A Computational Effective Document Semantic Representation

Robert Williams

School of Information Systems, Curtin University of Technology, Perth, Western Australia

e-mail: bob.williams@cbs.curtin.edu.au

Abstract— **A technique based on Noun Phrase and Verb Clause slot structures is described for representing the semantics of the sentences making up a text document. Thesaurus head word index numbers are placed in the appropriate document sentence clause slots to represent the meta level meaning of the sentences. Many different expressions of the same document content can thus be represented by one semantic representation. An implementation of such a technique is described, and sample output is presented. The document summarisation thus produced is suitable for manipulation by computers for a variety of document processing tasks. The technique has primarily been developed for an Automated Essay Grading system, where a robust context free representation of documents is required.**

Index Terms—**document semantic representation, thesaurus, meta level meaning, document summarisation, automated essay grading.**

I. Introduction

Representing the semantics of a text document for computational uses is problematic. How can we formally code the meanings of words, phrases, sentences, paragraphs and so on, so that useful computational work can be done robustly, effectively and efficiently? The computational work may involve text understanding, question answering, or essay grading to name a few possible applications.

In this article we discuss one such technique that is being used in a system being developed for Automated Essay Grading (AEG).

The technique allows a formal representation of free unseen text to be quickly and robustly built for further analysis by the AEG system.

II. Contemporary Semantic Representation

Poesio [6] discusses some current techniques for representing the meaning of sentences, including First-Order Logic and Semantic Networks. First-Order Logic uses mathematical expressions representing set membership relations for objects in the sentence belonging to sets to represent the meaning of a sentence. Computationally, this is very difficult to use for unlimited unseen text, as generally domain specific information needs to be hand coded. Semantic Networks use classifications of objects into a network of relationships, and arc traversal of the nodes can be used to imply relationships amongst the nodes. Again, substantial domain specific knowledge needs to be hand coded prior to their use.

There is a need for a semantic representation that does not need substantial hand coding of knowledge structures prior to use, and that can deal with unlimited unseen text.

III. Practical Limitations of Context Free Phrase Structure parsers for Preliminary Processing

Many Natural Language Processing (NLP) systems use some kind of a parser to initially extract the syntax of sentences in a document as an initial step prior to further processing. Semantic analysis then follows. The use of Context Free Phrase Structure Grammar (CFPSG) parsers is commonly suggested in the literature. They require an extensive set of grammar rules which define legitimate syntax structures. However it is virtually impossible to build a set of grammar rules for free unseen text in practice, because thousands of grammar rules are typically required, and over-generation of possible parse trees results. Increases in parsing time become exponential as the parse trees proliferate. So context free CFPSG parsing cannot be used in all but simple toy domains.

IV. Chunking as an Alternative to Full Parsing

Useful preliminary linguistic computation can be done with less structured parsing. Phrase chunking can be an effective alternative to full parsing at the initial processing stage. Chunking has the advantage that it does not require an extensive set of grammar rules – a few simple rules suffice. Specifically chunking breaks a sentence into syntactically structured components representing noun clauses or phrases and verb clauses or phrases. Often, these structures are sufficient as a preliminary to further processing.

The technique outlined in this paper uses chunking to extract noun phrase and verb clause structures for further processing.

V. Structures to Hold Chunk Semantic Details

The technique described in this paper makes use of chunking to get the structure of sentences in terms of subject and predicate, as represented by Noun Phrases (NP) and Verb Phrases (VP). Generally the NP nominates the subject of discussion, and the VP the actions being performed on or by the subject. However VPs are notoriously complex to deal with in comparison to NPs, because they typically can have many clusters of a Verb Clause (VC) and a NP together. It is far easier to identify VCs instead of the complex VPs. The basis of the technique used is to represent the meaning of the words making up the NPs and VCs in a sequence of structured slots containing a numerical value representing the thesaurus index number for the root meaning of the word in the slot. A numerical summary of the meaning of the sentences in the document being considered is thus built up.

The exact structure of the NP and VC slots is discussed further below, but to illustrate the concept and to give a practical example, consider the following. A typical sentence would comprise alternating NPs and VCs as follows. A typical first NP slot word and numerical contents would be

DET ADJ ADJ N
The small black dog
100 143 97 678

A typical first VC slot word and numerical contents would be

V ADV ADV
walked slowly down
34 987 67

A typical concluding NP slot word and numerical contents would be

DET N
the street
100 234

where the numbers are the thesaurus index numbers for the corresponding words. The numbers here are fictitious, for illustration purposes only. A sentence generally consists of groups of alternating NPs and VCs, not necessarily in that order, so a sentence summary would be represented by a group of NP slots and VC slots containing numerical thesaurus indices. A document summary would then consist of a collection of these groups. Note that a sentence does not have to start with a NP, but can start equally well with a VP.

