Creativity in Natural Language: Studying Lexical Relations

Mateus Mendes[*], **Francisco C. Pereira**[†], **Amílcar Cardoso**[†]

[*]Escola Superior de Tecnologia e Gestão de Oliveira do Hospital
R. General Santos Costa, 3400-124 Oliveira do Hospital, Portugal
mmendes@estgoh.ipc.pt

[†]Departamento de Engenharia Informática da FCTUC
Pólo II, Pinhal de Marrocos, 3000 Coimbra, Portugal
{camara, amilcar}@dei.uc.pt

Abstract

There are already many systems provided with the capacity of automatically generating sentences. Most of them were developed for reliability, others for creativity. Dupond uses lexical relations to transform a sentence, following certain criteria. It is able to produce new sentences keeping the original meaning. It was developed as part of a larger project whose goal is to understand how lexical relations can be used to influence creativity in natural language. But Dupond is suitable for use in applications such as chatter bots and other sentence generators.

1. Introduction

Due to the generalised use of computers, the problem of automatic text generation has become of crucial relevance in recent years. Many systems have already been developed which generate natural language, but most of them invariably produce well known sentences based on rigid templates or other strict rules that make them repeat themselves with little variance. Some systems produce novel sentences, but these don't usually limit their output to a given topic. Dupond was built with the purpose of studying how lexical relations can be used to achieve some creative automatic discourse. It is able to produce different sentences to express the same idea.

Before a truly creative sentence generator can be built, it is necessary to understand what creativity in natural language is. Then we can go further to mimic it. Dupond can presently be fed a sentence and, using selected lexical relations, translate it into another one. Ideally, this new sentence should express the same idea carried by the original one.

Below is a short review of some related work. Section 3. explains the system's theoretical principles, based on the properties of natural language. Section 4. describes the system's capabilities. Sections 5. and 6. contain a short description of its internal modules and how they work. Finally, section 7. discusses some preliminary results.

2. Related Work

2.1. Random sentence generators

Random sentence generators are the simplest ones and don't usually require a complete and well structured knowledge base. They simply pick-up random words or phrases and fit them together in a particular, grammatically correct, order. They are not at all reliable, and their interest, on a scientific view, is very limited. The most frequent practical applications for random sentence generators are word games. Spew and Yak (Schwartz, 1999) are examples of these kind of generators. They are simple word-fitting systems, built just for play. Hypercard Random Sentence Generator (Kelly, 1993) is another example, with the particularity that it applies the theory of random text generation to language teaching.

2.2. Straight sentence generators

Straight sentence generators produce their output in a carefully studied way, and their reliability makes them suitable for many different purposes. Their creativity is very limited, if it ever exists at all. Long interactions with these systems are often boring, and they are not supposed to be used as creativity-aid tools. They are very useful for tasks such as translation, question-answering, report and letter writting, or summarising.

The simplest strict sentence generators are template-based. They contain a set of templates with empty slots that can be filled with known pieces of information. This approach is widely used, because of its low complexity. Most modern text editors are good examples of these systems, since they provide the user with template-based letters, reports and other documents. Another example is Eliza (Weizenbaum, 1966), a computer program built in the sixties, which emulates the discourse of a psychotherapist. Eliza is considered the first great automatic chatterer. *She* works based on tricks like string substitution and canned responses triggered by keywords.

More complex systems usually produce the sentences from formal specifications and grammatical rules. Penman (Matthiessen, C.M.I.M. and Bateman, 1991) is one of the most well known systems of this kind. It receives as input a formal specification of a sentence and translates it into words using the theory of Systemic Functional Linguistics. Internally Penman consists of a network of over 700 nodes, each node representing a single minimal grammatical alternation. In order to generate a sentence, Penman traverses the network guided by its inputs and default settings. At each system node Penman selects a feature until it has assembled enough features to fully specify a sentence. After constructing a syntax tree and choosing words to satisfy the features selected, Penman then generates the sentence.

