On parametering the choice of words in text generation and its usefulness in machine translation

Chadia Moghrabi

Université de Moncton. Canada
moghrac@umoncton.ca

Abstract

This paper describes briefly the overall architecture of a machine translation system between French and Arabic in the sub-world of cooking recipes. It continues to describe in more detail the design of the generation component and how this design allows a variety of outputs all expressing the same conceptual meaning. This system is of the family of knowledge-based interlingua translation systems as it emphasises the importance of the meaning of the text being processed and articulates all its available knowledge-bases in order to achieve one major goal: flexible meaningful wording.

We agree with S. Nirenburg that "the ability and the right to subdivide sentences or to combine them together in the Target language are powerful tools in the hands of human translators." These are some tools that we want our MT systems to be able to use.

The way the system is modularised allowed us to experiment with the generation of: sentence to sentence translations, text to text translations, more concise as opposed to more generalised wording, and varying word orders. The modules are declarative and loosely coupled. This strategy allowed us to experiment with regenerating back the French text. As a matter of fact, the generation component of this MT system is multilingual and capable of accommodating Arabic and French; two languages belonging to two different origins, namely Indo-European and Semitic.

This system, being functional in the domain of cooking recipes, allowed us to concentrate on the lexical semantics of its vocabulary and on the modularisation of its linguistic knowledge, whether it is morphological, syntactic or stylistic, as opposed to its pragmatic knowledge.

Now that we have tested the design on different languages, we are studying its feasibility in new domains where texts are mainly constituted of verbal phrases, such as in gardening and Chemistry laboratory manuals.

1 System's background

The first version of this system was designed at the University of Paris on an IBM mainframe by two PhD. students. At the time, the AI group at l'Université de Paris 6 had been experimenting with NLP in a variety of restricted domains including cooking recipes. The module for understanding cooking recipes in French was written by M. O. Cordier in 1979 and the text generator for recipes in Arabic was written by C. Moghrabi in 1980. These systems were respectively the subjects of the doctorate dissertations of the two researchers.

In 1982, Cordier and Moghrabi presented a joint MT system for cooking recipes at the European Conference on Artificial Intelligence (see ECAI'82 proceedings). They combined their systems and tested them to see to what extent they could translate from French into Arabic. The MT system adopted an interlingua approach where the analyser and the generator were completely blinded from one another except for the list of actions executable by a robot. The interlingua was

precisely the actions that the robot had to do when it "understood" the text. The translation was done text by text, not sentence by sentence. Based on Moghrabi's stylistic rules, the system can generate a variety of wording for the same recipe.

Moghrabi, who is now working for l'Université de Moncton in Canada, continued testing the text generator for more vocabulary and another language, namely French. As a matter of fact the MT system can read the French recipe and regenerate it back into French. In 1985, she presented her multi-lingual text generator at the APICS conference in Halifax, Canada. (See the conference proceedings for more detail.) Since then, Moghrabi has reprogrammed the system in C++, moved it to a PC, and is experimenting with new command/task intensive domains.

The following presentation first introduces the system's general characteristics along with an example of a French into Arabic translation thus allowing a better understanding of the variety of outputs that can be produced. It explains, later on, the system's components, its stylistic rules which allow output variety, and the newly added activators which prefer those Arabic realisations that are "closer" to the original French text.

For the sake of simplicity the translated text will be "paraphrased" in English!

2 General presentation and examples

Although the system's general architecture appears to be similar to those of other interlingua systems since it uses morphological, syntactical, and inference modules as well as dictionaries, this system's handling of its various knowledge-bases is quite different. All the system's knowledge is described symbolically in a specialised formalism; it is "read" as input, and it is used through those specialised interpreters which actually "do" the system's tasks.

Some of the similarities are: Simmons' semantic networks are similar to our conceptual structures; Goldman's defining characteristics are incorporated in the concept dictionary; Woods' ATNs inspired our syntactic graph. The system uses stylistic rules to do some aggregation, but in a slightly different way from that suggested by Dalianis and Hovy.

