Controlling Lexical Substitution in Computer Text Generation[1]

Robert Granville
MIT Laboratory for Computer Science
545 Technology Square
Cambridge, Massachusetts 02139

Abstract

This report describes *Paul*, a computer text generation system designed to create cohesive text through the use of lexical substitutions. Specifically, this system is designed to deterministically choose between pronominalization, superordinate substitution, and definite noun phrase reiteration. The system identifies a *strength of antecedence recovery* for each of the lexical substitutions, and matches them against the *strength of potential antecedence* of each element in the text to select the proper substitutions for these elements.

1. Introduction

This report describes *Paul*, a computer text generation system designed to create cohesive text through the use of lexical substitutions. Specifically, this system is designed to deterministically choose between pronominalization, superordinate substitution, and definite noun phrase reiteration. The system identifies a *strength of antecedence recovery* for each of the lexical substitutions, and matches them against the *strength of potential antecedence* of each element in the text to select the proper substitutions for these elements.

Paul is a natural language generation program initially developed at IBM's Thomas J. Watson Research Center as part of the ongoing Epistle project [5, 6]. The emphasis of the the work reported here is in the research of *discourse phenomena*, the study of cohesion and its effects on multisentential texts [3, 9]. *Paul* accepts as input LISP knowledge structures consisting of case frame [1] formalisms representing each sentence to be generated. These knowledge structures are translated into English, with the appropriate lexical substitutions being made at this time. No attempt is made by the system to create these knowledge structures.

2. Cohesion

The purpose of communication is for one person (the speaker or writer) to express her thoughts and ideas so that another (the listener or reader) can understand them. There are many restrictions placed on the realization of these thoughts into language so that the listener may understand. One of the most important requirements for an utterance is that it seem to be unified, that it form a *text*. The theory of text and what distinguishes it from isolated sentences that is used in *Paul* is that of Halliday and Hasan [3].

One of the items that enhances the unity of text is *cohesion*. Cohesion refers to the linguistic phenomena that establish relationships between sentences, thereby tying them together. There are two major goals that are accomplished through cohesion that enhance a passage's quality of text. The first is the obvious desire to avoid unnecessary repetition. The other goal is to distinguish new information from old, so that the listener can fully understand what is being said.

{1} The room has a large window. The room has a window facing east.

{1} appears to be describing two windows, because there is no device indicating that the window of the second sentence is the same as the window of the first sentence. If in fact the speaker meant to describe the same window, she must somehow inform the listener that this is indeed the case. Cohesion is a device that will accomplish this goal.

Cohesion is created when the interpretation of an element is dependent on the meaning of another. The element in question cannot be fully understood until the element it is dependent on is identified. The first *presupposes* [3] the second in that it requires for its understanding the existence of the second. An element of a sentence presupposes the existence of another when its interpretation requires *reference* to another. Once we can trace these references to their sources, we can correctly interpret the elements of the sentences.

The very same devices that create these dependencies for interpretation help distinguish old information from new. If the use of a cohesive element presupposes the existence of another reference of the element for its interpretation, then the listener can be assured that the other reference exists, and that the element in question can be understood as old information. Therefore, that act of associating sentences through reference dependencies helps make the text unambiguous, and cohesion can be seen to be a very important part of text.

3. Lexical Substitution

In [3], Halliday and Hasan catalog and discuss many devices used in English to achieve cohesion. These include reference, substitution ellipsis, and conjunction. Another family of devices they discuss is known as lexical substitution. The lexical substitution devices incorporated into *Paul* are pronominalization, superordinate substitution, and definite noun phrase reiteration.

Superordinate substitution is the replacement of an element with a noun or phrase that is a more general term for the element. As an example, consider Figure 1, a sample hierarchy the system uses to generate sentences.

