsets('belief')[0]

In [74]: s1.wup_similarity(s2)
Out[74]: 0.8571428571428571
```

Figure 5. Similarity Between “Nationalism1” and “Belief0”

After analyzing all possibilities, we found that each word had several different senses, which sense would be used was due to the other words’ senses in the same sentence. Humans have a natural ability to judge whether a word is similar to another word. For instance, we all know that orange is a kind of fruit, and orange is similar to fruit, not similar to computer. But this is hard for machine language. Before analyzing word senses, in calculating the similarity between two words, the senses were chosen randomly, that's why the result was too different from we expected.

Accompanied by the generation of this problem, Word Sense Disambiguation (WSD) problem became the core part of this project. “WSD is one of the most important NLP tasks” [9]. To resolve WSD problem, the input would be a given sentence or a context. We should use an algorithm to find the most appropriate sense to a word in the particular sentence or context. And these senses could be used from WordNet package, they appear as synsets. So before calculating the semantic similarity, we need to use a WSD method to get correct senses.

WordNet, Synset, Hypernym, and Hyponym

WordNet

WordNet is an NLTK corpus reader. It is a lexical database developed at Princeton University with the attempt to model the lexical knowledge of a native speaker of English, using synsets, helps to find conceptual relationships between words such as hypernyms, hyponyms, synonyms, antonyms and so on. WordNet groups nouns, verbs, adjectives and adverbs into sets of synsets, each expressing a distinct concept. WordNet package contains several semantic similarity methods functions. We could use one of them to compute semantic similarity value.

WordNet has 117000 different synsets, each synset represents a definition of a word, and each synset would be unique in the WordNet. And “a synset contains one or more short sentences illustrating the use of the synset members” [10]. A polysemous word has different synsets to represent distinct meanings.

Synset

NLTK comes with a list of synset instances to look up words in WordNet. To look up any word in WordNet, we should use wordnet. synsets(‘word’) to get a list of Synsets.

Synset is a set of synonyms that share a common meaning. Each synset contains one or more lemmas, which represent a specific sense of a specific word [11]. Many words have only one synset, some have several.

Hypernym and Hyponym

One sense is a hyponym of another if the first sense is more specific, denoting a subclass of the other, conversely is a hypernym. For example, a car is a hyponym of a vehicle, a vehicle is a hypernym of a car, banana is a hyponym of fruit, then fruit is a hypernym of banana. Figure 6 shows a WordNet function to get the hypernyms of word car.

```
In [86]: from nltk.corpus import wordnet as wn
    ...:
    ...: car = wn.synset('car.n.01')
    ...:
    ...: hyp = lambda s:s.hypernyms()
    ...:
    ...: from pprint import pprint
    ...: pprint(car.tree(hyp))
    ...:
[Synset('car.n.01'),
 [Synset('motor_vehicle.n.01'),
  [Synset('self-propelled_vehicle.n.01'),
   [Synset('wheeled_vehicle.n.01'),
    [Synset('container.n.01'),
     [Synset('instrumentality.n.03'),
      [Synset('artifact.n.01'),
       [Synset('whole.n.02'),
        [Synset('object.n.01'),
         [Synset('physical_entity.n.01'), [Synset('entity.n.01')]]]]]]],
    [Synset('vehicle.n.01'),
     [Synset('conveyance.n.03'),
      [Synset('instrumentality.n.03'),
       [Synset('artifact.n.01'),
        [Synset('whole.n.02'),
         [Synset('object.n.01'),
          [Synset('physical_entity.n.01'), [Synset('entity.n.01')]]]]]]]]]]]
```

Figure 6. Hypernyms of The Word “Car”

If a word has different senses, then each sense will have different hypernyms. Table2 [12] shows the hypernyms for different senses of the word plant.

Table 2. Hypernyms Synsets of “Plant”

plant#1	plant#2	plant#3	plant#4
building complex #1	life form#1	contrivance#3	actor#1
structure #1	entity#1	scheme#1	performance#1
artifact#1		plan of action#1	entertainer#1
object#1		plan#1	person#1
entity#1		idea#1	life form#1
		content#5	entity#1
		congnition#1	
		psychological feature#1	

Lesk Algorithm

The Lesk algorithm is one of the most popular methods to solve WSD problem, it was introduced by Michael E. Lesk in 1986. The Lesk algorithm uses dictionary definitions to disambiguate a polysemous word in a sentence context. In order to extract definitions, Lesk adopted the Oxford Advanced Learner’s dictionary. The major objective of Lesk algorithm is to count the number of words

that are shared between two definitions [8] [13]. The more same words two definitions shared, the more similar the senses are.

Methodology

To get the correct sense of a target word, the Lesk algorithm allows the definition of the target word to compare with definitions of other words. A word is assigned to the sense whose definition shares the largest number of words in common with the definitions of the other words. Figure 7 [8] shows the graphic representation of the Lesk Algorithm.

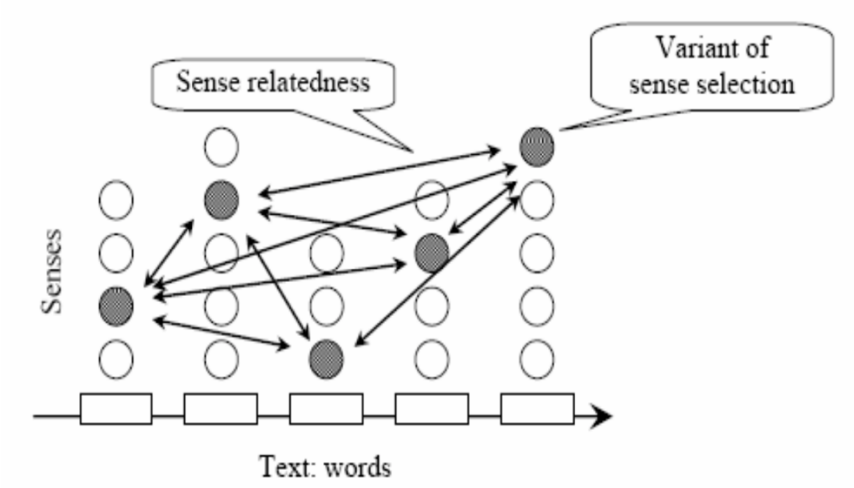

Figure 7. Graphic Representation of Lesk Algorithm

For example: In performing disambiguation for the word "book" in the sentence "I want to book a hotel with cheaper price in Las Vegas." The word "book" has 15 different synsets and definitions:

1) *(Synset('book.n.01'), a written work or composition that has been published (printed on pages bound together)'*
2) *(Synset('book.n.02'), physical objects consisting of a number of pages bound together'*
3) *(Synset('record.n.05'), a compilation of the known facts regarding something or someone'*
4) *(Synset('script.n.01'), a written version of a play or other dramatic composition; used in preparing for a performance'*
5) *(Synset('ledger.n.01'), a record in which commercial accounts are recorded'*
6) *(Synset('book.n.06'), collection of playing cards satisfying the rules of a card game'*
7) *(Synset('book.n.07'), a collection of rules or prescribed standards on the basis of which decisions are made'*
8) *(Synset('koran.n.01'), the sacred writings of Islam revealed by God to the prophet Muhammad during his life at Mecca and Medina'*
9) *(Synset('bible.n.01'), the sacred writings of the Christian religions'*
10) *(Synset('book.n.10'), a major division of a long written composition'*

11) *Synset('book.n.11'), a number of sheets (ticket or stamps etc.) bound together on one edge'*

12) *Synset('book.v.01'), engage for a performance'*

13) *Synset('reserve.v.04'), arrange for and reserve (something for someone else) in advance'*

14) *Synset('book.v.03'), record a charge in a police register'*

15) *Synset('book.v.04'), unregister in a hotel booker'*