A. Proposed NP Structure

Martha Kolln [2] on page 433 states a rule for defining an NP under transformational grammar as follows

(1) NP = (DET) + (ADJ) + N +(PREP PHR) + (S)

and on page 429 a Prep Phr as follows

PREP PHR = PREP + NP

When considering the slots to be provided for a NP, (1) above can now be rewritten as

(2) NP = DET ADJ N PREP NP S

The basic component of an NP appears to be

(3) NP = DET ADJ N and some appended structures. It has been found in practice that

(4) NP = DET ADJ ADJ ADJ N

to be a better structure. If we take this as a basic core structure in a NP, the complete NP structure can be built in terms of this core structure by linking multiple occurrences of this core structure by PREPs. It has been found in practice that we should also allow linking by CONJs. So finally we conclude that the basic component should be

(5) NP = CONJ PREP : DET ADJ ADJ ADJ N

where the 2 slots before the colon are the linking slots, and those following the content slots. Practice indicates that we should allow about 40 occurrences of this basic component as the NP slot template should handle many practical NPs encountered in general English text. So a 40x7 array with the following structure will be needed in the program. Fig. 1 shows the first 10 rows of this array.

CONJ	PREP	DET	ADJ	ADJ	ADJ	N

Fig. 1. Noun Phrase Semantic Structure

The first core component in the sentence generally will have the CONJ and PREP slots set to blank (in fact the number 0). Any empty slots will likewise be set to 0.

B. Proposed VC Structure

Martha Kolln [2] on page 428 states a rule for defining a VP under transformational grammar as follows

(6) VP = AUX + V + (COMP) + (ADV)

COMP is explained as an NP or ADJ, so by removing this from the VP we end up with a VC as follows

(7) VC = AUX + V + ADV

It has been found in practice that if we modify this VC definition by the addition of extra AUXs and ADVs we obtain a more useful structure as

(8) VC = AUX AUX ADV ADV V AUX AUX ADV ADV

VCs can often be introduced with CONJs, and it has been found in practice that we should also allow PREPs in a VC, so a complete VC definition would be

(9) VC = CONJ PREP AUX AUX ADV ADV V AUX AUX ADV ADV

If we allow for 40 occurrences of this basic VC component to handle VCs encountered in practice, we will need the following 40x11 array structure in the program. Fig. 2 shows the first 10 rows of this array.

CONJ	PREP	AUX	AUX	ADV	ADV	V	AUX	AUX	ADV	ADV

Fig. 2. Verb Clause Semantic Structure

If a sentence happens to start with a VC, then the CONJ slot will be set to blank (in fact the number 0). Any empty slots will likewise be set to 0.

VI. Algorithms for Building a Chunked Semantic Representation

It is all well and good to postulate a theoretical model of semantic representation, but can it be implemented in a practical way? The answer is yes, and details of the author's implementation of the concepts are discussed below.

The following algorithm describes the process.

For each sentence in the document
Tag each word with POS
Convert POS tags to standard format
Stem each word
For each word and/or stem
Extract thesaurus indices and POS (many)
Determine within context thesaurus index and POS (one)
Store thesaurus index, POS, Word
Chunk sentence
Store chunks in NC and VC slots
End

The following sentence will be used as an example in the explanation that follows. It has been chosen for its relative complexity, to show that the system can handle more than trivial sentences.

"For example if people working on a group project did work their own way and on their own schedule it would be extremely difficult to coordinate their work and assure the quality and timeliness of the end product."