Straight sentence generators have long been used for many different purposes. Examples include the CO-OP paraphraser (McKeown,), the AGILE (Hana, 2001) translator, the SummariserPort (Oliveira, Paulo et al., 2002) text summariser and the IDAS (Reiter, Ehud et al., 1992) documentation writer, among others.

3. Creativity, Natural Language and Fluency

Natural language is usually analysed in three different layers: syntax, semantics and pragmatics. Creativity can be spotted in any of these layers.

At the syntactic level creative sentences can arise from an original sentence form or an irregular word or phrase ordering. Since syntax in most languages is ruled by well known grammatical rules, creativity at this level is limited to either respecting these rules and have little liberty or breaking them and produce ungrammatical sentences - either meaningless or not. At the semantic level creativity can be the product of using some word or expression to mean something unusual. Poets and some writers do it all the time, producing the literary discourse. Since semantic rules are not as strict as the syntactical ones, it's easier to work on creativity at the semantic level. Pragmatics relates to the *context*, and can be exploited to disambiguate words and make semantic shifts meaningful and useful. Creativity in this level depends on things such as one's culture, values and education.

Writers exploit both syntax, semantics and pragmatics' properties to achieve a fluent discourse, through the use of **figures of speech**. Most of the figures of speech are the product of conceptual relations (metaphor and simile, for instance) and require knowledge and careful reasoning about the world. So far, Dupond doesn't use figures of speech theory in order to produce its output.

The use of lexical relations is another way to express the same idea in different ways. Lexical relations are the following: antonymy, hypernymy/hyponymy, antonymy, homophony, homonymy, polysemy, metonymy and collocation (Yule, 2001). Collocation is an aspect of language which characterises words which tend to occur with other words. For instance, many people associate the pairs *salt-pepper* and *table-chair*. This is just a characteristic that seems of little use for Dupond. Metonymy is a whole-part relation between some words (*car-wheels*, *house-roof*) that makes possible the use of one for replacing another. Most examples of metonymy are highly conventionalised and easy to interpret. However, many others depend on an ability to infer what the speaker has in mind. Thus, this interchangeability requires pragmatic analyses and a good database of knowledge. Polysemy can be defined as one form of a word having multiple meanings, which are all related. For example the words *head*, meaning something or someone *on top* of something. Homonymy can be defined as one form of a word having multiple meanings, but which are not related. For example, *race [speed]* and *race [ethnic group]*. Homophony happens when two differently written words have the same pronunciation (*bare-bear*, for instance). Polysemy, homonymy and homophony make it possible to do some language tricks, but the latter is only suitable for oral speech, and the formers shall not be used if one wants the system to be reliable. Antonymy occurs when two words have opposite meanings, and it is mostly convenient for us to transmit meaning. For instance, our natural explanation for *dirty* is *not clean*. But antonymy is not a general relation we can use in all the situations. Consider the word *beautiful*. Searching the WordNet [1] (Fellbaum, 1998) for antonyms we find *ugly*, but we cannot say the sentence *It's a beautiful morning* is the same as *It's a not ugly morning*. Antonymy is good for explaining relationships with other words in many different situations, but its use requires some common-sense knowledge, so that one knows where to use it.

Hypernymy/hyponymy relations happen when the meaning of one word is included in the meaning of another. A typical pair is *dog-animal*, where dog is an hyponym of animal and the later is a hypernym of the former. One can replace any word in any sentence by one hypernym without changing the original idea. At most the result is an odd sentence or a general, ambiguous sentence. For example, consider the word *girl*. Searching WordNet for hypernyms we find *girl* is a kind of *woman*, *woman* is a kind of *female*, and there are 4 more relations before getting to the top word *entity*. All these words are semantically valid replacements for *girl*. In practise, though, replacements above 1 or 2 levels usually sound unnatural.

