STAIRS 2016
D. Pearce and H.S. Pinto (Eds.)

doi:10.3233/978-1-61499-682-8-75

Tackling the Winograd Schema Challenge Through Machine Logical Inferences

Nicos ISAAK [a,1], Loizos MICHAEL [a]
[a] *Open University of Cyprus*

Abstract. Levesque has argued that the problem of resolving difficult pronouns in a carefully chosen set of twin sentences, which he refers to as the *Winograd Schema Challenge (WSC)*, could serve as a conceptually and practically appealing alternative to the well-known Turing Test. As he said, probably anything that answers correctly a series of these questions is thinking in the full-bodied sense we usually reserve for people. In this paper we examine the task of resolving cases of definite pronouns. Specifically, we examine those for which traditional linguistic constraints on co-reference as well as commonly-used resolution heuristics are not useful, or the procedure they follow is very similar to a statistical approach, without invoking common logic like humans do.

Keywords. Winograd Schema Challenge, WSC, Machine Learning, Knowledge and Reasoning, NLP

1. Introduction

One of the most important challenges in computer science is the understanding of how systems that acquire and manipulate commonsense knowledge can be created [1]. With the creation of cognitive systems that are based on machine learning, humanity aims for systems that will replace or substitute basic human abilities, so that humans can relate and interact with them. Past experimental work, that was based on unaxiomatized knowledge acquisition from text, was promising for extracting knowledge from text automatically [2]. It aims to make systems possible to acquire knowledge on a large scale by learning, and then use it robustly for reasoning. It is widely believed that logical inferences are necessary in order to build natural language representations, as well to reason about information encoded in representations. In this work, we present a technique that focuses on commonsense knowledge which can be retrieved and learned via a supervised learning approach, called auto-didactic [3]. It includes, among others, the acquisition and the extraction of general inference rules that could help us to solve different RTE problems, like the WSC problem.

[1]Corresponding Author: Nicos Isaak, Open University of Cyprus, PO Box 12794, 2252, Latsia, Cyprus; E-mail: nicos.isaak@st.ouc.ac.cy

2. The Winograd Schema Challenge (WSC)

The WSC can be seen as a new type of Turing Test. Rather than basing the test on the sort of short free-form conversation suggested by the Turing Test, a machine should be able to demonstrate that it is thinking without having to pretend to be somebody [4]. This shift of interest might have happened because there are systems that have been developed to deceive, and this is not a demonstration of an intelligent behavior [5,6].

The WSC consists of sentence pairs (twin sentences), and the objective is to resolve a definite pronoun to one of its two co-referents, in each sentence. The co-referents belong to the same gender, and both are either singular or plural. This property makes the resolution harder because the pronoun can not be resolved just by considering the gender and its number in order to match it with each of the present entities. Additionally, the sentence also contains a special word, which when replaced by another word, the answer also changes. In some WSC works the pronoun is given (e.g., [7]), and in others it has to be resolved through a question (like in this work).

The following WSC sentence pair illustrates how difficult the problem can be.
(1). The cat caught the mouse because it was clever. (2). The cat caught the mouse because it was careless. Humans through commonsense ability can resolve the pronoun (*it -> cat*), but co-reference and statistical resolvers do not have this ability.

The motive behind the WSC is to simulate human-like reasoning in machines in order to test the machines ability to answer commonsense questions regarding sentence comprehension. A machine presents that kind of behavior when it reaches a conclusion from a situation like humans do. The ultimate goal is to build machines that autonomously acquire relevant background knowledge and use it afterwards to solve different kind of problems [8]. For learning to be meaningful to solve problems like the WSC, machines need to be able to deal with missing information and employ new learning techniques to act as expert assistants able to collaborate with humans. This challenge is of great significance and that became particularly apparent by the various competitions that have been announced, like the Nuance Communications annual competition, starting this summer.

2.1. Previous Work on the WSC

Rahman and Ng's system [7] tries to find the most probable pronoun candidate, through a number of lexicalized statistical techniques. Through a *Ranking based approach (SVM)* it combines the features derived from different knowledge resources (Web Queries, Framenet, OpinionFinder, etc.). Although this technique achieves a high score of prediction (73,05%), in the case of sentences that could be equally alike, the system fails [5]. Contrariwise, our system overcomes such problems due to its ability to acquire commonsense knowledge similar to the one employed by humans [9,10].