```
In [15]: s = 'I want to book a hotel with cheaper price in Las Vegas'

In [16]: s = s.split()

In [17]: print lesk(s, 'book')
Synset('book.v.04')
```

Figure 8. Using The Lesk Algorithm to Get Synset of “Book”

The Lesk algorithm compared all senses of "book" with senses of other words in this sentence, to see which sense has the largest number of the common word are shared. So Synset 15 is declared to be the most appropriate sense when the word "book" appears in this sentence.

Example in Program

We applied the Lesk algorithm to our program. After getting keywords, we used the Lesk algorithm to get correct synsets of all words. Then calculated

semantic similarity between synsets of words in terms and synsets of words in definitions. We used their maximum, minimum, average value, the average of first three largest values and median to pick a typical value to represent a list of semantic similarity values. For example, there is a text file which is called sample1 with five terms and definitions as listed:

Q1: Cotton gin

A1: A machine that cleaned raw cotton. It automatically separated the cotton seeds from the fluffy fibers.

Q2: Second National Bank

A2: This was the second national bank, established by Congress in 1816. It was overseen by the federal government, and it was to oversee and regulate the smaller state banks.

Q3: Nullification

A3: This was the theory that individual states had the right to reject federal laws, for example, laws that required the paying of tariffs on foreign goods. Southern states declared such laws null and void and threatened to leave the union if they were forced to pay such tariffs.

Q4: Lowell System

A4: A system used by the textile industry in Lowell, Massachusetts. Using farm girls as workers, they were the first ones to have an innovative way to weave cloth from a thread.

Q5: Sectionalism

A5: This is the belief that one own section, or region, of the country, is more important than the whole.

We used the Lesk algorithm to get lists of synsets for terms and definitions.

For example, this is a list of synsets for A1:

[Synset('machine.n.05'), Synset('scavenge.v.04'), Synset('raw.s.02'), Synset('cotton.n.01'), Synset('disjointed.s.03'), Synset('cotton.n.01'), Synset('seed.v.08'), Synset('downy.s.01'), Synset('character.n.03')].

Then we computed the semantic distance using these synsets. For example, this is the list of semantic similarity values between Q1 and A1:

[0.3333333333333333, 0.3333333333333333, 0.3333333333333333, 0.2857142857142857, 0.2857142857142857, 0.2222222222222222, 0.2222222222222222, 0.2, 0.18181818181818182, 0.14285714285714285, 0.14285714285714285, 0.14285714285714285, 0.13333333333333333].

And in this example, we picked the first three greatest values out from the list and calculated their average numbers. Table3 shows the result of first three greatest semantic similarity values between each term and definition in sample1.

Table 3. Average of Three Max Semantic Similarity Values for Sample1

Defs \Terms	Q1	Q2	Q3	Q4	Q5
A1	0.3333	0.3333	0.2543	0.2353	0.2719
A2	0.4421	0.7917	0.2103	0.4339	0.2593
A3	0.3333	0.5556	0.3497	0.2081	0.4955
A4	0.5545	0.6565	0.2013	0.7143	0.2143
A5	0.3651	0.3556	0.2679	0.3824	0.3316

Limitation

Using the Lesk algorithm to pick synsets is much better than getting synsets randomly. To make sure all words get accurate senses, we analyzed all synsets we got from the Lesk algorithm. But the result showed that only lower than 50% of words got their correct senses in sentences.

For example: in Q1: [‘cotton’,’ gin’], word cotton has five senses, and word gin has six senses, the senses we expected to get was synset1 of word cotton and synset3 of word gin. But after using the Lesk algorithm, especially for the word gin, the definition of its synset3 had most numbers of the common word with word cotton, but the Lesk algorithm picked synset1, which made no sense in this "cotton gin" phrasal.

Cotton:

1) *Synset('cotton.n.01'), soft silky fibers from cotton plants in their raw state'*
2) *Synset('cotton.n.02'), fabric woven from cotton fibers'*

3) *Synset('cotton.n.03'), erect bushy mallow plant or small tree bearing bolls containing seeds with many long hairy fibers'*
4) *Synset('cotton.n.04'), thread made of cotton fibers'*
5) *Synset('cotton.v.01'), take a liking to'*

```
In [6]: sent = ['cotton','gin']

In [7]: print lesk(sent,'cotton')
Synset('cotton.n.04')
```

Figure 9. Using The Lesk Algorithm to Get Synset of “Cotton”

Gin:

1) *Synset('gin.n.01'), strong liquor flavored with juniper berries'*
2) *Synset('snare.n.05'), a trap for birds or small mammals; often has a slip noose'*
3) *Synset('cotton_gin.n.01'), a machine that separates the seeds from raw cotton fibers'*
4) *Synset('gin.n.04'), a form of rummy in which a player can go out if the cards remaining in their hand total less than 10 points'*
5) *Synset('gin.v.01'), separate the seeds from (cotton) with a cotton gin'*
6) *Synset('gin.v.02'), trap with a snare'*

