PRONOUN DISAMBIGUATION: WITH APPLICATION TO THE WINOGRAD SCHEMA CHALLENGE

Martin J Wheatman

Yagadi Ltd, United Kingdom

ABSTRACT

A value-based approach to Natural Language Understanding, in particular, the disambiguation of pronouns, is illustrated with a solution to a typical example from the Winograd Schema Challenge. The worked example uses a language engine, Enguage, to support the articulation of the advocation and fearing of violence. The example illustrates the indexical nature of pronouns, and how their values, their referent objects, change because they are set by contextual data. It must be noted that Enguage is not a suitable candidate for addressing the Winograd Schema Challenge as it is an interactive tool, whereas the Challenge requires a preconfigured, unattended program.

KEYWORDS

Natural Language Understanding, Winograd Schema Challenge, Enguage, Interactive Computation, Peircean Semiotics

1. INTRODUCTION

Natural language is subjective because the meaning of an utterance is dependent not only on the words used, but also on the current state of contextual data. In particular, the value of each pronoun—its referent object—depends on this context. Because of their indexical nature, the effect of context is to flip pronouns between values. There is a naive assumption that this is trivial and deemed to be [one facet] of common sense. However, to create a mechanism to replicate this process is not so straightforward. The Winograd Schema Challenge [1] has been devised to verify this understanding of common sense, presenting many such examples. This paper presents a solution to one such example, and in doing so illustrates these complexities.

The example Challenge has been taken from the WCS Wikipedia page [2] and it is a representative example of this type of deceptively easy conundrum. Given the two sentences:

Councillors refused demonstrators a permit because they advocate violence.
Councillors refused demonstrators a permit because they fear violence.

The system must be able to show:

1) Who are they: who is it that advocates, and who fears, violence; and,
2) that the value of the pronoun they is different in the two cases.

Such challenges are succinct descriptions of utterance set against context. To properly illustrate this contextualisation several assumptions need to be made at the outset.

First, it must be assumed that there are only two groups, demonstrators and councillors, each of which speak with one voice. So that all demonstrators or all councillors either advocate, or fear, violence. Otherwise, because some demonstrators advocate violence does not use they and so, explicitly, gives the game away.

Second, it assumed that the two answers are different, they don't both fear or advocate violence.

Last, the WSC example implies that fear and advocacy are naturally opposed. For the purposes of this paper this is assumed to be true, but in practice this might not be the case.

This paper starts by describing Enguage, the engine which is used to demonstrate natural language understanding and elaborates on its use of pronouns. It goes on to analyse the challenge and its value content. It then describes the concepts required to support a solution, the test which forms part of the Enguage unit test and the output results. Then follows a discussion on how this could be used to develop a generic solution to the WSC.

2. ENGUAGE

EnguageTM is a natural language interpreter but to describe what it is, it is easier to describe what it is not. It is not a chatbot, it is a user interface rather than an attempt to maintain a conversation. Nor is it a front-end to Internet search such as Watson [3] or Siri [4]. The most obvious analogy is with the Skills-based retail device Alexa [5], or the Behaviour Driven Development tool Cucumber [6]; however, with these comes an immediate interpretation of an utterance in JavaScript or Gherkin. The emphasis with Enguage is depth of understanding [7] because it is programmed, itself, in natural language utterances [8, 9, 10]. So, rather than using the syntax-to-semantic mapping of context free languages, or some implementation of a linguistics view of language, it views each utterance as a single value: an extremely large, natural number, represented in letters. Hence, there is no stop-list of discarded words; the whole utterance must be consumed to achieve understanding. Thus, an utterance encompasses not just anything that can be said but any string array. Interpretation is the arbitrary mapping of an utterance onto one of several content dependent machine utterances, or replies. So, '*I need a coffee*' will initially be met with '*ok, you need a coffee'*, but subsequently with '*I know*'.

As such, understanding is highly subjective, so by using simple pattern matching techniques many varied utterances can be combined to use the same interpretation producing a practical, essentially objective, system. So, '*I need a tea*' will use the same interpretation. While it won the 2016 BCS Machine Intelligence Competition, it does not seek *intelligence*: it is not designed to pass the Turing Test [11]; it does not attempt to maintain a conversation. It is simply affects machine state by mapping utterances, and with the application of speech-to-text software, it affords computing machinery interaction by voice. Further, these arbitrary mappings can be programmed by voice, which allows the modification of interpretation—machine behaviour—on-the-fly. As such it could be deemed machine learning, but only in a very broad sense.