There is another work [11] that uses an Integer Linear Programming approach. It acquires statistics in an unsupervised way from multiple knowledge resources (Gigaword corpus, Wikipedia Wikifier, Web Queries and polarity information), through the training of a co-reference model by learning a pairwise mention scoring function. It separates the sentences into three types and tries to solve the first two (from the previous work's dataset). Although it achieves a high score of prediction (76%), the system fails to answer 27% of sentences that belong to the third type.

The WSC dataset that was used in the presented work, was also used in another work [12]. However, in that work, only the 34% of the dataset could be tested. Contrary to our system, it rejects a large amount of sentences and fails to resolve the pronouns that refer to person names. It consists of four answering modules which use world knowledge with an aggregation mechanism (ConceptNet, Web Queries, Narrative chains, sentiment analysis) and achieves a score of 73%.

Another technique that is based on Answer Set Programming (ASP) [5,13] tries to retrieve the background knowledge directly from *Google* through fixed queries (e.g., ".*not.*lift.*because.*weak.*"), forcing it to return specific sentences that are semantically and structurally similar to the given WSC sentence. In contrast to our system, which can test any WSC sentence, the ASP based technique rejects a large amount of WSC sentences. It can only test two types of WSC sentences, causal and direct causal (38% of the WSC dataset), and correctly resolves the 70%.

Our work differs from previous work mainly in three aspects. *(1.)* It tries to solve the WSC problem through logical inferences similar to that employed by humans. *(2.)* It uses only one source for knowledge acquisition (the Wikipedia). *(3.)* It does not use any traditionally Web Queries for knowledge acquisition, but an upgrade off-line version of the Websense Engine [9,10]. Below we present the Websense Engine's structure, followed by our methodology on the WSC problem.

3. The Websense Engine

The Websense engine can output logical inferences, so that we can relate and interact with it. It is able to respond to user queries provided in natural language text, with inferences that are implied by the given queries according to the collected human knowledge (see Table 1). It has been developed in the strong sense that even an individual piece of knowledge might not be stated explicitly in a single Web page, but be implicitly encoded across the Web. Our goal was to use the Websense engine's *Logical inference* feature in order to solve the WSC problem.

Table 1. User interaction with the Websense engine

Query: `Users pay with something.`
Scene Constructor transformation: $\texttt{users}(t_1) \wedge \texttt{something}(t_4) \wedge \texttt{pay}(t_1, t_4)$
Logical Inferences: $\texttt{money}(t_4)$
Response: `users pay money.`

3.1. Websense Engine Structure

The Websense engine works in two modes. In the *first mode* it can learn anything from the Web related to a single word. This results from a component called the *Crawler* which locates the best results that are consisted in HTML format. Then, it downloads them and removes the HTML tags. Afterwards, the clean-pages are being split into sentences, through a component called the *Splitter*, and parsed through the *Stanford Parser* [14]. Stanford typed dependencies were designed to provide a simple description of the grammatical relationships in a sentence that can easily be understood and used by people to extract textual relations [15]. Afterwards, each parsed sentence is being parsed again

through another component (the *Scene Constructor*), which produces first order semantic scenes (see table 1). Next, the *Scene Constructor's* scenes are being given as an input to another component called the *Learner* [2], to produce the required knowledge we are interested in. In the *second mode*, the Websense engine can accept any user query to return the commonsense conclusions through another component called the *Reasoner* [2]. The two-mode procedure is being controlled by a component called the *Manager*.

4. Our Approach-Methodology

For the purpose of this work, we modified and use an off-line version of the Websense Engine (we call it the *Wikisense*) that acquires knowledge from a downloaded version of the English Wikipedia. We did this for two reasons. Firstly, the web knowledge acquisition through the *Crawler* is very lengthy (e.g., for a 9 word sentence, the searching can last from 30 minutes to 24 hours). Secondly, Wikipedia is being developed on the WWW, is open source and free for anyone to write. For this reason the Wikipedia is indirectly related with the Websense Engine development.

Let us take the following WSC example, which it will subsequently be referred to as the *catch* example: *{[WSC sentence: The cat caught the mouse because it was clever.] [Question: Who is the clever?] [Possible pronoun targets: cat, mouse] [Correct pronoun target: cat]*}. *Wikisense's* purpose is to take the input sentence, the question and the two possible pronoun targets, to return the correct pronoun target.