(Source: [1])

A. Tag Each Word with POS

As with many NLP systems, we start with Part of Speech (POS) tagging of the words in the sentence, one sentence at a time. This allows the system to have a preliminary understanding of the words in the sentence it will be dealing with. The Qtag tagger from Mason [3] is currently used. It produces

<w pos="IN">For</w>
<w pos="RB22">example</w>
<w pos="CS">if</w>
<w pos="NN">people</w>
<w pos="VBG">working</w>
<w pos="IN">on</w>
<w pos="DT">a</w>
<w pos="NN">group</w>
<w pos="NN">project</w>
<w pos="DOD">did</w>
<w pos="NN">work</w>
<w pos="PP$">their</w>
<w pos="DT">own</w>
<w pos="NN">way</w>
<w pos="CC">and</w>
<w pos="IN">on</w>
<w pos="PP$">their</w>
<w pos="DT">own</w>
<w pos="NN">schedule</w>
<w pos="PP">it</w>
<w pos="MD">would</w>
<w pos="BE">be</w>
<w pos="RB">extremely</w>
<w pos="JJ">difficult</w>
<w pos="IN">to</w>
<w pos="VB">coordinate</w>
<w pos="PP$">their</w>
<w pos="VB">work</w>
<w pos="CC">and</w>
<w pos="VB">assure</w>
<w pos="DT">the</w>
<w pos="NN">quality</w>
<w pos="CC">and</w>
<w pos="NN">timeliness</w>
<w pos="IN">of</w>
<w pos="DT">the</w>
<w pos="NN">end</w>
<w pos="NN">product</w>

B. Convert POS Tags to a Standard Format

To reduce the number of tags the system has to deal with, the numerous tags produced by Qtag are reduced to a standard set, which eases the computational load in later processing. The system changes the above tag information to the following.

P For
N example
CONJ if
N people
V working
P on
DET a
N group
N project
AUX did
N work
N their
DET own
N way
CONJ and
P on
N their
DET own
N schedule
N it
AUX would
AUX be
ADV extremely
ADJ difficult
P to
V coordinate
N their
V work
CONJ and
V assure
DET the
N quality
CONJ and
N timeliness
P of
DET the
N end
ADJ product

C. Stem each Word

The words produced above will be input to a database containing an electronic version of a thesaurus to attempt to find a head word index number. Additional words, particularly conjunctions and prepositions have been added to this document to rectify their omission from a standard thesaurus. These are represented by index numbers over 1000. If the word cannot be found in the thesaurus, the word's stem is input in attempt to find the word's base form. Many words, such as 'working' do not appear in the thesaurus, but its stem 'work' does. In this case we use the thesaurus index number for 'work' instead of 'working', without losing substantial meaning of the word.

The stemming program used is an implementation of the Porter stemming algorithm documented in [5]. It produces the following output.

for
exampl
if
peopl
work
on
a
group
project
did
work
their
own
wai
and
on
their
own
schedul
it
would
be
extrem
difficult
to
coordin
their
work
and
assur
the
qualiti
and
timeli
of
the
end
product

D. For each Word and/or Stem

a) Extract Thesaurus Indices and POS (many)

We now extract the POS and head word index numbers from the thesaurus. Only the POS that matches the POS for the input word is output. This process produces the following output. Notice that many words have multiple entries. Eg 'working'. An index number of 8888 indicates that an entry could not be found for the word in the thesaurus.

3027 P For
22 N example
4012 CONJ if
997 N people
677 V working
680 V working
686 V working
3034 P on
2000 DET a
712 N group
8888 N project
5008 AUX did
154 N work
170 N work
415 N work

593 N work
625 N work
686 N work
2008 N their
8888 DET own
26 N way
180 N way
627 N way
4003 CONJ and
3034 P on
2008 N their
8888 DET own
86 N schedule
8888 N it
5032 AUX would
5002 AUX be
8888 ADV extremely
868 ADJ difficult
3046 P to
60 V coordinate
2008 N their
677 V work
680 V work
686 V work
4003 CONJ and
858 V assure
2007 DET the
5 N quality
157 N quality
812 N quality
875 N quality
4003 CONJ and
8888 N timeliness
3032 P of
2007 DET the
620 N end
8888 ADJ product

E. For each Word and/or Stem

b) Determine within Context Thesaurus Index and POS (one). Store Thesaurus Index, POS, Word

As can be seen, many words have multiple entries in the output above. This process now selects the most appropriate entry by using a 'within context' algorithm. The entry chosen is the one which makes the most sense in the context of the other words in the sentence. This is done by using broader groups of word categories that are indicated in the related words of the thesaurus classification.

These processes produce the following output. Notice that 'working' now has only one entry.