Along with a pattern to determine which interpretations apply to an utterance, there is either a direct reply, or one or more internal utterances, or '*thoughts*', to determine—or *mediate*—the appropriate reply. When a reply is reached, *reply X*, interpretation ends and that reply is returned. From within the list of '*thoughts*', there are the options to issue further utterances; to operate machine interaction (including connection to servers across the internet), and; replies to return values to the utterer. Each such interpretation forms a Peircean *Sign*: an utterance—a *Representamen*—refers through contextual data—*Interpretant*—formed of "further more developed Signs", to one of many possible replies—the referent *Object*—see [12]. For example, '*I need a coffee*' can be interpreted by the needs concept [10]:

```
On "SUBJECT needs NUMERIC-QUANTITY PHRASE-OBJECT":
        set output format to "QUANTITY,UNIT of,,LOCATOR
        LOCATION"; QUANTITY OBJECT exists in SUBJECT needs
        list;
        reply "I know";
        if not, add QUANTITY OBJECT to SUBJECT needs
        list; then, reply "ok, SUBJECT needs ... ".
```

These interpretations are gathered together into a cogent repertoire to support a concept, in this case *need* or *needs*. The repertoire uses lowercase words as contextual values (i.e. constant string literals); uppercase words as parameters (i.e. variables.). Having identified variables, it is a short development to substitute these using appropriate pronouns.

When Enguage was originally developed, the issue of pronouns was addressed, as special cases, in the *need+needs* repertoire [10]. A generic, and naïve, solution has since been developed, based on a simple carry-forward, anaphoric, algorithm. A pronoun is associated with a variable, eg *it*↔OBJECT. When that variable is matched, the associated pronoun is set to its value; unless it is the pronoun, in which case the value of the pronoun, if set, is used. Singular pronouns work well, in cases such as '*I need milk*', where it matches *i need OBJECT* and is followed by '*it is from the dairy aisle*'. Where it breaks down is where it would not be considered a reasonable use of natural language: 'I need milk and cheese'. '*It is in a four pint tub*'. This association of pronoun with variable can be achieved through utterance and ultimately by voice.

Enguage is programmed by voice using sixteen, or so, in-built utterances. The original written repertoires, as given above, were interpreted using 12 in-built signs. This self-generating, or *autopoietic*, property is a feature of information systems: the C compiler is built using the C compiler. So, the example Challenge could be programmed in Enguage in the three utterances.

1. *To the phrase the councillors refused the demonstrators a permit because the feared violence think set they to the councillors.*

2. *To the phrase the councillors refused the demonstrators a permit because the advocated violence think set they to the demonstrators.*

3. *To the phrase who are they reply they are variable they.*

These three utterances tell Enguage how to interpret the Challenge utterances: to set a persistent variable, *they*, to the appropriate value. Then the utterance '*who are they*' simply returns the value of the pronoun *they*. However, this is not used as it is too simple for the utterances to be fully understood. For example, it should be possible to ask '*why was it that the councillors refused the demonstrators a permit*'.

The solution given in this paper uses the concepts of why and because: the *because* sign has a *what* and a *why* clause (*cause* and *effect*) [13]. Each of these is an utterance in its own right, so *because* is a generic function, and the *effect* is uttered and if reasonable/successful, or *felicitous*, then the *what* is uttered.

3. CHALLENGE ANALYSIS

This section describes the process by which a schema is developed. As there are several solutions, rather than pulling an answer out of a hat and presenting it as the truth, it is worth illustrating its development. Essentially, the system needs to be informed of contextual information: there is nothing intrinsic to the words *councillors* or *demonstrators*.

3.1. Pronouns

WSC Challenges seem largely based around ascribing values to group pronouns, in particular *they*. In Linguistic terms, *they* is a *deictic* word - one that requires contextual information to ascribe value, and therefore determine meaning. This not only includes the personal pronouns, *he, she, you, they,* and also the word *I*; and the spatial deixis, including *here* and *there*; and, the temporal deixis including *before, after, later, soon, tomorrow, yesterday* etc. In Peircean

Semiotic terms [12], these words come under the *Secondness* of how a *Sign/Representamen* refers to an *Object*, i.e. *Indexical* reference. A Sign may refer to an Object: by quality, i.e. *Iconic*; by pointing to it, *Indexical*; or by convention, Symbolic. So, the value of *they* indicates what the answer is to '*who are they'*.

The special case of *they* is that it is a plural pronoun, so may apply to groups of subjects or objects. In this Challenge, with *the demonstrators* and *the councillors* both being plural, they could refer to either or both, hence the ambiguity. Enguage uses they successfully on *and*- and *or*- lists, such as '*I need milk and eggs. They are from the dairy aisle'*.