4.1. Learning Framework for the WSC

We tested the *Wikisense* on the WSC through the learning for each word in a WSC sentence, but we were unable to resolve pronouns. We started wondering how we could train it to be a pronoun resolution expert. To strengthen its knowledge and reasoning abilities, we focused on human's knowledge and reasoning ability. When someone has to figure out pronouns, he mainly focuses on sentence's verbs, nouns and adjectives, and subsequently through these *relations* tries to find the correct answer (these are very important words [2]). However, in order to achieve that, we had to steer the *Wikisense's* components to work on the WSC problem.

Initially we made some changes to the *Crawler*. Because of the *Crawler's idleness* we created an off-line version for the English Wikipedia (we call it the *Indexer*). We updated the *Indexer* to search through a combination of subjects (e.g., noun, verb, adjective) instead to a single one. We added *synonym* capabilities to search and learn more widely (e.g., instead of searching only for the word *cat*, we could also search for the words *moggy, kitty*). Furthermore, we added pluralization capabilities and an extra verb capability which is related to the verb's root.

We limited the output of the *Splitter* to return only sentences that contain the words we are interested in. We did this to drive the learning procedure to a specific domain excluding generalities and likelihood of confusion and to reduce time complexity.

Also, we updated the *Scene Constructor*. The *Scene Constructor* takes as input the typed dependencies that are produced from the *Stanford Parser* (see table 2) and returns first order semantic scenes. For example, if we give as input the *catch* sentence dependencies, through a pre-running semantic option (e.g., S_DobjNsubj) it returns the scene

$\texttt{cat}(t_2) \wedge \texttt{mouse}(t_5) \wedge \texttt{catch}(t_2, t_5)$, which tells us that a cat catches a mouse.

Table 2. Stanford and Spacy Parser Output: For the Catch Sentence

Stanford:	`nsubj(caught-3, cat-2) dobj(caught-3, mouse-5) ....`
Spacy:	`verb: catch catch_subject: cat, catch_object: mouse`

This relation has been created because there is a direct connection between the entities that are being connected through *dobj* and *nsubj* dependencies, and through them we can create *subject-verb-object* relations [15]. However, we had to create more semantic rules like the above for using the *Scene Constructor* on the pronoun resolution. New relations have been created through the study of the *Stanford Parser's* typed dependencies manual, and at the end the *Scene Constructor* ended with multiple useful, but time consuming relations. This is why we updated the parsing procedure to use it in combination with a faster one. Stanford parser is a well tested parser which is used widely, but the speed is not one of its advantages. We wanted to parse thousands of sentences to test our system abilities and not to be restricted by the parser's speed. For this reason, we are using a new parser on the NLP field that is being called the *Spacy* (see table 2). *Spacy* features a high performance parser which offers the fastest syntactic parsing in the world (https://spacy.io). Hence, we updated the *Scene Constructor* to work also with *Spacy*. Through *Spacy*, the *Scene Constructor* finds the sentence's subjects and objects and tries to create semantic scenes. Because of *Spacy's* speed, and *Sanford Parser's* legacy and acceptance, we decided to use both parsers. We use the *Stanford Parser* to parse the WSC sentence to locate the candidate pronoun targets positions in the sentence and to correlate the question with the WSC sentence. If the *Stanford Parser* cannot give any useful output then we also use the *Spacy*. Afterwards, we use only the *Spacy* to parse the Wikipedia's sentences that are being located by the *Indexer*.

Since the *Wikisense's* procedure was controlled by the *Manager*, we strengthened the *Manager* with the ability to guide it to the pronoun resolution. For better understanding, we are showing the *Manager's* controlling procedure through the next simplified example.

4.2. A Snapshot of a Simplified Running Example

At first the *Manager* loads and parses the WSC sentence to produce the typed dependencies. *Manager* also checks the question and in correlation with the two possible answers that have to be located in the WSC sentence creates the *Indexer's* searching keywords. If at least one of the two possible answers is a proper name, the *Chambers and Jurafsky's* [16] narrative chains are used to replace them. In this way, similar issues faced by other works (e.g., [12]) are avoided.