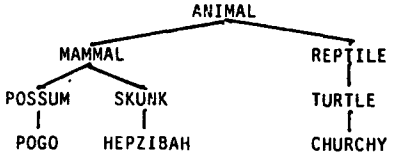

Figure 1a

```
1. POGO IS A MALE POSSUM.
2. HEPZIBAH IS A FEMALE SKUNK.
3. CHURCHY IS A MALE TURTLE.
4. POSSUMS ARE SMALL, GREY MAMMALS.
5. SKUNKS ARE SMALL, BLACK MAMMALS.
6. TURTLES ARE SMALL, GREEN REPTILES.
7. MAMMALS ARE FURRY ANIMALS.
8. REPTILES ARE SCALED ANIMALS.
```

Figure 1b: A Sample Hierarchy for *Paul*

[1]This research was supported (in part) by Office of Naval Research contract N00 14-80-C-0505, and (in part) by National Institutes of Health Grant No. 1 P01 LM 03374-04 from the National Library of Medicine.

In this example, the superordinate of *POGO* is *POSSUM*, that of *POSSUM* is *MAMMAL*, and again for *MAMMAL* the superordinate is *ANIMAL*. Superordinates can continue for as long as the hierarchical tree will support.

The mechanics for performing superordinate substitution is fairly easy. All one needs to do is to create a list of superordinates by tracing up the hierarchical tree, and arbitrarily choose from this list. However, there are several issues that must be addressed to prevent superordinate substitution from being ambiguous or making erroneous connotations. The erroneous connotations occur if the list of superordinates is allowed to extend too long. An example will make this clear. Let us assume that we have a hierarchy in which there is an entry *FRED*. The superordinate of *FRED* is *MAN*, for *MAN HUMAN*, *ANIMAL* for *HUMAN*, and *THING* for *ANIMAL*. Therefore, the superordinate list for *FRED* is *(MAN HUMAN ANIMAL THING)*. While referring to Fred as *the man* seems fine, calling him *the human* seems a little strange. And furthermore, using *the animal* or *the thing* to refer to Fred is actually insulting.

The reason these superordinates have negative connotations is that there are essential qualities that humans possess that separate us from other animals. Calling Fred an "animal" implies that he lacks these qualities, and is therefore insulting. "Human" sounds strange because it is the highest entry in the semantic hierarchy that exhibits these qualities. Talking about "the human" gives one the feeling that there are other creatures in the discourse that aren't human.

Paul is sensitive to the connotations that are possible through superordinate substitution. The system identifies an essential quality, usually intelligence, which acts as a block for further superordinate substitution. If the item to be replaced with a superordinate has the property of intelligence, either directly or through semantic inheritance, a superordinate list is made only of those entries that have themselves the quality of intelligence, again either directly or through inheritance. If the item doesn't have intelligence, the list is allowed to extend as far as the hierarchical entries will allow. Once the proper list of superordinates is established, *Paul* randomly chooses one, preventing repetition by remembering previous choices.

The other problem with superordinate substitution is that it may introduce ambiguity. Again consider Figure 1. If we wanted to perform a superordinate substitution for *POGO*, we would have the superordinate list *(POSSUM MAMMAL ANIMAL)* to choose from. But *HEPZIBAH* is also a mammal, so *the mammal* could refer to either *POGO* or *HEPZIBAH*. And not only are both *POGO* and *HEPZIBAH* animals, but so is *CHURCHY*, so *the animal* could be any one of them. Therefore, saying *the mammal* or *the animal* would form an ambiguous reference which the listener or reader would have no way to understand.

Paul recognizes this ambiguity. Once the superordinate has been selected, it is tested against all the other nouns mentioned so far in the text. If any other noun is a member of the superordinate set in question, the reference is ambiguous. This reference can be disambiguated by using some feature of the element being replaced as a *modifier*. In our example of Figure 1, we find that all possums are grey, and therefore *POGO* is grey. Thus, *the grey mammal* can refer only to *POGO*, and is not ambiguous. In the Pogo world, the features the system uses to disambiguate these references are gender, size, color, and skin type (furry, scaled, or feathered). Once the feature is arbitrarily selected and the correct value has been determined, it is tested to see that it genuinely disambiguates the reference. If any of the nouns that were members of the superordinate set have the same value for this feature, it cannot be used to disambiguate the reference, and it is rejected. For instance, the size of *POGO* is small, but saying *the small mammal* is still ambiguous because *HEPZIBAH* is also small, and the phrase could just as likely refer to her. The search for a disambiguating feature continues until one is found.