3.2. Boolean Versus Mutually Exclusive Options

There is a difference between the binary notion of a Boolean, and two-state information, such as *advocation* or *fear*. A Boolean presents *true* or *false*, one being the logical negation of the other—logically there are only two possible values. So, if *advocation* or *fear* type really is a Boolean type it should be able to be modelled as advocation or not advocation.

However, the states of *fear* and *advocation* can be shown not to be binary opposites. Determining one contradicts the other; so, if demonstrators advocate violence, they do not fear violence. But, by not explicitly advocating violence does not determine fear.

Further, in real world systems—such as with *advocation* or fear—unless the two are true opposites and there is a safe, initial value, a third value is needed: unknown. Edgar Codd recognised this in his development of the Relational data model [14], where even his Boolean was a trinary value of *unknown*, *true* and *false*, for which truth tables were devised. The safe initial value is therefore *unknown*, not *false* as is often typical in information systems. Reflecting on the Boolean type, it can be seen as a single-state system. Given that it is modelled *false* as zero and *true* as non-zero, this single-state system would be *false* or *anything else.*

Furthermore, it is entirely possible that you might not advocate violence, but also not be afraid of it either. A third possible option—an alternative to *fear* or *advocation*—is that of abhorring violence. There will be the need for an *unknown* or *unstated* value. There might also be a *common-sense* value to fall back upon: under normal circumstances a councillor would be a servant of the people; however, there may be a case where a councillor is corrupt and advocates violence.

Moreover, this mutually exclusive choice should be supported in a non-contradictory way. It should not be possible to contradict oneself, for example by consecutively stating:

The demonstrators advocate violence.
The demonstrators fear violence.

Values can change but to reverse the demonstrators advocate violence, one would have to utter:

The demonstrators do not advocate violence.
The demonstrators fear violence.

In practice this verbal reasoning is quite natural. This mutually exclusive choice therefore supports an advocating case; a fearing case; a possible abhorring case; and—in-between each—a return to an unstated case. This step out to an unstated value can be circumvented where the speaker refutes an earlier statement by invoking the Enguage *undo* operation with the pattern *no, ...* :

The demonstrators advocate violence.
No, the demonstrators fear violence.

3.3. Contextual Information

In interpreting the Challenge examples, humans rely on contextual information. The machine has not been given such information, so in giving it, it looks as if we are simply answering our own question. However, this can be seen in other examples. Another example from the Wikipedia page [2], as being easy to determine is that of:

The women stopped taking pills because they were [pregnant/carcinogenic].
Which individuals were [pregnant/carcinogenic]?

In explaining this reasoning: *pills do not get pregnant, women do; women cannot be carcinogenic, but pills can*: this is somehow easier than the councillor example. But, by having to write this we are showing that this information is not known without expression but rather is known through expression. While this may not have been said to the thinker, the components of it have, or at least been thought by the thinker. There is a need to provide the answer in a reasoned manner.

It appears that *they* is set as a variable, using the *set/unset/get* repertoire. All the engine has to do is to set *they* to the appropriate value and return that value to support the utterance: who are they. There is also the need to think using the pronoun they. This is: *they X*. A naive interaction may be as follows.

```
=> Set they to the demonstrators.
Ok, they is set to the demonstrators.
=> The councillors refused the demonstrators a permit because
   they advocate violence.
Ok, the councillors refused the demonstrators a permit because
   the demonstrators advocate violence.
=> Who are they.
They are the demonstrators.

=> Set they to the councillors.
Ok, they is set to the councillors.
=> The councillors refused the demonstrators a permit because
   they fear violence
ok, the councillors refused the demonstrators a permit because
   the councillors fear violence.
```

```
=> Who are they.
They are the councillors.
```

This works, in that the question '*who are they*' is correct in both cases, and the value in the unequivocal reply is correctly expanded in both cases; however, this is not what the WSC is seeking because *they* needs to be set each time and the Challenge statement seems to play no role in the example. We need to be able to *configure* who fears or advocates violence: contextual data which does not change, or changes over a long timescale.

So, common sense needs to be voiced. How the value is obtained, whether by opinion or statistical fact, is immaterial. The mechanical approach is that nothing can be assumed without being said. One model solution is as follows. This example may seem a little sparse although much elaboration can be applied, such as whether councillors are unreasonable in their belief in the violent intentions of the demonstrators.

```
=> The demonstrators advocate violence.
=> The councillors fear violence.

=> The councillors refused the demonstrators a permit because
   they advocate violence.
=> Who are they.
They are the demonstrators.

=> The councillors refused the demonstrators a permit because
   they fear violence.

=> Who are they.
They are the councillors.
```

This makes explicit the assumption that councillors fear violence and the demonstrators advocate violence. These values can be seen as either a naive assumption or as common sense. This approach also allows for the case where the demonstrators fear violence, and the councillors advocate violence in some dystopian scenario. This implementation could be augmented with the ability to define which values can be ascribed to councillors and demonstrators, *the councillors may fear violence*, but this is beyond the scope of this paper.