Synonymy is the most simple relation one can use, once the correct sense of a word is found. The vast majority of the words can be replaced by synonyms in almost all the contexts, although the result can be an odd sentence, or a different sentence in terms of formality. For instance, consider the sentences *Cathy had one answer correct on the test* and *My dad bought a bigcar*. Using synonyms for replacing words we can get to: *Cathy had one reply right on the examination*, which sounds odd, and *My father purchased a large automobile*, which sounds more formal.

4. Dupond's Features

For now, Dupond is able to disambiguate words, replace words by synonyms and hypernyms and suppress unnecessary words. Each of its features can be configured from *wanted* (always do that, if possible) to *not wanted*. In the middle level it is expected to do that half the time.

4.1. Disambiguating words

Disambiguation is done in function of the context. For instance, in the sentence *That woman is a dog*, the meaning of *dog* is probably *{dog,frump}*, and the system finds it realising that the word *woman* is found in the sentence and the WordNet gloss for the sense *{dog,frump}*. If the word *dog* has never been used in this sense in the current session, Dupond accepts this sense with a given confidence. If it has been used in another sense, then the previous confidence is

[1]Wordnet is available at http://www.cogsci.princeton.edu/˜wn/.

pondered and the preferred sense is a function of the previous confidence and the confidence in the current disambiguation. If the word cannot be disambiguated in function of the context but it has been used before, then the sense with the highest rank is accepted as its current meaning. If the word has not been used before and cannot be disambiguated, then the most frequent sense is preferred with no confidence at all.

4.2. Selecting replacement words

Once we have a word and a set of synonyms in a given context, there are various possible criteria to choose a valid replacement.

One possible criterion is to pick the one with less senses, thus minimising the probability of misinterpretation. Another possible criterion is to pick the one with more senses, thus maximising the probability of that being a known word. Dupond can follow any of the criteria or simply pick a synonym randomly. It can also use hypernymy relations, up to 7 levels, to find valid replacements.

4.3. Other features

Dupond can also be configured to prefer previously used replacements and/or replacement methods, thus producing a more coherent discourse. For instance, consider it chose the word *miss* to replace *woman*. If it is configured to reuse previous replacements, the following occurrences of *woman* will always be replaced by *miss*.

Another feature is its ability to suppress unnecessary words. For instance, consider the sentence *John ate cookies and Mary [ate] cake*. The word in square brackets can be suppressed without the sentence loosing significance and it becomes simpler.

The fact that the system is based on the product of probabilities gives it an infinite flexibility.

5. Dupond's architecture

The system's architecture is as shown in figure 1. All the processing is coordinated by the server module, which receives sentences and orders from its clients through a message queue, performs the necessary steps and sends the new sentences and responses back to them. Users are not expected to interact with the server directly. There is a web client interface where the users can set their preferences and send their sentences in a comfortable way. The client then communicates with the server through the message queue. The server can attend many different clients at the same time. That led to the need of a module for user authentication. When a client sends his first message, it is assigned an identification number and a data structure is created for it. User preferences and some data about the ongoing dialogue are stored, for better performance.

After receiving a sentence, the very first step the server performs is to parse it. A sentence which cannot be parsed, either because it is ungrammatical or for some other reason, is not translated. For parsing Dupond uses Link Grammar Parser[2], a free parser based on link grammar (Sleator and Temperley, 1993). Once the sentence is successfully parsed, the server obtains an equivalent tree-structure which contains all the necessary information about it. The grammatical category of each word and its connection with other words in the same sentence should be well known. Figure 2 illustrates an example parse tree. Suffixes indicate the grammatical category of each word. For instance, ".n" is appended to nouns, and ".v" is appended to verbs.

Presently the parse tree is not used - only the tags attached to each word. In the future the tree may be used to replace phrases or other portions of the sentence.

But knowing the grammatical category of a word is not enough for this system. Consider the word"girls": we need to know not only that it is a name but also that it's in the plural form. To solve this problem there's an additional module, named **Morphy**. Morphy can be interfaced in two different ways. If it is given as input a word in its context it returns complete information about it. For instance, when asked for the word *girls*, morphy would find it's a plural noun and its base form is *girl*. On the other hand, it can be asked what the plural form for the noun *girl* is, and the output would be *girls*.