For the keyword creation, the *Manager* keeps only the verbs, nouns and adjectives from the WSC sentence and the question. It splits the keyword procedure into two parts (sentence and question) and produces the necessary *Indexer's* keywords for the Wikipedia searching. The objective is to have keywords that correlate the two candidate pronoun targets between them and also with the question (e.g., see table 3). The first sentence's part keyword (e.g., `mouse*cat/catch`) is the only one that is created through the two parsers and the *Scene Constructor*. If the two parsers are not able to return re-

sults, an alternative heuristic procedure is called which creates the keyword, using the positions of the two candidate pronoun targets. If the first sentence's part keyword cannot be created, the current WSC sentence is abandoned. Similarly, if the two candidate nouns cannot be located then the sentence is also abandoned (e.g., they might be stated in the answer as *mouse, feline* instead of *mouse, cat*; we maximally eliminate these kind of problems, using another heuristic approach).

If the WSC sentence is not abandoned, then the keyword of the first sentence's part can be directly used to output semantic scenes, to the knowledge needed to solve the pronoun resolution (e.g., $\texttt{cat}(t_2) \wedge \texttt{mouse}(t_5) \wedge \texttt{catch}(t_2, t_5)$). All semantic-like parsers extract only some of the semantic relations encoded in a given text and for the other parts logic is needed. Hence, for the other keywords and for finding the correct pronoun target we have to use the other *Wikisense's* components. The other keywords are being created as follows, in the following order (see table 3): *(1).* Between verbs that are included in the first sentence's part keyword and the verbs, adjectives, nouns from the question's part (e.g., 2Q). *(2).* Between the two candidate pronoun targets, and the verbs, adjectives, nouns from the question's part (e.g., 3Q).

Table 3. Indexer's Keyword Queries for the Catch Sentence

(1Q) cat*mouse/catch, (2Q) catch/clever, (3Qa) cat/clever, (3Qb)mouse/clever

After the keywords creation, the *Manager* calls the *Indexer* with the new keyword (e.g., `catch/clever`), and waits until it returns the requested amount of sentences. The default is one thousand Wikipedia's sentences for each WSC sentence (with this number our tests show useful results at a minimum time process). For each keyword the *Indexer* tries to find the specified number of requested sentences through different settings. It tries to find one thousand sentences that contains the searching words and if that amount of sentences cannot be found, it tries with other keyword settings (e.g., synonyms). If the specified number cannot be retrieved then the procedure continues with the current retrieved number (*>0*).

When the *Indexer* finishes (it runs in correlation with the *Splitter*), the *Manager* calls the other components that follow. All the sentences that contain keyword synonyms or keyword verb roots are being replaced with the original keyword words. Then, the *Scene Constructor's* semantic scenes are being given as an input to the *Learner*, and if important scenes are being included, then a knowledge file is being produced. This is the file that will be used for drawing conclusions (reasoning) based on knowledge that will show the pronoun target. If the knowledge file cannot be produced then the whole procedure runs again with another *Indexer's* searching keyword.

In our example, after adding the keyword's `cat*mouse/catch` semantic scenes and after the running of the keyword `catch/clever`, a useful knowledge file is being produced (see table 4). If we carefully observe this knowledge file we can extract the following meaning: *Between the cat and the mouse, the clever cat is catching the mouse.*

All variables in the head of each rule are assumed to be universally quantified over that rule. Furthermore, the variables that appear in a line of the body but not in the head must be explicitly quantified and their scope is the formula appearing in that line. This type of rule encodes a linear threshold, where a sufficient number of formulas in the body of the rule need to hold for the head to also hold (threshold=1.0) [9]. Also, the learning

Table 4. Learner's Knowledge. The first rule means that a cat catches a mouse. The second rule that a mouse is being caught by somebody else, and the third rule that the clever catches somebody.

Rule 1:
cat(x)
∃v : catch(x, v) weight(1.03031)
∃v : catch(x, v) ∧ mouse(v) weight(1.000000)
Rule 2:
mouse(x)
∃v : catch(v, x) weight(1.110000)
Rule 3:
clever(x)
∃v : catch(x, v) weight(1.044202)

algorithm which is being employed provides a priori guarantees on the appropriateness of the responses [8].

For the final pronoun resolution, the *Manager* extracts the answer through the *Reasoner*. The *Reasoner* draws inferences by applying a relational knowledge base on a set of input predicates. After the Websense modification, the *Reasoner* automatically takes a query from the *Manager* with a preformed scene. The *Manager* indirectly asks if the *Reasoner* can conclude anything else for this query. It asks if it knows anything more for the first sentence's part keyword semantic scene. For instance, in our *catch* example we are taking as an inference that the *cat* is also *clever* (for the query: $\texttt{cat}(v_1) \wedge \texttt{mouse}(v_2) \wedge \texttt{catch}(v_1, v_2)$ the Reasoner conclusion is the: $\texttt{clever}(v_1)$).