Pronominalization, the use of personal pronouns in place of an element, is mechanically simple. The selection of the appropriate personal pronoun is strictly grammatical. Once the syntactic case, the gender, and the number of the element are known, the correct pronoun is dictated by the language.

The final lexical substitution available to *Paul* is the definite noun phrase, the use of a definite article, *the* in English, as opposed to an indefinite article, *a* or *some*. The definite article clearly marks an item as one that has been previously mentioned, and is therefore old information. The indefinite article similarly marks an item as *not* having been previously mentioned, and therefore is new information. This capacity of the definite article makes its use required with superordinates.

{2} My collie is smart. The dog fetches my newspaper every day.

***My collie is smart. A dog fetches my newspaper every day.**

While the mechanisms for performing the various lexical substitutions are conceptually straightforward, they don't solve the entire problem of using lexical substitution. Nothing has been said about how the system chooses which lexical substitution to use. This is a serious issue because lexical substitution devices are *not* interchangeable. This is true because lexical substitutions, as with most cohesive devices, create text by using *presupposed dependencies* for their interpretations, as we have seen. If those presupposed elements do not exist, or if it is not possible to correctly identify which of the many possible elements is the one presupposed, then it is impossible to correctly interpret the element, and the only possible result is confusion. A computer text generation system that incorporates lexical substitution in its output must insure that the presupposed element exists, and that it can be readily identified by the reader.

Paul controls the selection of lexical substitution devices by conceptually dividing the problem into two tasks. The first is to identify the *strength of antecedence recovery* of the lexical substitution devices. The second is to identify the *strength of potential antecedence* of each element in the passage, and determine which if any lexical substitution would be appropriate.

4. Strength of Antecedence Recovery

Each time a cohesive device is used, a presupposition dependency is created. The item that is being presupposed must be correctly identified for the correct interpretation of the element. The relative ease with which one can recover this presupposed item from the cohesive element is called the *strength of antecedence recovery*. The stronger an element's strength of antecedence recovery, the easier it is to identify the presupposed element.

The lexical substitution with the highest strength of antecedence recovery is the definite noun. This is because the element is actually a repetition of the original item, with a definite article to mark the fact that it is old information. There is no real need to refer to the presupposed element, since all the information is being repeated.

Superordinate substitution is the lexical substitution with the next highest strength of antecedence recovery. Presupposition dependency genuinely does exist with the use of superordinates, because some information is lost. When we move up the semantic hierarchy, all the traits that are specific to the element in question are lost. To recover this and fully understand the reference at hand, we must trace back to the original element in the hierarchy. Fortunately, the manner in which *Paul* performs superordinate substitution facilitates this recovery. By insuring that the superordinate substitution will never be ambiguous, the system only generates superordinate substitutions that are readily recoverable.

The third device used by *Paul*, the personal pronoun, has the lowest strength of antecedence recovery. Pronouns genuinely are nothing more than place holders, variables that maintain the positions of the elements they are replacing. A pronoun contains no real semantic information. The only readily available pieces of information from a pronoun are the syntactic role in the current sentence, the gender, and the number of the replaced item. For this reason, pronouns are the hardest to recover of the substitutions discussed.

5. Strength of Potential Antecedence

While the forms of lexical substitution provide clues (to various degrees) that aid the reader in recovering the presupposed element, the actual way in which the element is currently being used, how it was previously used, its circumstances within the current sentence and within the entire text, can provide additional clues. These factors combine to give the specific reference a *strength of potential antecedence*. Some elements, by the nature of their current and previous usage, will be easier to recover independent of the lexical substitution device selected.

Strength of potential antecedence involves several factors. One is the *syntactic role* the element is playing in the current sentence, as well as in the previous reference. Another is the *distance* of the previous reference from the current. Here distance is defined as the number of clauses between the references, and *Paul* arbitrarily uses a distance of no more than two clauses as an acceptable distance. The current expected

focus of the text also affects an element's potential strength of antecedence. In order to identify the current expected focus, *Paul* uses the detailed algorithm for focus developed by Sidner [10].