The disambiguation module tries to find the correct sense of a word, based on the present context and any previous concepts. For example, consider the sentence "The bird went to the market". Searching the WordNet for *bird* we find 5 senses for the noun and 1 for the verb. Since we parsed the sentence we know *bird* is a noun. When asked for the correct sense of this noun in this context, the disambiguator module would return sense 3, indicating that *bird* refers to a girl with an acceptable confidence. If we had been using the noun *bird* in sense 1 *(warm-blooded egg-laying vertebrates characterised by feathers and forelimbs modified as wings)* for a long time before, the disambiguator would most probably return sense one with little confidence. If it cannot disambiguate the word, the module returns the most frequent sense with no confidence.

The **Replacer** module receives the disambiguated word and the set of user preferences. In function of the user's preferences, it picks an appropriated word that could replace the original one. The server uses all these modules to parse the sentence, disambiguate each word, get its base form, find a valid replacement word, put it in the correct grammatical form and rebuild a new sentence.

6. Finding valid replacements

Dupond is controlled internally by "state words". This state words represent sets of probabilities whose values the user can change in order to get different behaviours. Figure 3 shows the system's interface.

If the state words are null, all the probabilities are zero and the system's output is equal to the input.

Once the sentence is parsed, the first optional step Dupond can perform is to disambiguate each word. For this step the user can choose between disambiguation in function of the context, picking the most frequent sense or pick a sense randomly. If the user assigns 7 to the "Disambiguate words" option, the system will always try to disambiguate. 0 means Dupond should never disambiguate, and try any of the other options if they are selected. Once a sense is selected for a given word, it's necessary to choose a valid replacement for it. For example, considering the

[2] http://www.link.cs.cmu.edu/link/

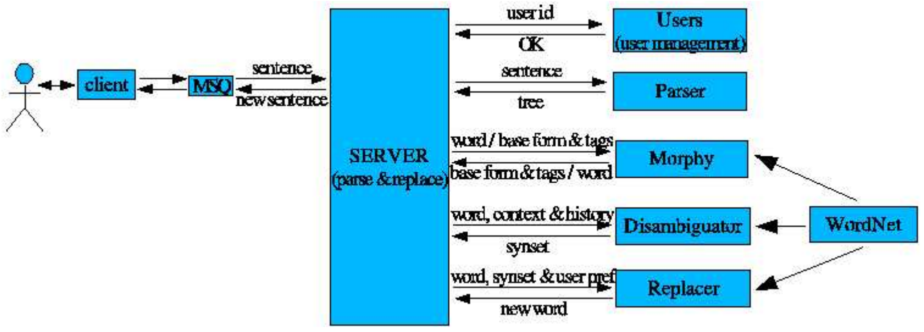


Figure 1: System Overview.

```
   +-Dmc-+---Sp--+--PP-+---Op--+
   |     |       |     |       |
the girls.n have.v got.v flowers.n
```

Figure 2: Parse tree for the sentence "The girls have got flowers".

word *confess* in the last sentence shown in figure 3, the disambiguation process would return the sense 1: *{confess, squeal, shrive}*. The replacement module should then select a valid word from this setfor replacing *confess*. If the user had assigned 7 to the "Prefer synonym with less senses:" option, Dupond would always select the verb *shrive*, since this word contains only one sense and *confess* and *squeal* contain more. The "Trust memory and acquired concepts" option tells the system to repeat previous replacements. If this was selected in the example above, the word *progress* in this sense would always be replaced by *advancement*. The option "Prefer previously used methods:" intends to make the system be more coherent with past behaviour. It tells Dupond to reuse previously applied methods. For instance, if it explored an hypernymy relation to replace a noun (*e.g. dog* -> *canine*), it should use hypernymy to find replacements for subsequent nouns (*e.g. cat* -> *feline*).