```
In [10]: print lesk(sent,'gin')
Synset('gin.v.01')
```

Figure 10. Using The Lesk Algorithm to Get Synset of “Gin”

The major limitation of the Lesk algorithm is the low accuracy during process and the type of a sample is limited by this algorithm. Since our input would be randomly chosen by test taker, we could not make sure each word in the test file would share same words in their definitions, even they appeared in one sentence to represent a complete meaning. And a small sample, and a big number of fine senses in WordNet, many of which are not that distinguishable from each other. Only when the words in one context are most likely related to each other, the Lesk algorithm could perform a better disambiguation.

Resnik Algorithm

The Resnik algorithm is also one of the most useful methods to solve WSD problem. It was designed by Philip Resnik in 1995. Different from the Lesk algorithm, this algorithm would not compare the same words between two senses, it would pick the correct sense by analyzing the most informative subsumer of two senses, if the two senses contain a more informative subsumer, then these two senses would be more related to each other. “Their most

informative subsumer provides information about which sense of each word is the relevant one" [19].

The most informative subsumer also called most specific subsumer or least common subsumer is the most specific common ancestor of two concepts found in a given ontology. It represents the commonality of the pair of concepts. In a WordNet hypernym hierarchy, the most informative subsumer is the deepest node in each path that covers all children. For example, in Figure 11, "automobile" is the ancestor of "car", while "vehicle" is an ancestor of "car". "Vehicle" is also an ancestor of "boat". In this case, the most informative subsumer of both the "boat" and the "car" is "vehicle", since it's the most specific concept which is an ancestor of both the "boat" and the "car".

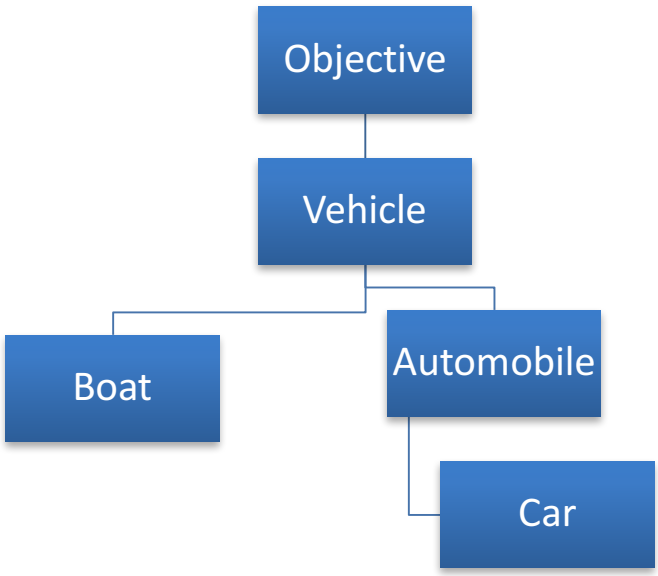

Figure 11. Fragment of The WordNet Hypernym Hierarchy.

This algorithm does not have a direct function can be used from NLTK libraries as the Lesk algorithm. So we need to design a function ourselves based on its algorithm.

Problem Statement

The problem is stated as follows. Given a set of words W = {w_1, w_2, …, w_n}, with each word w_i having an associated set Si = {s_{i1}, s_{i2}, …, s_{im}} of possible senses. Assume that there exists some set W' $\subseteq \cup$ Si representing the set of word senses that an ideal human judge would conclude belong to the group of senses corresponding to the word grouping W [20].

The Resnik algorithm is to define a function that takes w_i, s_{im} and set W as input and create a formula to analyze the relationship between wi in W and s_{im} in S_i. Then compute a value between 0 and 1 to represent the confidence with which one can state that sense s_{im} belongs to W'.

For example, we picked a group of words from Brown cluster, this word group contained lawyer, doctor, nurse. We treated this group as W. By getting senses of these three words in WordNet, we found that word "lawyer" contained only one sense: Synset('lawyer.n.01'), a professional person authorized to practice law; conducts lawsuits or gives legal advice'. But both the word "doctor" and the word "nurse" are polysemous.

Senses of "doctor":

1) *Synset('doctor.n.01'), a licensed medical practitioner'*
2) *Synset('doctor_of_the_church.n.01'), (Roman Catholic Church) a title conferred on 33 saints who distinguished themselves through the orthodoxy of their theological teaching'*
3) *Synset('doctor.n.03'), children take the roles of physician or patient or nurse and pretend they are at the physician's office"*
4) *Synset('doctor.n.04'), a person who holds Ph.D. degree (or the equivalent) from an academic institution'*
5) *Synset('sophisticate.v.03'), alter and make impure, as with the intention to deceive'*
6) *Synset('doctor.v.02'), give medical treatment to'*
7) *Synset('repair.v.01'), restore by replacing a part or putting together what is torn or broken'*

Senses of "nurse":

1) *Synset('nurse.n.01'), one skilled in caring for young children or the sick (usually under the supervision of a physician)'*
2) *Synset('nanny.n.01'), a woman who is the custodian of children'*
3) *Synset('nurse.v.01'), try to cure by special care of treatment, of an illness or injury'*
4) *Synset('harbor.v.01'), maintain (a theory, thoughts, or feelings)'*
5) *Synset('nurse.v.03'), serve as a nurse; care for sick or handicapped people'*

6) *Synset('nurse.v.04'), treat carefully'*

7) *Synset('breastfeed.v.01'), give suck to'*

In this group, all three words are kinds of professional people. Especially for the word “doctor” and the word “nurse”, they are professional people working in the health professions. After using this algorithm, we should assign a high value to the unique sense of “doctor” as Synset('doctor.n.01'), a licensed medical practitioner. And it should also assign a high value to the sense of “nurse” as Synset('nurse.n.01'), one skilled in caring for young children or the sick (usually under the supervision of a physician). A low value should be assigned to other senses.

Resnik Similarity

The Resnik similarity is the core of this disambiguation algorithm, it evaluates semantic similarity in an IS-A taxonomy [14], based on the information Content (IC) of the least Common Subsumer.

Given two words c_1 and c_2, the Resnik similarity is calculated as:

$$\mathrm{sim}(c_1, c_2) = \max_{c \in S(c_1, c_2)} [-\log \mathrm{p}(c)] \quad (1)$$

For words c_1 and c_2, they have some different hypernyms as their subsumers. For example, in Figure 12 [15], nurse1 and doctor1 has a lot of different subsumers: health professional, professional, adult, person. Concept c is a common subsumer of c_1 and c_2, and p(c) is the probability of c occurs in a