3027 P For
22 N example
4012 CONJ if
997 N people
677 V working
3034 P on
2000 DET a
712 N group
8888 N project
5008 AUX did
154 N work
2008 N their
8888 DET own
26 N way
4003 CONJ and
3034 P on
2008 N their
8888 DET own
86 N schedule
8888 N it
5032 AUX would
5002 AUX be
8888 ADV extremely
868 ADJ difficult
3046 P to
60 V coordinate
2008 N their
677 V work
4003 CONJ and
858 V assure
2007 DET the
5 N quality
4003 CONJ and
8888 N timeliness
3032 P of
2007 DET the
620 N end
8888 ADJ product

F. Chunk Sentence

The above output is now input into the chunking process. This process uses generic sequences of POS to determine the start of NPs and VCs, and then fills the slots for the clauses with the composing words and index numbers. The object-oriented context free Phrase Structure Grammar parser written in C++ described by Perelman-Hall [4] has been substantially adapted to implement the concepts described previously. This process produces the following output. Slots containing blanks and zeroes have been eliminated because of space limitations.

```
NOUN PHRASE
------------------------------------------------------------
 FOR    EXAMPLE
IF     PEOPLE

0 3027 0 0 0 0 22 0
4012 0 0 0 0 0 997 0

VERB PHRASE
------------------------------------------------------------
    WORKING

0 0 0 0 0 0 677 0 0 0 0 0

NOUN PHRASE
------------------------------------------------------------
 ON A    GROUP
```

PROJECT

0 3034 2000 0 0 0 712 0
0 0 0 0 0 0 8888 0

VERB PHRASE

DID WORK

0 0 5008 0 0 0 677 0 0 0 0 0

NOUN PHRASE

THEIR
OWN WAY
AND ON THEIR
OWN SCHEDULE
IT

0 0 0 0 0 0 2008 0
0 0 8888 0 0 0 26 0
4003 3034 0 0 0 0 2008 0
0 0 8888 0 0 0 86 0
0 0 0 0 0 0 8888 0

VERB PHRASE

WOULD BE EXTREMELY

0 5032 5002 8888 0 0 0 0 0 0 0

NOUN PHRASE

TO DIFFICULT

0 3046 0 868 0 0 0 0

VERB PHRASE

COORDINATE

0 0 0 0 0 0 60 0 0 0 0 0

NOUN PHRASE

THEIR

0 0 0 0 0 0 2008 0

VERB PHRASE

AND WORK
ASSURE

4003 0 0 0 0 0 677 0 0 0 0 0
0 0 0 0 0 0 858 0 0 0 0 0

NOUN PHRASE

THE QUALITY
AND TIMELINESS
OF THE END
PRODUCT

0 0 2007 0 0 0 5 0
4003 0 0 0 0 0 8888 0
0 3032 2007 0 0 0 620 0
0 0 0 8888 0 0 0 0

VERB PHRASE
----

0 0 0 0 0 0 0 0 0 0 0 0

VII. Discussion

The semantic representation derived by the process described is now ready for further processing. Comparisons between documents can easily be made for looking for similar content, even if the documents use completely different wording. The programming of such content matching is relatively straightforward, because of the numerical array structures of the data. The AEG system uses such a technique when comparing student essays against model answers.

The AEG system based on this technique can process a 400 word essay in about 3 seconds.

The POS tagger, as with most taggers, does not accurately tag words in all cases, and so the chunking process does not produce completely accurate chunks. However this does not seem to hinder substantially the construction of a meaningful sentence summary.

The stemming program does not produce stems in many cases that are of the form required. Many of the stems are not real words, and so when the stems are used in the lookup process using the thesaurus, the words are not found. The stemming process needs to be modified to produce whole words, so that the success rate of stem lookups is improved.

VIII. References

[1] S.Alter, *Information Systems A Management Perspective*, Addison-Wesley, Reading, Mass.: 1992, 168.
[2] M.Kolln, *Understanding English Grammar*, MacMillan, New York. 1994.
[3] O.Mason, http://web.bham.ac.uk/O.Mason/software/tagger/
[4] D.Perelman-Hall, "A Natural Solution", *Byte*, 17, 2, February, 1992, 237-244.
[5] M.F.Porter, "An Algorithm For Suffix Stripping", *Program*, 14, 3, July, 1980, 130-137. (Code written by B.Frakes and C.Cox, 1986, and changed by C.Fox in 1990 and 1991).
[6] M.Poesio, "Semantic Analysis", in R.Dale, H.Moisl and H.Somers, *Handbook of Natural Language Processing*, Marcel Dekker, New York: 2000.