7. Preliminary Results

The main goal of this project is to study how important the lexical relations may be to produce sentences in an original way. This involves two steps: 1) build a system able to receive a sentence and, using lexical relations, produce a different one with an equivalent meaning; 2) study how different, meaningful and interesting this automatically rebuilt sentences are for the people. Dupond was built for performing step 1. It can be fed English sentences and rebuild them in function of the user's preferences.

Figure 3 shows a sample session, using sentences selected from the first paragraphs of the book "The return of Sherlock Holmes"[3], with the options shown in the figure.

8. Conclusions

Sentence generators are being used more and more in modern intelligent systems. Creativity will play an important role if one wants to overcome the present limitation that makes machines' speech sound unnatural and repetitive. Dupond is an automatic word replacer ready for being used in the study of natural language and/or other applications. Namely, it may be adapted for automatic chatter bots, documentation and letter writers, message generators and similar systems. Indeed, its main limitation is that it isn't a stand-alone system, thus not suitable for any purpose on its own.

In future work Dupond will be used to study how lexical relations may be used to improve the creativity of natural language generation systems. Possible questions to be answered are: "Do people prefer the more common or the less common words? What makes a sentence look like odd? Do people prefer words with more or less senses?".

Dupond may also be improved for dealing with some figures of speech, replacing phrases and sets of words as well as working on the syntactic and pragmatic levels.

9. References

Fellbaum, Christiane (ed.), 1998. *WordNet: An Electronic Lexical Database*. USA: Bradford Books.

Hana, Jirí, 2001. The agile system. In *PBML*. Praha, pages 39–67.

Kelly, Charles, 1993. A hypercard random sentence generator for language study. *Bulletin of Aichi Institute of Technology*, 28, Part A:51–55.

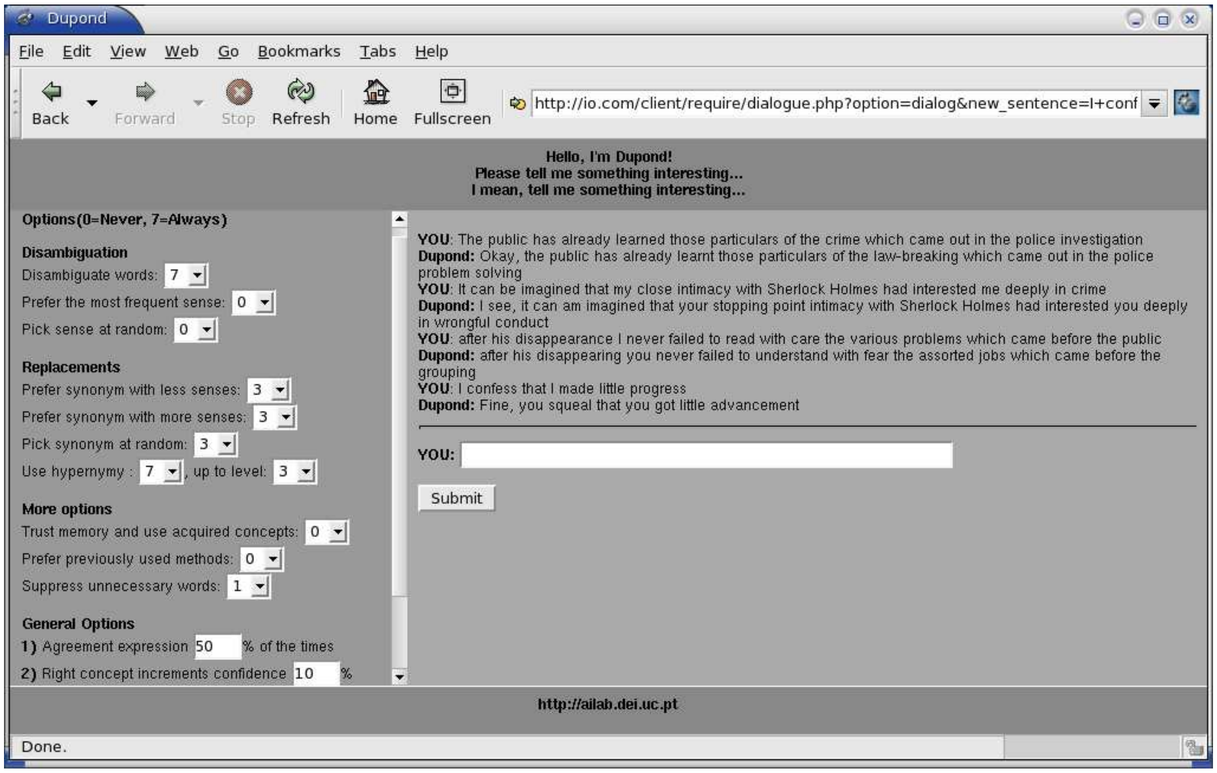
Matthiessen, C.M.I.M. and J.A. Bateman, 1991. *Text Generation and Systemic-Functional Linguistics*. London: Pinter.