Generally the *Manager* overviews the learning procedure's returned amount of knowledge. If the problem is solved, it returns the pronoun target. Otherwise, it keeps into the knowledge base only the first sentence's part semantic scenes (e.g., mouse*cat/catch) and proceeds to the next *Indexer's* call. If a keyword is not available, the *Manager* returns a message that the pronoun cannot be resolved. Below, we explain in detail the *Wikisense's* knowledge acquisition process.

4.3. Detailed Wikisense Process for knowledge Acquisition

We wanted to learn which one was the best keyword sequence that could return the best knowledge results, and also to build a principle mechanism to combine in a single *Wikisense's* inference, multiple *Wikisense's* inferences.

Other works in the past used different hybrid approaches to combine shallow analysis (through synonyms) with both theorem to solve different RTE problems [17]. They mentioned that the use of synonyms helped the learning procedure. Hence, the usage of the synonyms as a direct keyword *plugin* could also help the *Wikisense*. Continuing from the synonyms we also decided to use the *antonyms* because both synonyms and antonyms share common principles.

To discover and use the best keyword sequence or the best keyword setting we used the WSC sentences from Rahmans and Ng's work [7]. We call these sentences *supporting* and not training because we used them only as a *guide* to the *Wikisense's* learning procedure. For this reason, we built another component (*The Questionnaire*) to transform the Rahmans and Ng's WSC sentences to include also a question for each WSC sentence. Next, we updated the *Reasoner* to return the *value confidence* for each WSC sentence's running keyword. This is an integer value that shows the *Reasoner* confidence for the returning results. For instance, we need to know how many rules in the knowledge file can specify that the subject of the word *catch* is also *clever* (see table 4). Particularly,

we were calling the *Wikisense* forcing it to return a feature vector, that we were going to examine in a later step. The vector consists of fourteen values which show verb relations, verb-synonym relations, verb-antonym relations, and noun relations. Below, we explain how the *Manager* creates the feature vector for totally 1697 supporting WSC sentences.

(1.) Through the two parsers, the *Manager* determines if a negation is addressed to any of the two possible pronoun targets (labeled as negF). Negation is important because it changes the rules direction [11,7]. *(2.)* Also, it determines whether the two nouns appear in reverse order, contrary to the way that they appear in the WSC answers (labeled as revF). *(3.)* It runs the first keyword that connects the sentence with the question (e.g., `catch/clever`) and stores the *value confidence* without synonyms or antonyms enabled (labeled as Vx-Vy). *Vx* is the confidence for the first candidate noun and *Vy* is the confidence for the second candidate noun (e.g., *Vx* shows that the subject of the verb catch is also clever). *(4.)* The *Manager* also runs the same keyword as previously (e.g., `catch/clever`) and stores the *value confidence* directly through synonyms (labeled as Sx-Sy). For instance, if catchSyn1, catchSyn2 are the synonyms of the word "catch" and cleverSyn1, cleverSyn2 are the synonyms of the word "clever", the *Manager* calls the *Indexer* for all the correlations between the two word synonyms (e.g., `catch/cleverSyn1`, `catch/cleverSyn2`, etc.). For each correlation it determines the *value confidence* of the first noun (*Sx*) and of the second noun (*Sy*) and adds them to a score counter (Sxcounter for noun1, Sycounter for noun2). At the end it exports the counters to the feature vector (e.g., *Sy* shows that the object of the verb is also clever). *(5.)* The same process is repeated for the antonyms of the same keyword and the resulting *value confidence* is stored (labeled as Ax-Ay). *(6.)* It runs the noun keywords (e.g., `cat/clever`, `mouse/clever`) and stores the *value confidence* without synonyms and antonyms enabled (labeled as Nx-Ny). *Nx* shows that the first noun (e.g., *cat*) is clever and the *Ny* that the second noun (e.g., *mouse*) is clever. *(7.)* Through a heuristic approach, the *Manager* determines also the times that the first noun (e.g., *cat*) appears right before (labeled as NBx) or right after (labeled as NBy) the question's part word (e.g., *clever*). After this, it applies semantic scenes in the knowledge file without the usage of the *Spacy* (in case that the *Spacy* might not returned useful results). Also, it determines the times that the first keyword's verb word (e.g., *catch*) appears before (labeled as VBx) or after (labeled as VBy) the question's word (e.g., *clever*).