corpus. This probability is between 0 and 1. Let s_1 and s_2 be common subsumers of c_1 and c_2. If s_1 is-a s_2, then $p(s_1) \leq p(s_2)$.

A frequent subsumer has a high value of probability, and an infrequent subsumer has a low value of probability. And a subsumer that occurs rarely is more special, it means that this subsumer carries more information. In Figure 12, the person is more abstract than health professional to doctor1 and nurse1, so health professional is a more informative subsumer to doctor1 and nurse1.

Words/concepts that appear frequently and in the context of many diverse subject domains are not typical/indicative of a specific subject domain. When such a common word is used in a communication, it is difficult to tell which specific topic this word is meant to allude (because there are so many possible contexts in which the word could be used). Thus high frequency words, that is, words with high p(c), carry little informational content ("there is nothing special about them").

On the other hand, low frequency words, which are those that are used only rarely and in some very specific contexts, are very indicative of the topic being communicated. That is, low probability p(c) is synonymous with high informational content.

Probabilities are numbers between 0.0 and 1.0; logarithms of such numbers are negative, and numbers closer to 0.0 have smaller ("more negative") values than number closer to 1.0. By maximizing over the negative log p(c), this will reverse. The result is that larger values are now indicative of concepts c with

larger information content. Thus, the subsuming concept that is least common (of smallest probability) is the one that carries the greatest amount of information.

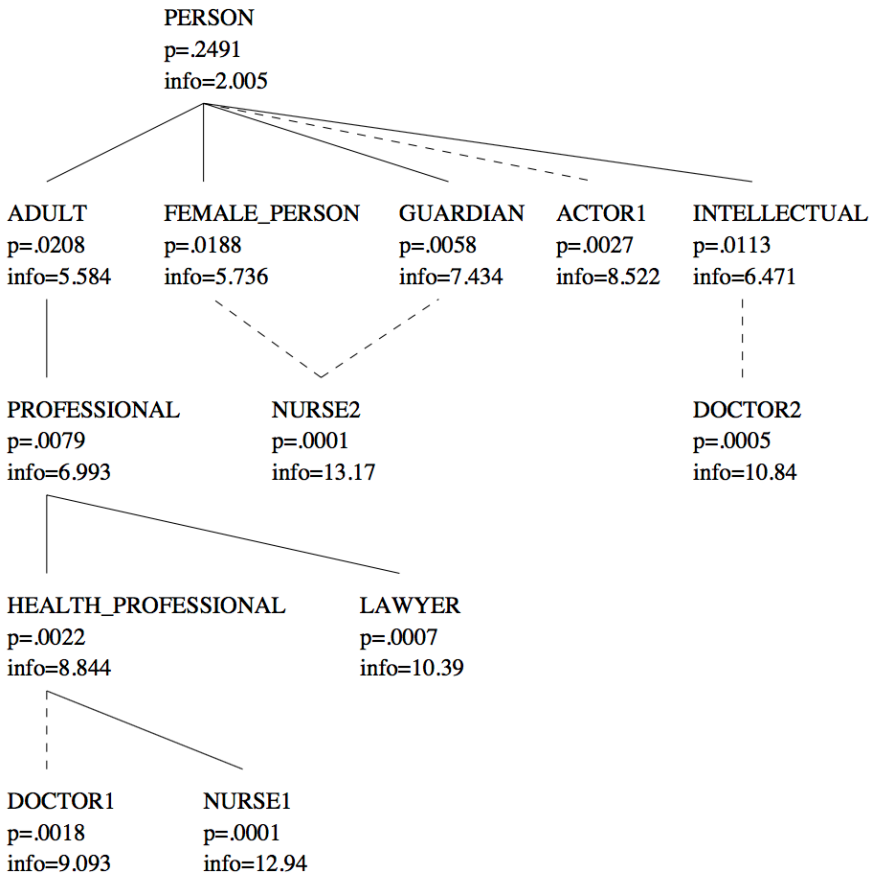

Figure 12. Fragment of The WordNet Taxonomy.

Probability p(c) estimates are derived from a corpus by computing [14]:

$$\text{freq}(c) = \sum_{n \in \text{words}(c)} \text{count}(n) \quad (2)$$

Where words(c) is the set of nouns having a sense subsumed by concept c. N is the total number of nouns observed, probabilities are then computed simply as relative frequency [14]:

$$\hat{\text{p}}(c) = \frac{\text{freq}(c)}{N} \quad (3)$$

Table 4 shows examples of semantic similarity computed by the Resnik similarity for several word pairs. From Figure 12, we can see that word “doctor” with sense 1 as a medical person has hypernyms: health professional, professional, adult, and person. The doctor1 is similar to the word “lawyer”. They have the least common subsumer as a professional person. But it is even similar to nurse since both of them are the professional person working specifically within the health professions. Moreover, the Resnik similarity is a more specialized concept than association or relatedness. As we can see in Table 4, even doctor and medicine are highly associated, but the Resnik similarity would not judge them to be particularly similar.

Table 4. Computation of Similarity for Several Pairs of Words

Word1(c1)	Word2(c2)	Sim (c1,c2)	Most informative subsumer
doctor1(medical)	nurse1(medical)	7.930359	health_professional
doctor1(medical)	lawyer	6.394069	professional

doctor1(medical)	adult	4.910252	adult
doctor1(medical)	person	2.333545	person
doctor1(medical)	medicine	0.000000	entity

Disambiguation Algorithm

When two polysemous words are similar, their most informative subsumer provides information about which sense of each word is the relevant one [15]. Figure 13 is the disambiguation algorithm for noun groupings. The key idea is to consider the nouns in a word group pairwise [14] [15]. For word group pairwise w_i and w_j, the algorithm goes across all possible senses of both to get the most informative subsumer of them. Word w_i can potentially get support from each of its k senses. And a sense supports w_i if the corresponding synset has most informative subsumer among its ancestors.

Here is an example, where W = {w_1, w_2, w_3}, for w_1 = doctor, w_2 = nurse, w_3 = lawyer. The word “doctor” and the word “nurse” are polysemous, here we considered two senses of the word doctor, we assumed doctor1= medical person, doctor2 = a person who holds a Ph.D. degree. We also picked two senses of the word “nurse”, we assumed nurse1 = medical person, nurse2 = nanny. In this example, we want to get values of doctor1 and doctor2.

- *Pairwise1: for w_1 = doctor and w_2 = nurse, the most subsumer $c_{1,2}$ = Health professional, and information content $v_{1,2}$ = 7.930359. So the support for doctor1 and nurse1 is incremented by 7.930359. And also, $c_{1,2}$*

is not an ancestor of doctor2 and nurse2, so neither doctor2 nor nurse2 receives increment for their support.

- *Pairwise2: for w_1 = doctor and w_3 = lawyer, the most subsumer $c_{1,3}$ = professional, and the information content $v_{1,3}$ = 6.394069. The support for doctor1 and lawyer is incremented by 6.394069. And also, doctor2 cannot receive any increment for support as $c_{2,3}$ is not an ancestor of doctor2.*
- *Pairwise3: for w_2 = nurse and w_3 = lawyer, the most subsumer $c_{2,3}$ = professional, and the information content $v_{2,3}$ = 6.394069. The support for nurse1 and lawyer is incremented by 6.394069. And also, nurse2 cannot receive any increment for support as $c_{2,3}$ is not an ancestor of nurse2.*

Doctor1 participated comparisons in pairwise1 and pairwise2, received support 7.930359 + 6.394069 out of a possible of 7.930359 + 6.394069, support / normalize = (7.930359 + 6.394069) / (7.930359 + 6.394069) = 1.000000, so the value assigned to doctor1 is 1.000000. doctor2 received 0 out of 7.930359 + 6.394069, support / normalize =0 / (7.930359 + 6.394069) = 0.000000, so the value assigned to doctor2 is 0.000000.

Algorithm (Resnik, 1998a). Given $W = \{w_1, \ldots, w_n\}$, a set of nouns:

```
for i and j = 1 to n, with i < j
{
   v_{i,j} = wsim(w_i, w_j)
   c_{i,j} = the most informative subsumer for w_i and w_j

   for k = 1 to num_senses(w_i)
        if c_{i,j} is an ancestor of sense_{i,k}
             increment support[i, k] by v_{i,j}

   for k' = 1 to num_senses(w_j)
        if c_{i,j} is an ancestor of sense_{j,k'}
             increment support[j, k'] by v_{i,j}

   increment normalization[i] by v_{i,j}
   increment normalization[j] by v_{i,j}
}

for i = 1 to n
   for k = 1 to num_senses(w_i)
   {
        if (normalization[i] > 0.0)
             φ_{i,k} = support[i, k] / normalization[i]
        else
             φ_{i,k} = 1 / num_senses(w_i)
   }
```

Figure 13. Disambiguation Algorithm for Noun Groupings.

Examples

We used two groups of words from Brown clusters as inputs. The first group was W1, which contained the words “tie”, “jacket”, and “suit”. The second group was W2, which contained the words “doctor”, “lawyer”, and “nurse”.

After using the Resnik algorithm, we got their senses and assigned values to different senses. From the examples, we could see the function we defined by the Resnik algorithm assigned different values to different senses. And in example2, this algorithm assigned value of 1.0 to the unique sense of the word “doctor” and the sense of the word “nurse” as we predicted before.

Example1:

Input: Given W1 = ['tie', 'jacket', 'suit']

Output:

jacket:

___ 1.000000 Synset('jacket.n.01'): a short coat

___ 0.000000 Synset('jacket.n.02'): an outer wrapping or casing

___ 0.000000 Synset('crown.n.11'): (dentistry) dental appliance consisting of an artificial crown for a broken or decayed tooth

___ 0.000000 Synset('jacket.n.04'): the outer skin of a potato

___ 0.000000 Synset('jacket.n.05'): the tough metal shell casing for certain kinds of ammunition

tie:

___ 1.000000 Synset('necktie.n.01'): neckwear consisting of a long narrow piece of material worn (mostly by men) under a collar and tied in knot at the front

___ 0.000000 Synset('affiliation.n.01'): a social or business relationship

___ 0.000000 Synset('tie.n.03'): equality of score in a contest

___ 0.000000 Synset('tie.n.04'): a horizontal beam used to prevent two other structural members from spreading apart or separating

___ 0.000000 Synset('link.n.02'): a fastener that serves to join or connect

___ 0.000000 Synset('draw.n.03'): the finish of a contest in which the score is tied and the winner is undecided

___ 0.000000 Synset('tie.n.07'): (music) a slur over two notes of the same pitch; indicates that the note is to be sustained for their combined time value

___ 0.000000 Synset('tie.n.08'): one of the cross braces that support the rails on a railway track

___ 0.000000 Synset('tie.n.09'): a cord (or string or ribbon or wire etc.) with which something is tied

suit:

___ 1.000000 Synset('suit.n.01'): a set of garments (usually including a jacket and trousers or skirt) for outerwear all of the same fabric and color

___ 0.000000 Synset('lawsuit.n.01'): a comprehensive term for any proceeding in a court of law whereby an individual seeks a legal remedy

___ 0.000000 Synset('suit.n.03'): (slang) a businessman dressed in a business suit

___ 0.000000 Synset('courtship.n.01'): a man's courting of a woman; seeking the affections of a woman (usually with the hope of marriage)

___ 0.000000 Synset('suit.n.05'): a petition or appeal made to a person of superior status or rank

___ 0.000000 Synset('suit.n.06'): playing card in any of four sets of 13 cards in a pack; each set has its own symbol and color

Example 2:

Input: Given W2: ['doctor', 'lawyer', 'nurse']

Output:

nurse:

___ 1.000000 Synset('nurse.n.01'): one skilled in caring for young children or the sick (usually under the supervision of a physician)

___ 0.000000 Synset('nanny.n.01'): a woman who is the custodian of children

lawyer:

___ 0.446375 Synset('lawyer.n.01'): a professional person authorized to practice law; conducts lawsuits or gives legal advice

doctor:

___ 1.000000 Synset('doctor.n.01'): a licensed medical practitioner

___ 0.000000 Synset('doctor_of_the_church.n.01'): (Roman Catholic Church) a title conferred on 33 saints who distinguished themselves through the orthodoxy of their theological teaching

___ 0.000000 Synset('doctor.n.03'): children take the roles of physician or patient or nurse and pretend they are at the physician's office

___ *0.000000 Synset('doctor.n.04'): a person who holds Ph.D. degree (or the equivalent) from an academic institution*

JIGSAW Algorithm

The JIGSAW algorithm was improved based on the Resnik algorithm. The JIGSAW algorithm takes a list of words as input, and then returns a list of WordNet synsets $S = \{s_1, s_2, \ldots, s_k\}$ in which each element s_i is obtained by disambiguating the target word w_i based on the information obtained from WordNet about a few immediately surrounding words [16].

Methodology

The JIGSAW algorithm differs from the Resnik algorithm in three parts as follows:

- The using of Gaussian distribution, which takes into account the distance between the words in the list to be disambiguated. In the Resnik similarity, we need to compare every pair of words in a list, but for a long sentence, or even a text with more words, not every pair of words are highly associated. If the position of one word is far away from another word, their senses may have low probability of influencing each other. We could see from Figure 14, for each pair of words w_i and w_j, similarity = sim (w_i and w_j) $\times$ G(pos(w_i), pos(w_j)). The closer their positions are, the larger their Gaussian factor is, the larger their similarity is. Therefore, we put positions of all words into a Gaussian function to choose the nearest most informative subsumer in all pairs of synsets.

- Setting depth1 and depth2 to limit the search for most informative subsumer to k ancestors. Although the limitation of depths of searching for the ancestor may guarantees that "too abstract" most informative subsumers will be ignored. We ignored this difference in our program. Because our input was irregular, it leaded to the result that we could not set values to depths stable. If the values of depths are inappropriate, in other words, if one of the depths is too small, it may cause the most specific subsumer of two words would not be found.
- Making use of factor R, which gives more importance to the synsets that are more common than others, according to the frequency score in WordNet [16] [17]. For word w_i, s_{ik} is one sense of w_i, the factor R(k) that takes into account the rank of s_{ik} in WordNet. R(k) is computed as:

$$R(k) = 1 - 0.8 * \frac{k}{n-1} \quad (4)$$

Where n is the cardinality of the sense inventory S_i for w_i and k is the rank of s_{ik} in S_i, starting from 0 [16]. And the JIGSAW algorithm also assigned two parameters α and β to control expression 4, where α controls the normalized support, and β controls R(k). In this algorithm, α= 0.7 and β = 0.3.

Algorithm 1 The procedure for disambiguating nouns derived from the algorithm by Resnik

procedure $JIGSAW_{nouns}(W, depth1, depth2)$ ▷ finds the proper synset for each polysemous noun in the set $W = \{w_1, w_2, \ldots, w_n\}$, $depth1$ and $depth2$ are used in the computation of MSS
  for all $w_i, w_j \in W$ **do**
    if $i < j$ **then**
      $sim \leftarrow sim(w_i, w_j, depth1) * G(pos(w_i), pos(w_j))$ ▷ $G(x, y)$ is a Gaussian function which takes into account the difference between the positions of w_i and w_j
      $MSS_{ij} \leftarrow MSS(w_i, w_j, depth2)$ ▷ MSS_{ij} is the most specific subsumer between w_i and w_j, search for MSS restricted to $depth2$ ancestors
      for all $s_{ik} \in S_i$ **do**
        if is-ancestor(MSS_{ij},s_{ik}) **then** ▷ if MSS_{ij} is an ancestor of s_{ik}
          $sup_{ik} \leftarrow sup_{ik} + sim$
        end if
      end for
      for all $s_{jh} \in S_j$ **do**
        if is-ancestor(MSS_{ij},s_{jh}) **then**
          $sup_{jh} \leftarrow sup_{jh} + sim$
        end if
      end for
      $norm_i \leftarrow norm_i + sim$
      $norm_j \leftarrow norm_j + sim$
    end if
  end for
  for all $w_i \in W$ **do**
    for all $s_{ik} \in S_i$ **do**
      if $norm_i > 0$ **then**
        $\varphi(i, k) \leftarrow \alpha * sup_{ik}/norm_i + \beta * R(k)$
      else
        $\varphi(i, k) \leftarrow \alpha/|S_i| + \beta * R(k)$
      end if
    end for
  end for
end procedure

Figure 14. The JIGSAW Algorithm for Noun Groupings.

Gaussian Distribution

The normal (or Gaussian) distribution is a very common continuous probability distribution [20]. The probability density of the normal distribution is:

$$f(x \mid \mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \quad (5)$$

In this algorithm, we used Gaussian distribution to take into account of the difference between the positions of w_i and w_j. To build an appropriate Gaussian function, we imported Numpy package to insert formula 5, and set μ = 0.0, σ = 0.7, then we got the Gaussian function [19] as Figure 15. We used expression 6 to evenly distribute positions of words to the Gaussian distribution we got in Figure 15 from 0.0 to $3 \times 0.7 = 2.1$. In expression 6, “dist” represents the distance between two positions, and “numwords” represents the total numbers of words.

$$Gauss = \mu + \frac{dist * 3 * \sigma}{numwords} \quad (6)$$

And finally, we assigned G(pos(w_i), pos(w_j)) to the calculation of the Resnik similarity between w_i and w_j.