McKeown, Kathleen R. Paraphrasing using given and new information in a question-answer system. In *17th Annual Meeting of the Association for Computational Linguistics*. ACL.

Oliveira, Paulo, Khurshid Ahmad, and Lee Gillam, 2002. A financial news summarisation system based on lexical cohesion. In *Workshop at the International Conference on Terminology and Knowledge Engineering*.

Reiter, Ehud, Chris Mellish, and John Levine, 1992. Automatic generation of on-line documentation in the idas

[3]"The return of Sherlock Holmes", by Arthur Conan Doyle. Downloaded from the project Gutenberg: http://gutenberg.net.


Dupond
File Edit View Web Go Bookmarks Tabs Help
Back Forward Stop Refresh Home Fullscreen
Submit
Done.

http://io.com/client/require/dialogue.php?option=dialog&new_sentence=I+conf

Hello, I'm Dupond!
Please tell me something interesting...
I mean, tell me something interesting...

Options(0=Never, 7=Always)

Disambiguation
Disambiguate words: 7
Prefer the most frequent sense: 0
Pick sense at random: 0

Replacements
Prefer synonym with less senses: 3
Prefer synonym with more senses: 3
Pick synonym at random: 3
Use hypernymy : 7, up to level: 3

More options
Trust memory and use acquired concepts: 0
Prefer previously used methods: 0
Suppress unnecessary words: 1

General Options
1) Agreement expression 50 % of the times
2) Right concept increments confidence 10 %

YOU: The public has already learned those particulars of the crime which came out in the police investigation
Dupond: Okay, the public has already learnt those particulars of the law-breaking which came out in the police problem solving
YOU: It can be imagined that my close intimacy with Sherlock Holmes had interested me deeply in crime
Dupond: I see, it can am imagined that your stopping point intimacy with Sherlock Holmes had interested you deeply in wrongful conduct
YOU: after his disappearance I never failed to read with care the various problems which came before the public
Dupond: after his disappearing you never failed to understand with fear the assorted jobs which came before the grouping
YOU: I confess that I made little progress
Dupond: Fine, you squeal that you got little advancement

YOU:

http://ailab.dei.uc.pt

Figure 3: Screenshot showing Dupond's client after a short session.

project. In *Procs of the 3rd Conference on Applied Natural Language Processing*. Trento, Italy.

Schwartz, Randal, 1999. Writing randomly. *Linux Magazine*. Http://www.stonehenge.com/merlyn/LinuxMag/col04.html.

Sleator, Daniel and Davy Temperley, 1993. Parsing english with a link grammar. Technical report, Third International Workshop on Parsing Technologies, Carnegie Mellon University.

Weizenbaum, Joseph, 1966. Eliza - a computer program for the study of natural language communication between man and machine. *Communications of the ACM*, 9(1):35–36.

Yule, George, 2001. *The study of language*, chapter 11. Cambridge, pages 114–125.