The problem with defining what happens with stating what can happen, is that it leaves many unstated values. This is a much more interesting and useful mechanism, as it allows what is the otherwise unthinkable: demonstrators who may, or may not, be peaceful; councillors who are corrupt and might advocate violence. Now, swapping fear and advocacy changes who they are, but in doing so this also raises some unobtainable cases.

This still leaves many outcomes, whereas, to an English speaker there are only one or possibly two outcomes. What has been ignored so far is the what clause, which, possibly, provides an answer to who *they* are. They fear/advocate violence picks up the value of the pronoun *they*, set elsewhere as contextual data; however, that contextual data could be partially defined by the clause which does not change between the utterances.

3.4. Because and Implies

In uttering *The councillors refused the demonstrators a permit because they advocate/fear violence*, Enguage recognises this as a *because* utterance [13] and first thinks of the cause (*they fear/advocate violence*.) If this is felicitous, it then thinks what the effect is: *the councillors refused the demonstrators a permit*, which indicates the action of the councillors but it would seem this has little bearing on this Challenge, because the same *effect* is used in both example utterances. This is reflected, initially at least, in the sign simply being felicitous and the reply being *ok*.

While the *effect* utterance within the *because* concept [13] is the same in both cases, it is in the public interest that councillors prevent dangerous events occurring, so their refusal to permit something means they are acting in the public good.

If the councillors were corrupt and wanted to encourage violence, refusing a permit would not achieve this end: if the demonstrators advocated violence, they should *grant* a permit; conversely, if the demonstrators feared violence, refusing a permit wouldn't force them into being violent at a protest either. Refusing a peaceful demonstration does not act against public good (other than to prevent lawful protest). So, while it may be a possibility that councillors may be corrupt in some dystopian scenario, it is unlikely in this case.

Therefore, if the councillors are refusing a demonstration that they—rightly or wrongly—fear violence, and we can add that thought at runtime as an *inductive* utterance:

> *The councillors refused the demonstrators a permit implies the councillors fear violence.*

It is still possible that the demonstrators fear violence—they may simply have not persuaded the councillors of that fact. However, for the purposes of this test this is not the case.

> *The councillors refused the demonstrators a permit implies the demonstrators advocate violence.*

These two utterances automate the contextual information, by injecting the appropriate thoughts into the *refused* sign. For the moment this is not implemented in the test and is left for further work

4. THE CONCEPT OF VIOLENCE

The main concept required for this paper, which articulates the advocation and fear of violence, is now defined. These are the interpretations of the utterances which contain the word violence: the patterns supported, what possible replies can be made, and how those replies are determined. An unequivocal reply is necessary to ensure the user has confidence that their intention has been enacted. The completed repertoire, violence.txt, which is loaded when there is the word *violence* in the utterance, and the Enguage unit test source code in 4.2, are all available for inspection and to run as open source on GitHub [10].