```
In [63]: gauss = np.linspace(-4,4,1000)

In [64]: plt.scatter(gauss, allgauss)
Out[64]: <matplotlib.collections.PathCollection at 0x11de66c50>
```

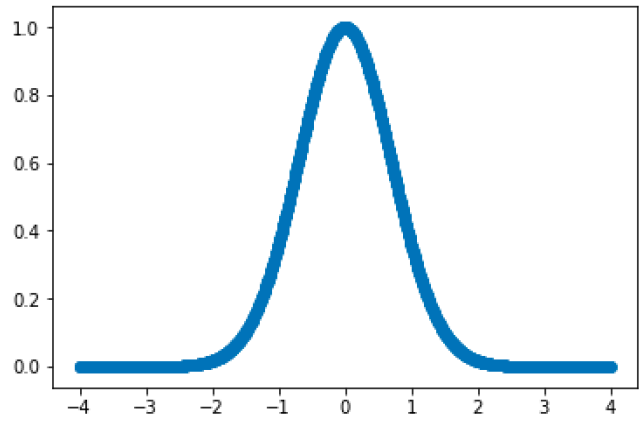

Figure 15. Gaussian Distribution.

Examples

We picked A2 from sample1 as an example, and got a list of keywords as follows: ['bank', 'government', 'state']. And then we compared results from both the JIGSAW algorithm and the Resnik algorithm.

Comparing the Lesk algorithm to the Resnik algorithm, the JIGSAW algorithm got the highest accuracy. Although it was not perfect, it was good enough to be applied to our program.

Table 5. “State”: The Resnik Algorithm and The JIGSAW Algorithm

Synset	Definition	Resnik algorithm	Jigsaw algorithm
Synset('state.n.01')	the territory occupied by one of the constituent administrative districts of a nation.	0.000000	0.300000
Synset('state.n.02')	the way something is with respect to its main attributes	0.000000	0.270000
Synset('state.n.03')	the group of people comprising the government of a sovereign state	0.117468	0.940000
Synset('state.n.04')	a politically organized body of people under a single government	0.117468	0.910000
Synset('state of _matter.n.01')	(chemistry) the three traditional states of matter are solids (fixed shape and volume) and liquids (fixed volume and shaped by the container) and gases (filling the container)	0.000000	0.180000
Synset('state.n.06')	a state of depression or agitation	0.000000	0.150000
Synset('country.n.02')	the territory occupied by a nation	0.000000	0.120000
Synset('department_of _state.n.01')	the federal department in the United States that sets and maintains foreign policies	0.117468	0.790000

Table 6. “Bank”: The Resnik Algorithm and The JIGSAW Algorithm

Synset	Definition	The Resnik algorithm	The Jigsaw algorithm
Synset('bank.n.01'):	sloping land (especially the slope beside a body of water)	0.000000	0.300000
Synset('depository_financial_institution.n.01')	a financial institution that accepts deposits and channels the money into lending activities	0.299483	1.676000
Synset('bank.n.03')	a long ridge or pile	0.000000	0.252000
Synset('bank.n.04')	an arrangement of similar objects in a row or in tiers	1.250000	0.228000
Synset('bank.n.05')	a supply or stock held in reserve for future use (especially in emergencies)	0.000000	0.204000
Synset('bank.n.06')	the funds held by a gambling house or the dealer in some gambling games	0.000000	0.180000
Synset('bank.n.07')	a slope in the turn of a road or track; the outside is higher than the inside in order to reduce the effects of centrifugal force	0.000000	0.156000
Synset('savings_bank.n.02')	a container (usually with a slot in the top) for keeping money at home	0.000000	0.132000
Synset('bank.n.09'	a building in which the business of banking transacted	0.000000	0.108000
Synset('bank.n.10')	a flight maneuver; aircraft tips laterally about its longitudinal axis (especially in turning)	0.000000	0.084000

Table 7. “Government”: The Resnik Algorithm and The JIGSAW Algorithm

Synset	Definition	The Resnik algorithm	The Jigsaw algorithm
Synset('government.n.01')	the organization that is the governing authority of a political unit	0.117468	1.000000
Synset('government.n.02')	the act of governing; exercising authority	0.000000	0.240000
Synset('government.n.03')	(government) the system or form by which a community or other political unit is governed	0.000000	0.180000
Synset('politics.n.02')	the study of government of states and other political units	0.000000	0.120000

CHAPTER EIGHT

SEMANTIC SIMILARITY

Using the JIGSAW algorithm, we got a list of synsets with the highest values. Then we used Wu-Palmer Similarity to calculate the semantic similarity between terms and definitions. Wu-Palmer [21] similarity returns a score denoting how similar two-word senses are, based on the depth of the two senses in the taxonomy and that of their Least Common Subsumer [18].

$$sim\ wup(c1, c2) = 2 * \frac{\text{depth(lcs)}}{depth(c1)+depth(c2)} \quad (7)$$

There are several models for computing semantic similarity, we chose Wu-Palmer method for two reasons: Firstly, Wu-Palmer [21] computes similarity by using words' senses, the input should be a pair of synsets, so the output from the JIGSAW algorithm could be used directly. And the second reason is that the score returns from Wu-Palmer Similarity is between 0 and 1, it is easy for the future comparison and analysis.

We created sample2 with ten terms and ten definitions. All terms and definitions were picked from a high school history book. We picked the first five terms and definitions from Spanish-American War chapter and the last five terms and definitions from Vietnam War chapter.

Sample2:

Q1: Why was the United States interested in expanding its territories in the late 1800s?

A1: The United States wanted new markets and military advantages. Some Americans wanted to spread their Christian faith to other countries. In addition, many Americans believed that expanding into the Pacific was their manifest destiny.

Q2: What did the United States gain and lose in the Spanish-American War?

A2: The United States gained control of Puerto Rico, Guam, and the Philippines (for $20 million). The United States also became an imperialist nation with more bases for trade and for resupplying its navy. The war cost the United States about $250 million, and about 5,400 soldiers lost their lives in the war.

Q3: Why did the United States support the Panamanian revolt against Colombian rule?

A3: Colombia's Senate refused to ratify a treaty that would have allowed the United States permanent use of the land needed to build a canal through the Isthmus of Panama. When Panamanian rebels revolted against Colombia, the United States supported them in hopes that after their victory, they would agree to the U.S. canal plan. When Panama declared independence, the United States swiftly recognized the new country, and a new canal treaty was soon signed.

Q4: Why the United States withdrew its troops from Mexico in 1917?

A4: Possible answers include any two of the follows: Having the troops in Mexico increased the risk of war between the United States and Mexico; U.S. soldiers made no progress in capturing Pancho Villa, and Wilson was compelled to pay attention to developments in the early years of World War I.

Q5: Why did the United States support the Panamanian rebellion and recognize the new Republic of Panama?

A5: The United States wanted a friendly government in Panama which would support negotiations to allow the United States to build a canal connecting the Atlantic and Pacific Oceans.

Q6: Why did President Kennedy initially send advisors and aid to Vietnam?

A6: Kennedy was a firm believer in the domino theory, and after the two Cold War disasters that began Kennedy's presidency, he hoped that aiding South Vietnam would be a sign of continued U.S. strength and resolve.

Q7: Why was President Truman unwilling to back Vietnamese independence from colonial rule?

A7: He was unwilling to back Vietnamese independence because he saw the struggle as part of the much larger Cold War against communism. He was unwilling to back the Vietminh because of Ho Chi Minh's membership in the Communist Party.

Q8: Why did President Johnson have difficulty reassessing his war strategy in 1968?

A8: He had difficulty because his own advisers disagreed on the best course to take. Some believed ground troops should invade North Vietnam, and some believed Johnson's war policies were too extreme.

Q9: How did the assassination of Robert Kennedy affect the 1968 presidential race?

A9: Kennedy had won the California primary and was a favorite to win the Democratic nomination. In what turned out to be a very close presidential election, had Kennedy, rather than Hubert Humphrey, faced Nixon, the Democrats might have won the election.

Q10: Why did Ho Chi Minh believe the United States would support the Vietnamese nationalist movement?

A10: He thought the movement would receive U.S. support because he believed that Vietnam's fight for independence from France was similar to the American fight for independence from Britain.