Hence, for each WSC we are storing the values: *NegF-RevF-Vx-Vy-Sx-Sy-Ax-Ay-Nx-Ny-NBx-Nby-VBx-VBy*. For instance, Rahmans and Ng's WSC *catch* sentence (feature vector) is *False-False-0-0-1-0-0-0-0-0-1-0-9-2*.

The feature vector procedure ran for almost a week. We found that one of the best patterns that had to to be followed was the one with the same sequence of steps, mentioned above (see Algorithm 1). In each step if the *Manager* can resolve the pronoun, it returns the correct pronoun target and proceeds to the next WSC sentence, otherwise it tries to resolve the pronoun through the next step. If no other step is available, it returns a message saying that the current pronoun cannot be resolved and proceeds to the next WSC sentence. We observed that more emphasis is given to the verb-like keyword relations (e.g., `catch/clever`) than to the noun-keyword relations. If we reverse these steps then the *Wiksense's* success on the pronoun resolution decreases. It also decreases if we reverse the steps, from bottom to top, showing our parsers usefulness (e.g., Vx-Vy step is more important than the last two steps). The *conf=30%* shows that if the *Wikisense* inference decision for e,g., the *Vx* is *stronger* than *Vy* by at least 30% then the pronoun

Algorithm 1. Wikisense's procedure for each WSC sentence

```
1:  function ResolvePronoun (sent, negF, revF, question, answers)
2:      conf=30%, pairs=[(Vx, Vy), (Sx, Sy), (Ax, Ay), (Nx, Ny), (NBx, NBy), (VBx, VBy)]
3:      for pair in pairs do
4:          correctIndex=CalcValues (pair, negF, revF, conf, sent, question)
5:          if correctIndex!=-1 then return answers [correctIndex]
6:      return -1
7:  function CalcValues (pair, negF, revF, conf, sent, question)
8:      x, y= RunAndEstimate (pair, sent, question)
9:      if negF==True then x,y=y, x
10:     if revF==True then x,y=y, x
11:     if x > y and (x-y)/x >= conf then
12:         return 0
13:     else if y > x and (y-x)/y >= conf then
14:         return 1
15:     else
16:         return -1
```

target is the subject of the verb, and if the opposite exists, then the pronoun target is the object of the verb. We determined this percentage from our testings with the supporting data and we observed that it is similar to human decisions. If we are between two options and cannot easily determine which one is the pronoun target then we return the option that, according to our experiences, has more weight.

5. Experimental Settings and Results

In this section we present results that were obtained by applying the methodology described in this paper. The WSC sentences we use in our testing experiments derived from a WSC Library which is intended to be used by participants while developing their systems for the Nuance competition (http://www.cs.nyu.edu/faculty/davise/papers/OldSchemas.xml). The Library consists of 286 WSC sentences and we do not exploit the fact that each sentence has a twin sentence.

The *Wikisense* processed 286 sentences and for each sentence it was asked to determine the correct pronoun target, always replying with a noun name, or with the answers *don't know* or *unaccomplished*. *Don't know* means that the *Wikisense* could not determine the correct pronoun target, while *unaccomplished* that it could not proceed to knowledge acquisition because the first sentence's part keyword could not be created (see 1Q on table 3).

5.1. Baselines-Results

Our baseline is the *Stanford Core NLP* [18]. As shown in table 5, it correctly resolves 107 pronouns, incorrectly resolves 112 of them, and does not make any decision on the remaining 67.

Our System correctly resolves 170 pronouns, incorrectly resolves 89 and from the remaining 27, it returns the *don't know* answer for 12 sentences and the *unaccomplished*

Table 5. Results of the Stanford Core NLP, and the Wikisense (where _A shows the Adjusted scores)

	Correct	C_A	Wrong	W_A	Unresolved	U_A
Stanford Core NLP	107	140.5	112	145.5	67	0
Wikisense	170	183.5	89	102.5	27	0

answer for 15 sentences (see table 5). Hence, the whole procedure ran for 259 WSC sentences and our system achieved 65.6% score of prediction.