4.1. Concept

In reading this concept, it must be reiterated that interpretation will stop when a *reply* is reached. Enguage also maintains an internal status of *felicity*: how happy an utterance is. For example in the last interpretation of this listing, if the utterance is '*get the value of they*' and either this cannot be interpreted or *they* is unset, the status will be infelicitous and the following reply of '*I don't know*' will be returned to the calling utterance or the user. If a value is returned, the final reply is reached and the ellipsis is replaced with the returned value, i.e. the value of *they*.

```
# fear and advocacy - alternative
states ### ADVOCATE
On "PHRASE-WHOM advocate
      violence": # check for a
      contradiction...
      WHOM exists in violence fear
      list; reply "no, WHOM fear
      violence";
      if not, add WHOM to violence advocate
      list; reply "ok, WHOM advocate
      violence".

On "PHRASE-WHOM do not advocate
      violence": WHOM exists in
      violence fear list; reply "i
      know";
      if not, remove WHOM from violence advocate
      list; reply "ok, WHOM do not advocate
      violence".

# common sense ADVOCATE
On "common sense would suggest PHRASE-WHOM advocate
      violence": # check for a contradiction...
      WHOM exists in csviolence fear list;
      reply "no, common sense would suggest WHOM fear
      violence"; if not, add WHOM to csviolence advocate
      list;
      reply "ok, common sense would suggest WHOM advocate violence".

On "common sense would suggest PHRASE-WHOM do not advocate
      violence": WHOM exists in csviolence fear list;
      reply "i know";
      if not, remove WHOM from csviolence advocate list;
      reply "ok, common sense would suggest WHOM do not
advocate violence".

### FEAR
On "PHRASE-WHOM fear violence":
      # check for a contradiction...
      WHOM exists in violence advocate
      list; reply "no, WHOM advocate
      violence";
      if not, add WHOM to violence fear
      list; reply "ok, WHOM fear
      violence".

On "PHRASE-WHOM do not fear violence":
      WHOM exists in violence advocate
      list; reply "i know";
      if not, remove WHOM from violence fear
      list; if not, reply "no, they do not";
      reply "ok, WHOM do not fear violence".

# common sense FEAR
On "common sense would suggest PHRASE-WHOM fear
      violence": # check for a contradiction...
      WHOM exists in csviolence advocate list;
      reply "no, common sense would suggest WHOM advocate
      violence"; if not, add WHOM to csviolence fear list;
      reply "ok, common sense would suggest WHOM fear violence".

On "common sense would suggest PHRASE-WHOM do not fear
      violence": WHOM exists in csviolence advocate list;
      reply "i know";
      if not, remove WHOM from csviolence fear list;
      if not, reply "no, common sense would suggest they do
      not"; reply "ok, common sense would suggest WHOM do not
      fear
violence".

### Building the test...
On "the councillors refused the demonstrators a
      permit", reply "ok".

On "they STATE violence":
      get violence STATE list;
      if not, get csviolence STATE
      list; set the value of they to
      ... ; reply "ok, they are
      ...";
      if not, unset the value of
      they; reply "i don't know".

On "who are they":
      get the value of they;
      if not, reply "i don't
      know"; reply "they are
      ...".
```

4.2. Test

This test code is taken from the Engage unit test, which is accumulating various supported examples of utterance. Currently this stands at over 340 tests and these are executed after evaluating:

```
     if (runThisTest( "WSC - advocacy and fear" )) {
```

which then outputs the given message.

A method, `tidyUpViolenceTest`(), is called after each test section, to reinitialise the test environment. Again, this is called using Enguage to interact with the test environment:

```
static private void tidyUpViolenceTest( String fname ) {
     Enguage.mediate(  new  Strings("delete  "+  fname  +"  advocate
     list")); Enguage.mediate( new Strings("delete "+ fname +" fear
     list"));  Enguage.mediate(  new  Strings("unset  the  value  of
     they"));
}
```

The static method mediate takes up to three parameters: the first is the utterance to be interpreted; the second and third are optional replies. If only one parameter is given, the reply is ignored. If two or three are given they are checked, for example the reply might be one of two in the case where Enguage is tested against a third party server, which may or may not be running. The method is called mediate as the intention is to find the most appropriate interpretation of an utterance, given that an utterance may well be ambiguous.
This first test is to show that the values of advocate/fear and do not advocate/fear can be recorded against demonstrator but once set they need to be unset so they can be re-set to another value, i.e. the set value cannot be contradicted.

```
audit.subtl("Contradiction test... can't swap between
states directly");
mediate( "demonstrators fear violence",
         "ok, demonstrators fear
violence" ); mediate( "demonstrators
advocate violence",
         "no, demonstrators fear violence"
); mediate( "demonstrators do not fear
violence",
         "ok, demonstrators don't fear
violence" ); mediate( "demonstrators advocate
violence",
         "ok, demonstrators advocate
violence" ); mediate( "demonstrators fear
violence",
         "no, demonstrators advocate
violence" ); mediate( "demonstrators don't
advocate violence",
         "ok, demonstrators don't advocate violence" );
// tidy up

tidyUpViolenceTest();
```

This next test is where the states of both the councillors and the demonstrators are set to the ‘correct’ opposing views

```
audit.subtl( "Openly stated: opposing
views" ); mediate( "the councillors fear
                           violence",
         "ok, the councillors    fear
violence" ); mediate( "the demonstrators
advocate violence",
         "ok, the demonstrators advocate violence" );
// test 1
mediate( "the councillors refused the demonstrators a
permit because they fear violence",
         "ok, the councillors refused the demonstrators a
permit because they fear violence" );
mediate( "who are they",
         "they are the councillors" );
// test 2
mediate( "the councillors refused the demonstrators a
permit because they advocate violence",
         "ok, the councillors refused the demonstrators a
permit because they advocate violence" );
mediate( "who are they",
         "they are the demonstrators" );
// tidy up

tidyUpViolenceTest();
```