Paul identifies five classes of potential antecedence strength. Class I being the strongest and Class V the weakest, as well as a sixth "non-class" for elements being mentioned for the first time. These five classes are shown in Figure 2.

Class I:

1. The sole referent of a given gender and number (singular or plural) last mentioned within an acceptable distance. **OR**

2. The *focus* or the head of the *expected focus list* for the previous sentence.

Class II:

The last referent of a given gender and number last mentioned within an acceptable distance.

Class III:

An element that filled the same syntactic role in the previous sentence.

Class IV:

1. A referent that has been previously mentioned, **OR**

2. A referent that is a member of a previously mentioned set that has been mentioned within an acceptable distance.

Class V:

A referent that is known to be a part of a previously mentioned item.

Figure 2: The Five Classes of Potential Antecedence

Once an element's class of potential antecedence is identified, the selection of the proper lexical substitution is easy. The stronger an element's potential antecedence, the weaker the antecedence of the lexical substitution. Figure 3 illustrates the mappings from potential antecedence to lexical substitution devices. Note that Class III elements are unusual in that the device used to replace them can vary. If the previous instance of the element was of Class I, if it was replaced with a pronoun, then the current instance is replaced with a pronoun, too. Otherwise, Class III elements are replaced with superordinates, the same as Class II.

```
Class I.....................Pronoun Substitution
Class II.....................Superordinate Substitution
Class III (previous reference Class I)
          ...................Pronoun Substitution
Class III.....................Superordinate Substitution
Class IV.....................Definite Noun Phrase
Class V.....................Definite Noun Phrase
```

Figure 3: Mapping of Potential Antecedence Classes to Lexical Substitutions

6. An Example

To see the effects of controlled lexical substitution, and to help clarify the ideas discussed, an example is provided. The following is an actual example of text generated by *Paul*. The domain is the so-called children's story, and the example discussed here is one about characters from Walt Kelly's *Pogo* comic strip, as shown in Figure 1 above.

Figure 4 contains the semantic representation for the example story to be generated, in the syntax of NLP [4] records.[2]

```
a1('like',exp:='a2',recip:='a3',stative);
a2('pogo');
a3('hepzibah');
b1('like',exp:='b2',recip:='a3',stative);
b2('churchy');
c1('give',agnt:='a2',aff:='c2',recip:='a3',
   active,effect:='c3');
c2('rose');
c3('enjoy\',recip:='a3',stative);
d1('want\',exp:='a3',recip:='d2',neg,stative);
d2('rose',possess:='b2');
e1('b2',char:='jealous',entity);
f1('hit\',agnt:='b2',aff:='a2',active);
g1('give',agnt:='b2',aff:='g2',
   recip:='a3',active);
g2('rose');
h1('drop\',exp:='h2',stative);
h2('petal',partof:='g2',plur);
i1('upset\',recip:='a3',cause:='h1',stative);
j1('cry\',agnt:='a3',active)[]
```

Figure 4: NLP Records for Example Story

If the story were to be generated without any lexical substitutions at all, it would look like the following.

POGO CARES FOR HEPZIBAH. CHURCHY LIKES HEPZIBAH, TOO. POGO GIVES A ROSE TO HEPZIBAH, WHICH PLEASES HEPZIBAH. HEPZIBAH DOES NOT WANT CHURCHY'S ROSE. CHURCHY IS JEALOUS. CHURCHY HITS POGO. CHURCHY GIVES A ROSE TO HEPZIBAH. PETALS DROP OFF. THIS UPSETS HEPZIBAH. HEPZIBAH CRIES.

While this version of the story would be unacceptable as the final product of a text generator, and it is *not* the text *Paul* would produce from the input of Figure 4, it is shown here so that the reader can more easily understand the story represented semantically in Figure 4.

To go to the other extreme, uncontrolled pronominalization would be at least as unacceptable as no lexical substitutions at all.