	0	1	2	3	4	5	6	7	8	9
0	0.911	0.427	0.869	0.837	0.823	0.643	0.354	0.709	0.680	0.769
1	0.735	0.872	0.590	0.694	0.309	0.667	0.612	0.750	0.571	0.548
2	0.911	0.573	0.963	0.852	0.717	0.727	0.564	0.830	0.690	0.815
3	0.921	0.482	0.963	1.000	0.915	0.759	0.518	0.896	0.610	0.859
4	0.911	0.573	1.000	0.915	0.947	0.772	0.552	0.830	0.704	0.847
5	0.812	0.577	0.859	0.763	0.668	0.987	0.556	0.848	0.887	0.687
6	0.532	0.571	0.639	0.582	0.334	0.828	0.715	0.617	0.828	0.572
7	0.666	0.685	0.749	0.704	0.301	0.961	0.700	0.963	0.836	0.602
8	0.562	0.532	0.563	0.654	0.331	0.724	0.564	0.568	0.724	0.454
9	0.378	0.644	0.411	0.398	0.413	0.434	0.889	0.376	0.279	0.733

Figure 16. Average of Three Max Semantic Similarity for Sample2

Figure 16 shows the result of sample2. Each term would select four definitions with largest semantic similarity values from this result as alternative choices:

- *Q1: [0.82299499 0.83664021 0.86904762 0.91071429]*
- *Q2: [0.69395712 0.73504274 0.75 0.87179487]*
- *Q3: [0.82962963 0.85185185 0.91071429 0.96296296]*
- *Q4: [0.91534392 0.9212963 0.96296296 1.000000]*
- *Q5: [0.91071429 0.91534392 0.94736842 1. 000000]*
- *Q6: [0.84848485 0.85925926 0.88666667 0.98666667]*
- *Q7: [0.63896104 0.71515152 0.82769231 0.82769231]*

- *Q8: [0.74918301 0.83602564 0.96102564 0.96296296]*
- *Q9: [0.56825397 0.65410628 0.72380952 0.72380952]*
- *Q10: [0.43386243 0.64444444 0.73333333 0.88888889]*

For example, the multiple selections for Q1are:

- *A5: 'The United States wanted new markets and military advantages. Some Americans wanted to spread their Christian faith to other countries. In addition, many Americans believed that expanding into the Pacific was their manifest destiny.' (SIM Q1 & A5: 0.82299499)*
- *A4: 'Colombia's Senate refused to ratify a treaty that would have allowed the United States permanent use of the land needed to build a canal through the Isthmus of Panama. When Panamanian rebels revolted against Colombia, the United States supported them in hopes that after their victory, they would agree to the U.S. canal plan. When Panama declared independence, the United States swiftly recognized the new country, and a new canal treaty was soon signed.' (SIM Q1 & A4: 0.83664021)*
- *A3: 'Possible answers include any two of the following: Having the troops in Mexico increased the risk of war between the United States and Mexico; U.S. soldiers made no progress in capturing Pancho Villa; and Wilson was compelled to pay attention to developments in the early years of World War I.' (SIM Q1 & A3: 0.86904762)*

- *A1: 'The United States wanted a friendly government in Panama which would support negotiations to allow the United States to build a canal connecting the Atlantic and Pacific Oceans.' (SIM Q1 & A1: 0.91071429)*

CHAPTER NINE

COMPARISON

We used sample2 as input, ran it with both the original program(old version) and improved program(new version). The improved program selected four choices based on calculated semantic similarities, and the original program selected four choices randomly. Both versions of the program would add the correct choice (definition or answer) to the list of choices to be presented to the test taker.

Table 8. Example1: New Version and Old Version.

New Version	Old Version
(1 of 5) Why was the United States interested in expanding its territories in the late 1800s? [1] The United States wanted new markets and military advantages. Some Americans wanted to spread their Christian faith to other countries. In addition, many Americans believed that expanding into the Pacific was their manifest destiny. [2] Possible answers include any two of the following: Having the troops in Mexico increased the risk of war between the United States and Mexico; U.S. soldiers made no progress in capturing Pancho Villa,	(1 of 5) Why was the United States interested in expanding its territories in the late 1800s? [1] The United States wanted new markets and military advantages. Some Americans wanted to spread their Christian faith to other countries. In addition, many Americans believed that expanding into the Pacific was their manifest destiny. [2] Kennedy was a firm believer in the domino theory, and after the two Cold War disasters that began Kennedy's presidency, he hoped that aiding South Vietnam would be a sign of continued U.S. strength and resolve.

and Wilson was compelled to pay attention to developments in the early years of World War I. [3] The United States wanted a friendly government in Panama which would support negotiations to allow the United States to build a canal connecting the Atlantic and Pacific Oceans. [4] Colombia's Senate refused to ratify a treaty that would have allowed the United States permanent use of the land needed to build a canal through the Isthmus of Panama. When Panamanian rebels revolted against Colombia, the United States supported them in hopes that after their victory, they would agree to the U.S. canal plan. When Panama declared independence, the United States swiftly recognized the new country, and a new canal treaty was soon signed.	[3] He was unwilling to back Vietnamese independence because he saw the struggle as part of the much larger Cold War against communism. He was unwilling to back the Vietminh because of Ho Chi Minh's membership in the Communist Party. [4] The United States wanted a friendly government in Panama which would support negotiations to allow the United States to build a canal connecting the Atlantic and Pacific Oceans.

In example1, new version of the improved program selected Q1 as a question, and selected four definitions with largest semantic similarity values as choices: A5: 0.82299499, A4: 0.83664021, A3: 0.86904762, A1: 0.91071429. All selections related to Spanish-American War. What's more, all four selections were definitions of a "why" question, all of the selections were plausible, which made this question hard to be used exclusion method to get the correct answer. The old version of the original program got four selections randomly, in this example, only selection1 and selection4 came from the chapter of Spanish-

American War, but selection2 and selection3 were about Vietnam War, they were easy to be ruled out. So in this example, comparing to old version of the original program, our new version of the improved program generated questions with more plausible choices.

Table 9. Example2: New Version and Old Version.

New Version	Old Version
How did the assassination of Robert Kennedy affect the 1968 presidential race? [1] Kennedy had won the California primary and was a favorite to win the Democratic nomination. In what turned out to be a very close presidential election, had Kennedy, rather than Hubert Humphrey, faced Nixon, the Democrats might have won the election. [2] Possible answers include any two of the following: Having the troops in Mexico increased the risk of war between the United States and Mexico; U.S. soldiers made no progress in capturing Pancho Villa; and Wilson was compelled to pay attention to developments in the early years of World War I. [3] Kennedy was a firm believer in the domino theory, and after the two Cold War disasters that began Kennedy's presidency, he hoped that aiding South Vietnam would be a sign of continued U.S. strength and resolve.	How did the assassination of Robert Kennedy affect the 1968 presidential race? [1] Possible answers include any two of the following: Having the troops in Mexico increased the risk of war between the United States and Mexico; U.S. soldiers made no progress in capturing Pancho Villa; and Wilson was compelled to pay attention to developments in the early years of World War I. [2] The United States wanted a friendly government in Panama which would support negotiations to allow the United States to build a canal connecting the Atlantic and Pacific Oceans. [3] The United States wanted new markets and military advantages. Some Americans wanted to spread their Christian faith to other countries. In addition, many Americans believed that expanding into the Pacific was their manifest destiny.