This next section is where the states of both the councillors and the demonstrators are aligned to the same view: advocacy. In the test labelled 1, neither the demonstrators nor the councillors are said to *fear violence* so the reply *I don’t know* is expected from the utterance *who are they*. In the test labelled 2, they both advocate violence so they both make up the value of *they*.

```
        audit.subtl( "Openly stated: aligned views -
        advocate" ); mediate( "the councillors   advocate
        violence",
                 "ok, the councillors  advocate
        violence" ); mediate( "the demonstrators advocate
        violence",
                 "ok, the demonstrators advocate violence" );

        // test 1
        mediate( "the councillors refused the demonstrators a
        permit because they fear violence",
                 "I don't think they fear
        violence" ); mediate( "who are they",
                 "I don't know" );

        // test 2
        mediate( "the councillors refused the demonstrators a
        permit because they advocate violence",
                 "ok, the councillors refused the demonstrators a
        permit because they advocate violence" );
        mediate( "who are they",
                 "they are the councillors , and the demonstrators" );

// tidy up
tidyUpViolenceTest();
```

This next section is the opposite aligned case, *fear*. Strayign slightly from the Challenge, this test introduces the notion of the voters, simply to show that this code is robust, and that it is beginning to experiment with different scenarios.

```
        audit.subtl( "Openly stated: aligned views - fear" );
        mediate( "the councillors fear violence because the voters
        fear violence",
                 "ok, the councillors fear violence because the
        voters fear violence" );
        mediate( "the demonstrators fear violence",
                 "ok, the demonstrators fear violence" );

        // test 1
        mediate( "the councillors refused the demonstrators a
        permit because they fear violence",
                 "ok, the councillors refused the demonstrators a
        permit because they fear violence" );
        mediate( "who are they",
                 "they are the voters, the councillors , and
        the demonstrators" );

        // test 2
        mediate( "the councillors refused the demonstrators a
        permit because they advocate violence",
                 "I don't think they advocate
        violence" ); mediate( "who are they",

                 "i don't know" );

// tidy up
tidyUpViolenceTest();
```

This last test shows that the confguration can be worded around the idea of common sense, so that values can be recorded well ahead of time and special values (i.e. non-common sense values) can overwrite these at runtime.

```
        audit.subtl( "the common sense view" );
        // This should be configured within the repertoire...
        mediate( "common sense would suggest the councillors
        fear violence",
                 "ok, common sense would suggest the councillors
        fear violence" );

        mediate( "the councillors refused the demonstrators a
        permit because they fear violence",
                 "ok, the councillors refused the demonstrators a
        permit because they fear violence" );
        mediate( "who are they",
                 "they are the councillors" );

// tidy up - N.B. these will affect previous tests on re-runs
tidyUpViolenceTest("csviolence");
```

4.3. Results

These results here are simply cut and pasted from the standard Enguage unit test run [10].

```
        Test 1: WSC - advocacy and fear
        ===============================

        Contradiction test... can't swap between states directly
        ++++++++++++++++++++++++++++++++++++++++++++++++++++++++
        user> demonstrators fear violence enguage> ok ,
        demonstrators fear violence

        user> demonstrators advocate violence enguage> no ,
        demonstrators fear violence

        user> demonstrators do not fear violence enguage> ok ,
        demonstrators don't fear violence

        user> demonstrators advocate violence enguage> ok ,
        demonstrators advocate violence

        user> demonstrators fear violence
        enguage> no , demonstrators advocate violence

        user> demonstrators don't advocate violence enguage> ok
        , demonstrators don't advocate violence


        Openly stated: opposing views
        +++++++++++++++++++++++++++++
        user> the councillors  fear  violence enguage> ok , the
        councillors fear violence


        user> the demonstrators advocate violence enguage> ok ,
        the demonstrators advocate violence
```

```
user> the councillors refused the demonstrators a permit
because they fear violence
enguage> ok , the councillors refused the demonstrators
a permit because they fear violence

user> who are they
enguage> they are the councillors

user> the councillors refused the demonstrators a permit
because they advocate violence
enguage> ok , the councillors refused the demonstrators
a permit because they advocate violence

user> who are they
enguage> they are the demonstrators


Openly stated: aligned views - advocate
++++++++++++++++++++++++++++++++++++++++
user> the councillors  advocate violence enguage> ok ,
the councillors advocate violence

user> the demonstrators advocate violence enguage> ok ,
the demonstrators advocate violence

user> the councillors refused the demonstrators a permit
because they fear violence
enguage> i don't think they fear violence

user> who are they enguage> i don't know

user> the councillors refused the demonstrators a permit
because they advocate violence
enguage> ok , the councillors refused the demonstrators
a permit because they advocate violence

user> who are they
enguage> they are the councillors , and the
demonstrators


Openly stated: aligned views - fear
++++++++++++++++++++++++++++++++++++
user> the councillors fear violence because the voters
fear violence enguage> ok , the councillors fear
violence because the voters fear violence

user> the demonstrators fear violence enguage> ok , the
demonstrators fear violence

user> the councillors refused the demonstrators a permit
because they fear violence
enguage> ok , the councillors refused the demonstrators
a permit because they fear violence

user> who are they
enguage> they are the voters , the councillors , and the
demonstrators
```

```
user> the councillors refused the demonstrators a permit
because they advocate violence
enguage> i don't think they advocate violence

user> who are they enguage> i don't know

the common sense view
++++++++++++++++++++
user> common sense would suggest the councillors fear
violence enguage> ok , common sense would suggest the
councillors fear violence

user> the councillors refused the demonstrators a permit
because they fear violence
enguage> ok , the councillors refused the demonstrators
a permit because they fear violence

user> who are they
enguage> they are the councillors
```