POGO LIKES HEPZIBAH. CHURCHY CARES FOR HER, TOO. HE GIVES A ROSE TO HER. WHICH PLEASES HER. SHE DOES NOT WANT HIS ROSE. HE IS JEALOUS. HE SLUGS HIM. HE GIVES A ROSE TO HER. PETALS DROP OFF. THIS UPSETS HER. SHE CRIES.

Again, this is unacceptable text, and the system would not generate it, but it is shown here to dramatize the need for control over lexical substitutions.

The text that *Paul* actually does produce from the input of Figure 4 is the following story.

POGO CARES FOR HEPZIBAH. CHURCHY LIKES HER, TOO. POGO GIVES A ROSE TO HER, WHICH PLEASES HER. SHE DOES NOT WANT CHURCHY'S ROSE. HE IS JEALOUS. HE PUNCHES POGO. HE GIVES A ROSE TO HEPZIBAH. THE PETALS DROP OFF. THIS UPSETS HER. SHE CRIES.

[2] For a discussion of the implementation of NLP for *Paul* see [2].

7. Conclusions

The need for good text generation is rapidly increasing. One requirement for generated output to be considered text is to exhibit cohesion. Lexical substitution is a family of cohesive devices that help provide cohesion and achieve the two major goals of cohesion, the avoiding of unnecessary repetition and the distinguishing of old information from new. However, uncontrolled use of lexical substitution devices will produce text that is unintelligible and nonsensical. *Paul* is the first text generation system that incorporates lexical substitutions in a controlled manner, thereby producing cohesive text that is understandable. By identifying the strength of antecedence recovery for each of the lexical substitutions, and the strength of potential antecedence for each element in the discourse, the system is able to choose the appropriate lexical substitutions.

8. Acknowledgments

I would like to thank Pete Szolovits and Bob Berwick for their advice and encouragement while supervising this work. I would also like to thank George Heidorn and Karen Jensen for originally introducing me to the problem addressed here, as well as their expert help at the early stages of this project.

9. References

1. Fillmore, Charles J. The Case for Case. In *Universals in Linguistic Theory*. Emmon Bach and Robert T. Harms, Ed., Holt, Rinehart and Winston, Inc., New York, 1968.

2. Granville, Robert Alan. Cohesion in Computer Text Generation: Lexical Substitution. Tech. Rep. MIT/LCS/TR-310, MIT,Cambridge, 1983.

3. Halliday, M. A. K., and Ruquaiya Hasan. *Cohesion in English*. Longman Group Limited, London, 1976.

4. Heidorn, George E. Natural Language Inputs to a Simulation Programming System. Tech. Rep. NPS-55HD72101A, Naval Postgraduate School, Monterey, Cal., 1972.

5. Heidorn, G. E., K. Jensen, L. A. Miller, R. J. Byrd, and M. S. Chodorow. The Epistle Text-Critiquing System. *IBM Systems Journal 21*, 3 (1982).

6. Jensen, Karen, and George E. Heidorn. The Fitted Parse: 100% Parsing Capability in a Syntactic Grammar of English. Tech. Rep. RC 9729 (# 42958), IBM Thomas J. Watson Research Center, 1982.

7. Jensen, K., R. Ambrosio, R. Granville, M. Kluger, and A. Zwarico. Computer Generation of Topic Paragraphs: Structure and Style. Proceedings of the 19th Annual Meeting of the Association for Computational Linguistics, Association for Computational Linguistics, 1981.

8. Mann, William C., Madeline Bates, Barbara J. Grosz, David D. McDonald, Kathleen R. McKeown, and William R. Swartout. Text Generation: The State of the Art and the Literature. Tech. Rep. ISI/RR-81-101, Information Sciences Institute, Marina del Rey, Cal., 1981. Also University of Pennsylvania MS-CIS-81-9.

9. Quirk, Randolph, Sidney Greenbaum, Geoffrey Leech, and Jan Svartik. *A Grammar of Contemporary English*. Longman Group Limited, London, 1972.

10. Sidner, Candace Lee. Towards a Computational Theory of Definite Anaphora Comprehension in English Discourse. Tech. Rep. AI-TR 537, MIT, Cambridge, 1979.