Since our system accepts as input the WSC sentence with the two possible pronoun targets, in order to ensure a fair comparison between the two systems, we have to ensure that the Stanford Core NLP will also resolve the correct pronoun target to one of the two candidate nouns. We can see this through the "Adjusted Score" (_A) columns of the table 5.

Comparison of the Adjusted scores (_A) shows that our system outperforms the Stanford Core NLP by 43 points. This suggests that our system is very useful on the co-reference resolution. It also shows that we can improve the Parser's usage on the co-reference resolution. A good pair of sentences that shows our system's usefulness is the following (in contrary to our system, Stanford Core NLP wrongly returns in both sentences the *table*): *(1) The table won't fit through the doorway because it is too wide. question: What is too wide? answers: The table, The doorway (2) The table won't fit through the doorway because it is too narrow. question: What is too narrow? answers: The table, The doorway*

5.2. Support Vector Machine Method

In order to show our system's usefulness on the WSC, we also tested the WSC library through a Support Vector Machine (SVM), that we created. As in Rahman and Ng [7], we used a ratio of 70-30 (70% of WSC sentences used for training and 30% used for testing). We randomly selected the 70-30 ratio from the WSC library and ran the procedure 100 times. The median score of prediction was 47% (minimum 32% and maximum 59%), showing that our initial *Wikisense* strategy yields better results.

5.3. Observations-Error Analysis

The *Wikisense* failed to answer 27 WSC sentences (9%). The *Manager* failed to create the first sentence's part keyword in 15 sentences and also failed to locate an important word from the question to create the necessary *Indexer's* searching keywords in another 4 WSC sentences. Moreover, it did not manage to find big amount of Wikipedia sentences or rich in content sentences for 8 WSC sentences.

WSC sentences were incorrectly answered because of the similar background knowledge that was found by the *Indexer*. For instance, for the twin sentences *30) The firemen arrived after the police because they were coming from so far away 31) The firemen arrived before the police because they were coming from so far away*, through the question *Who came from far away?* the system returned the same pronoun target (*the firemen*). We can see that the differences between the two sentences are negligible.

There are WSC sentences that are also very confusing, and this is another reason why our system incorrectly answers some of them. The majority of our WSC are Hard

WSC sentences and this is not an easy task [19]. We can see this from other works that test their systems on corpus subsets [12,5,13].

Also, ambiguity is ubiquitous in natural language and poses a special problem for text processing, where longer sentences tend to increase grammatical ambiguity. Furthermore, we have seen the *Wikisense* taking the same decision for some twin sentences and returning the same pronoun target. By removing these twin sentences the score rises to 70% (131 correct, 56 wrong). Hence, we can easily build on this system and enhance its decision mechanism.

6. Conclusions – Future Work

WSC suggests that emphasis on commonsense knowledge might benefit different NLP tasks. Designing and implementing such cognitive systems would arguably be a concrete step forward in endowing machines with the ability to *understand* text. In this work, we have demonstrated the usability of a technique, that can be applied on the WSC, through the acquisition of commonsense knowledge from the English Wikipedia. The results are very encouraging. Our study provides an insight on how learning and reasoning through knowledge acquisition can fruitfully interact for pronoun resolution. There is still room for improvement, but our approach works well with respect to the WSC.

Researchers can build on this work, to use it in various problems, or to enhance its capabilities. Future possible tasks for performance improvement may include the followings: *(1.) Manager* improvements through a better keyword generator. This can lead to the acquisition of richer knowledge and benefits the system's commonsense ability. Recall that our score rises to 70% if we remove the twin sentences that have the same *Wikisense's* inferences. We consider this property advantageous because it provides the ability to use another keyword setting or generator which will easily help to the *Wikisense's* improvement. *(2.)* Maybe *Wikipedia's* substitution with another knowledge resource can help to the acquisition of richer knowledge. There are critics that tried to downplay Wikipedia's role as a source of valid information and have often pointed to the Encyclopedia Britannica as an example of an accurate reference. *(3.)* Rahmans and Ng's supporting data questions were created automatically by us using a simple heuristic approach which might not be appropriate. Hence, with better supporting data or with a better heuristic approach we might achieve better results. *(4.)* Currently, we are pursuing a new objective: testing the Wikipedia's data richness by detecting more sentences. The *Wikisense* uses different sets of Wikipedia's sentences [10,100...10000] and for each set calls the other components 100 times. Each time it uses randomly selected sentences and one Wikipedia sentence can be used multiple times. Initial results show further improvement, that can enhance the *Wikisense'* capabilities through the acquisition of more Wikipedia's sentences.