5. DISCUSSION

Traditional approaches to machine understanding of language stem from the dualist model introduced by Ferdinand de Saussure in the early C20th [15]. A *Signifier* (written or spoken word) has a bi-directional link to a *Signified* mental image—the word conjures the image, and vice versa. This *Structuralism* was prevalent and pervaded analysis until the 1970s when philosophy and technologies, such as Object Orientation [16], allowed the Empirical notion that the real world is what is important, not the processes by which it is analysed. Enguage is, in this tradition, allowing language to model itself rather than resorting to syntax-trees with logical schemas, such as First Order Predicate Calculus, representing semantics.

The difference between speaking and listening, reading and writing must be highlighted. Speech is an innate human ability, whereas writing is a technology, a learnt skill. The two are different in their media, and in their use of language. The work of Peter B. Andersen [17] shows that speech is often unstructured, and can consist of grunts, in an effort, as Austin puts it, to do things with words [18]. Writing is more considered, worked and refined, and seen as an academic pursuit, along with the ease at which it can be studied up until recording became common place. This is presumably why Saussure chose to concentrate on the written word. Even today, linguists work on transcripts of the spoken word to produce, for example, concordance studies [19].

It must also be noted that Enguage is not a suitable tool to address the full Challenge as the user is not allowed to interact with the program under test. According to the Common-sense Reasoning web page, a file of answers must be output (eg A, A, B, B, etc) to ease marking [1]. Enguage is an interactive machine interface, so that the user would be unable to contribute to the machine's learning. The efficacy of this interaction will outweigh this restriction.

Clearly, the analysis in this paper looks at the representation of values within the example Challenge because Enguage sees natural language as a value-based, rather than a truth-based, medium. The link between programming language and natural language may be closer than one might think. It is no coincidence that such a Sign looks very much like the context-free function

of a programming language. So, the above example of the sign to process the utterance '*I need a coffee*' may be represented as:

```
function needs( string SUBJECT, int QUANTITY, string[]
  OBJECT) { format := "QUANTITY,UNIT of,,LOCATOR LOCATION";
  if (existsInNeedsList( QUANTITY, OBJECT,
   SUBJECT )) return "i know";
  else {
   ANSWER := addToNeedList( QUANTITY, OBJECT );
   return "ok, SUBJECT needs ANSWER";
  }
}
```

The *effect* because cause utterances can easily be transformed into *cause so effect* utterances [10]; however, this may be no easier to solve.

They advocated violence so the councillors refused the demonstrators a permit.
They feared violence so the councillors refused the demonstrators a permit.

This is still a problem if a naive pronoun matching algorithm chooses the last noun phrase; but it hints at it being the first noun phrase in the utterance being set/swapped with the pronoun as an improvement. This is left for further work.

Enguage was initially designed with written repertoires. It was soon seen that this was limiting because the user would have to resort to writing more program to develop the interface. The vocal mechanisms required to do this programming are naturally wrapped up in utterances, such as *PHRASE-X implies PHRASE-Y*. Furthermore, the simple algorithm used to process utterances allows more complex language defining repertoires, such as *want is like need*, in which utterances containing *want* or *wants* are processed like those with *need* or *needs* in them. Thus, a vocal schema can be used to develop repertoires interactively. This now includes the ability to save successful repertoires. Because Enguage is an interface to software, it could be that repertoires are produces to create a file which is appended with A or B on successful interpretation. For this author, passing the Winograd Schema Challenge is secondary to producing a compelling, generic, vocal solution.