[4] He had difficulty because his own advisers disagreed on the best course to take. Some believed ground troops should invade North Vietnam, and some believed Johnson's war policies were too extreme.	[4] Kennedy had won the California primary and was a favorite to win the Democratic nomination. In what turned out to be a very close presidential election, had Kennedy, rather than Hubert Humphrey, faced Nixon, the Democrats might have won the election.

In example2, new version of the improved program selected Q9 as a question, and selected four definitions with largest semantic similarity values as choices: A8: 0.56825397, A4: 0.65410628, A6: 0.72380952, A9: 0.72380952. Except for A4, the other three selections related to Vietnam War, especially selection1 and selection3 are highly related. But for old version of the original program, only correct answer related to question, the other three selections had nothing to do with the keyword "Kennedy", they were all easy to be ruled out.

CHAPTER TEN

LIMITATION

Comparing to pick selections randomly, our new version program highly improved the difficulty of questions. But natural language processing is not 100% accurate, our program has three limitations:

- The JIGSAW program could not recognize some special nouns like person names that were not in WordNet corpus. For example, Pablo Picasso, who was a Spanish Painter, this name could not be processed by the Resnik similarity model.
- The program worked better on long sentences than short sentences. The reason was that we analyzed semantic similarity between each keyword of sentences, more keywords we had, more accurate the result we got. Short sentences might not have enough keywords to represent its full meaning.
- We did not have a system or statistical method to analyze if the choices are hard enough, to decide which program could generate more challenging tests that are less amenable to solving by elimination; subject knowledge needed to make correct selection.

CHAPTER ELEVEN

CONCLUSION AND FUTURE WORK

This project has presented several natural language processing methods. Several NLTK libraries have been imported, these libraries provided functions as tokenizing sentence into words, removing stop words, tagging and extracting nouns. The core of this project was to develop a word sense disambiguation model. The Resnik similarity, as measured using information content and most informative subsumer, was shown to be useful in resolving word sense disambiguation problem in the Resnik algorithm. Based on the Resnik algorithm, the JIGSAW algorithm was developed by inserting Gaussian distribution and computing a factor R that takes into account the rank of senses in WordNet. We experimented with these WSD algorithms and semantic similarity measures for keywords in terms and definitions and evaluated their performance on several samples. Based on the comparison between our experimental results and old version program results, we find that our new version is performing much better than the original version of the program.

For this project, the author of the project (and advisor Dr. Voigt) were the judges who determined that the multiple choice questions generated by the new program were superior to the questions produced by the original QAW program. In virtually each instance we have tested, the improvement was very obvious, and it is difficult to imagine that anyone would disagree with our findings.

Naturally, further improvements to QAW, which would likely be more subtle, would warrant an independent and more objective evaluation.

As discussed earlier, there are several limitations in our program, a library named StanfordNERTagger can be used to tag special names has not been used yet, many other factors are not taken into account for solving WSD problem. Making use of these libraries and factors we may able to design better multiple-choice generator.

REFERENCES

[1] “Easy Notecards”, a quiz generator website. http://www.easynotecards.com/quiz/2307. (Accessed 22 January 2017)

[2] J. Perkins, “Python 3 Text Processing with NLTK 3 Cookbook”, Birmingham, UK: Packt Publishing Ltd, ISBN: 978-1-78216-785-3, November 2010.

[3] S. Bird, K. Ewan, J. Baldridge, and E. Loper, "Multidisciplinary instruction with the Natural Language Toolkit", Proceedings of the Third Workshop on Issues in Teaching Computational Linguistics, ACL, June 2008.

[4] Y. Duan, C. Cruz, “Formalizing Semantic of Natural Language through Conceptualization from Existence”, *International Journal of Innovation, Management and Technology*, pp. 37-42, 2011.

[5] K. Wolk, K. Marasek. "A Sentence Meaning Based Alignment Method for Parallel Text Corpora Preparation", *Advances in Intelligent Systems and Computing*. Springer, ISBN 978-3-319-05950-1. ISSN 2194-5357, pp. 107-114.

[6] S. Bird, K. Ewan, and E. Loper, “Natural Language Processing with Python”, Sebastopol, CA: O’Reilly Media, Inc., June 2009.

[7] “WS4J Demo” semantic similarity calculator. http://ws4jdemo.appspot.com. (Accessed 15 March 2017)

[8] S. Torres, A. Gelbukh, “Comparing Similarity Measures for Original WSD Lesk Algorithm”, Vol.43, pp. 155-166, 2009.

[9] E. Agirre and P. Edmonds, "Word Sense Disambiguation: Algorithms and Applications", Text, *Speech and Language Technology Series*, Springer, Vol. 33, ISBN: 978-1-4020-6870-6, 2007.

[10] G. A. Miller, R. Beckwith, C. D. Fellbaum, D. Gross, K. Miller, “WordNet: An online lexical database”. pp. 235-244, 1990.

[11] G. Hirst, D. Stonge, “Lexical chains as representations of context for the detection and correction of malapropisms”, in Fellbaum, pp. 305-332, 1998.

[12] Disambiguation Algorithm. https://www.cs.cmu.edu/afs/cs/project/jair/pub/volume23/montoyo05a-html/node9.html. (Accessed 20 March 2017)

[13] G. Ramakrishnan, B. Prithviraj, and P. Bhattacharyya, “A Gloss Centered Algorithm for Word Sense Disambiguation”, Barcelona, Spain: Proceedings of the ACL SENSEVAL, pp. 217-221, 2004.

[14] P. Resnik, “Disambiguating Noun groupings with respect to WordNet Senses”, In Proceedings of the Third Workshop on Very Large Corpora, pp. 54-68. Association for Computational Linguistics, 1995.

[15] P. Resnik, "Semantic Similarity in a Taxonomy: An Information-Based Measure and its Application Natural to Problems Language of Ambiguity in Natural Language”, in College Park, MD, pp. 96-130, 1999.

[16] P. Basile, M. de Gemmis, A.L. Gentile, P. Lops and G. Semeraro, "UNITA: JIGSAW algorithm for Word Sense Disambiguation", Bari, ITALY, Proceedings of

the 4th International Workshop on Semantic Evaluations, pp. 398-401. Association for Computational Linguistics, June 2007.

[17] C. Leacock, M. Chodorow, “Combining Local Context and WordNet Similarity for Word Sense Identification”, In C. Fellbaum, editor, *“WordNet: An Electronic Lexical Database”*, pp. 266-283. MIT Press, 1998.

[18] WordNet Interface. http://www.nltk.org/howto/wordnet.html. (Accessed 2 April 2017)

[19] Numpy function of Gaussian distribution. https://docs.scipy.org/doc/numpy/reference/generated/numpy.random.normal.html. (Accessed 16 April 2017)

[20] B. Wlodzimierz, “The Normal Distribution: Characterizations with Applications”, Springer-Verlag, ISBN 0-387-97990-5, 1995.

[21] Z. Wu, M. Palmer, “Verb semantics and lexical selection”. In Proceedings of the 32nd Annual Meeting of the Associations for Computational Linguistics, pp. 133-138, Las Cruces, New Mexico, 1994.

[22] O. Dameron, C. Bettembourg, and L. Joret, “Quantitative cross-species comparison of GO annotations: advantages and limitations of semantic similarity measure”, INSERM U936, France, 2010.

[23] M. Lesk. “Automatic sense disambiguation using machine readable dictionaries: How to tell a pine cone from an ice cream cone”. In Proceedings of SIGDOC ’86, pp. 20-29, 1986.

[24] J. Singh, M. Saini, S. Siddiqi, “Graph Based Computational Model for Computing Semantic Similarity”, India, Elservier Publications, pp. 501-507, 2013.

[25] K. Voigt, QAW.py, a Python program, October 2015.

[26] K. Voigt, Resnik.py, a Python program, February 2017.