Years ago, previous works demonstrated rather convincingly, that in order to do pronoun resolution one had to be able to do everything else and also that once everything else is done pronoun resolution comes *free* and *automatically* [20]. We hope that our work will positively contribute to this task, and through other future extensions and the cooperation with other researchers, maybe one day pronoun resolution will come *free* and *automatically*.

References

[1] Leslie G. Valiant. Knowledge Infusion. In *Proceedings of the 21st National Conference on Artificial Intelligence - Volume 2*, AAAI'06, pages 1546–1551. AAAI Press, 2006.

[2] Loizos Michael and Leslie G Valiant. A First Experimental Demonstration of Massive Knowledge Infusion. In *Principles of Knowledge Representation and Reasoning: Proceedings of the Eleventh International Conference, KR 2008, Sydney, Australia, September 16-19, 2008*, pages 378–389. AAAI Press, 2008.

[3] Loizos Michael. Partial observability and learnability. *Artif. Intell.*, 174(11):639–669, 2010.

[4] Hector J. Levesque. The Winograd Schema Challenge. In *AAAI Spring Symposium: Logical Formalizations of Commonsense Reasoning*, number SS-11-06. American Association for Artificial Intelligence, 2011.

[5] Arpit Sharma. Solving Winograd Schema Challenge: Using Semantic Parsing, Automatic Knowledge Acquisition and Logical Reasoning. Master's thesis, Arizona State University, 2014.

[6] Ackerman Evan. Can Winograd Schemas Replace Turing Test for Defining Human-level AI? *IEEE Spectrum*, 2014.

[7] Altaf Rahman and Vincent Ng. Resolving Complex Cases of Definite Pronouns: The Winograd Schema Challenge. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, EMNLP-CoNLL '12, pages 777–789, Stroudsburg, PA, USA, 2012. Association for Computational Linguistics.

[8] Loizos Michael. Reading Between the Lines. In *Proceedings of the 21st International Jont Conference on Artifical Intelligence*, IJCAI'09, pages 1525–1530, San Francisco, CA, USA, 2009. Morgan Kaufmann Publishers Inc.

[9] Loizos Michael. Machines with Websense. In *Proc. of 11th International Symposium on Logical Formalizations of Commonsense Reasoning (Commonsense 13)*, 2013.

[10] Nicos Isaak. A First Attempt of the Creation Of a Commonsense Conclusion Web Engine (In Greek). Master's thesis, Open University of Cyprus, 2011.

[11] Haoruo Peng, Daniel Khashabi, and Dan Roth. Solving hard coreference problems. *Urbana*, 51:61801.

[12] Tejas Ulhas Budukh. An intelligent co-reference resolver for winograd schema sentences containing resolved semantic entities, 2013.

[13] Arpit Sharma, Nguyen H Vo, Somak Aditya, and Chitta Baral. Towards addressing the winograd schema challenge-building and using a semantic parser and a knowledge hunting module. In *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI*, pages 25–31, 2015.

[14] Marie-Catherine De Marneffe, Bill MacCartney, Christopher D Manning, et al. Generating typed dependency parses from phrase structure parses. In *Proceedings of LREC*, volume 6, pages 449–454, 2006.

[15] Marie-Catherine De Marneffe and Christopher D Manning. The Stanford typed dependencies representation. In *Proceedings of COLING 08 Workshop on Cross- Framework and Cross-Domain Parser Evaluation*, pages 1–8. Association for Computational Linguistics, 2008.

[16] Nathanael Chambers and Daniel Jurafsky. Unsupervised learning of narrative event chains. In *ACL*, volume 94305, pages 789–797. Citeseer, 2008.

[17] Johan Bos and Katja Markert. Recognising Textual Entailment with Logical Inference. In *HLT/EMNLP 2005, Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing, Proceedings of the Conference, 6-8 October 2005, Vancouver, British Columbia, Canada*, pages 628–635. Association for Computational Linguistics, 2005.

[18] Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. The Stanford CoreNLP natural language processing toolkit. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 55–60, 2014.

[19] David Bender. Establishing a human baseline for the winograd schema challenge. In *MAICS*, pages 39–45, 2015.

[20] Jerry R Hobbs. Resolving pronoun references. *Lingua*, 44(4):311–338, 1978.