6. CONCLUSIONS

A solution to one example of the Winograd Schema Challenge is presented. The model provided by Enguage demonstrates linguistic reasoning in the why+because repertoire [13]. The processing of language can be performed without resulting to the structural syntax-semantics-pragmatics model devised in the 1930s [20].

While the WSC presents many such cases, the solution as provided is not ready to address the WSC in full, as the repertoire is still written. This needs to be implemented vocally, to address the challenge interactively. As it stands, this interactivity prevents Enguage from officially competing in the full Challenge. A generic, interactive, vocal solution will be an interesting experiment and is the next goal for this project.

ACKNOWLEDGEMENTS

The author would like to thank: Professor E Davis for clarification on the goals of the Winograd Schema Challenge; and Andrew Morris for proof reading the text.

REFERENCES

[1] Levesque, H., Davis, E., Morganstern, L. (2019), see http://commonsensereasoning.org/ winograd.html (Retrieved 25th Sept 2019)

[2] Wikipedia (2019), https://en.wikipedia.org/wiki/Winograd_Schema_Challenge, (Retrieved 25th Sept 2019)

[3] IBM (2019) https://www.ibm.com/watson/how-to-build-a-chatbot retrieved 3rd Oct 2019

[4] Apple (2019) https://www.apple.com/uk/siri/ retrieved 3rd Oct 2019

[5] Amazon (2019) https://developer.amazon.com/en-US/alexa/alexa-skills-kit retr., 3rd Oct 2019

[6] Cucumber (2019) https://cucumber.io/docs, retrieved 3rd Oct, 2019

[7] Wheatman, M. J. (2019) Building Conversational Interfaces, ITNOW, Volume 61, Issue 1, Spring 2019, Pages 48–49, https://doi.org/10.1093/itnow/bwz020

[8] Wheatman, M. J. (2014). An Autopoietic Repertoire. In: Bramer, M., Petridis, M. (Eds.), Research and Development in Intelligent Systems XXXI: Proceedings of the 34th SGAI International Conference on Innovative Techniques and Applications of Artificial Intelligence (pp 165-170). Cambridge, UK: Springer. doi:10.1007/978-3-319-12069-0

[9] Wheatman, M. J. (2018) Unifying Speech and Computation, In Liu K., Nakata K., Li W., Baranauskas C. (eds) Digitalisation, Innovation, and Transformation, ICISO 2018. IFIP Advances in Information and Communication Technology, Vol 527, Springer, pp 167-176

[10] Wheatman, M. J.(2019), https://github.com/martinwheatman/Enguage.jar, retrieved Oct., 3rd

[11] Loebner, H. G. (1994) In Response, Communications of the ACM, Vol. 37 Issue 6, 37(6) 1994

[12] Peirce, C. S. (1955) Logic as Semiotic: The Theory of Signs, Philosophical Writings of Peirce, Ed., J. Buchler, Dover Publications, New York, Pp 98-100

[13] Wheatman, M. J. (2018) On Because and Why: Reasoning with Natural Language International Journal of Conceptual Structures and Smart Applications, Vol. 6, Issue 2, July-Dec 2018, DOI: 10.4018/IJCSSA.2018070101

[14] Codd, E. F. (1970). A Relational Model of Data for Large Shared Data Banks. Communications of the ACM, 13(6), 377–387, DOI:10.1145/362384.362685d.

[15] Saussure, F. de (1983) A Course in General Linguistics (C. Bally & A. Sechehaye, Eds., R. Harris, Trans.). London: Duckworth. (Original work published 1916).

[16] Palme, J. (1970) SIMULA 67: An advanced programming and simulation language, Norwegian Computing Centre Publication.

[17] Andersen, P. B. (1990) A Theory of Computer Semiotics. Cambridge: Cambridge University Press.

[18] Austin, J. L. (1962) How to Do Things with Words. (Eds.). Oxford: Oxford University Press.

[19] Smith, N. (2019) https://www.lancaster.ac.uk/fss/courses/ling/corpus/blue/clc_top.htm Retrieved 3rd October, 2019.

[20] Morris, C. W. (1938) Foundations of the Theory of Signs, Encyclopaedia of Unified Science, 1(2), University of Chicago, Chicago.

AUTHOR

Dr. Martin J Wheatman

Dr Wheatman attained a degree in Computer Science from the Hatfield Polytechnic in 1986. Following the development of an Object Management System (MPhil, Lancaster 1999), he studied arbitrary text transformations in Software Engineering (PhD, Reading 2011). From this he developed a natural language engine, EnguageTM, which won the 2016 BCS Machine Intelligence Competition