The system's major difference lies within the above mentioned interpreters that allow the "parametering" of the program, thus making it functional for a different language, for different word choices and order, and for a different domain.

*Take the example of the sentence: "Ajouter le sucre au lait chaud" (add the sugar to hot milk). The analyser produces its semantic network and sends the following list of basic actions (which are executable by a robot) to the text generator. This list is the program's interlingua that is independent of the source language.

```
PUTIN          MILK          RECIPIENT
LIGHTSTOVE
PUTON     RECIPIENT     STOVE
WAITUNTIL      TEMP = "hot"
PUTWITH        SUGAR MILK
```

The generator can produce a number of "translations":

1. Heat the milk then add the sugar to it.
   or
2. Add the sugar to hot milk.
   or
3. Dissolve the sugar in hot milk.

Notice that the three forms conserve the meaning. From the point of view of a pure interlingua paradigm, the three texts are acceptable. But only the second wording "really" translates as it captures the meaning and the form.

How can the system produce all these forms and how could it go about to prefer wording number 2 ? It is achieved through what we call stylistic rules and their activators.

3 System components

The analyser and the generator use the same type of components but in different ways. They might appear to be the mirror image, one of the other, but they are not.

3.1 The semantic network

This network is made up of various types of conceptual structures (ACTION, CHANGE-OF-STATE, ELEMENT, QUALIF ...). Each structure is made up of a particular list of information slots that is specific to its type.

Ex.: Semantic network generated for the above example:

```
ACTION =        slot 1 - "ADD"...in Arabic
type
structure slot 2 -  ELEMENT =       slot 1 - "SUGAR"...in Arabic
                        type                slot 2
                        structure...

                    slot 3 -  ELEMENT =     slot 1 - "MILK"
                        type                ...
                        structure...

                                            slot n -  QUALIF =      slot 1 TEMP
                                                type                slot 2  =
                                                struct.             slot 3 "hot"
                    slot n -  ...
```

These structure types and the list of slots associated with each type is parametered in order to allow changes in types and in slot configurations in case one wants to experiment on other domains.

This component is similar to Simmons' semantic networks. An ELEMENT (an ingredient, a utensil, or an instrument) is a primitive concept node, and is not recognised at this level as being the direct or the indirect object. The only significant information is that the sugar, in our example, is the element that is added to milk, hence its location changes. ACTION and CHANGE-OF-STATE structures are also primitive concept nodes. In these types of structures, no slots are allocated for modality (tense, aspect, mood, etc.).

3.2 The dictionaries

The concept and the word dictionaries are read into the program symbolically. They are used through their specialised interpreters.

The entries in these dictionaries are the basic actions (PUTON, PUTIN, PUTWITH, WAITUNTIL, etc.) and the conceptual objects (SUGAR, SAUCEPAN). They contain those mechanisms that are specialised in the creation and interconnection of conceptual structures, in filling their slots as well as making the necessary inferences. In the generation phase they allow the system to choose the most concise verbs to be used later during the syntactic phase. In other words, the generation concept dictionary contains (among other things) that pragmatic information that Goldman included in the defining characteristics of verbs.

Our approach is quite different; instead of having a pre-defined tree for the choice of verbs, the concept dictionary traverses dynamically the necessary subtree that is found under the ingredients being used. It cannot decide what to say without looking at the consequences related to the ingredients. For example, putting the sugar in hot milk suggests the verb "dissolve", not putting pepper. This is true because of the characteristics found in the concept dictionary under the entry PUTWITH and under SUGAR:

```
PUTWITH     ACTION (X Y)... action with 2 parameters (sugar & milk)
            CREATE ACTION ADD ... allocate a conceptual structure
            CA<- "ADD" (in Arabic) ... fill slot 1
            PELT1 <- X PELT2 <- Y ... fill slots 2 and 3
            MIX X Y.....go see what happens when mixing x and y
            :CONSEQUENCE
            MODIFLOC X to Y.LOC ...move x to same location as y
SUGAR :DEF "Sugar" (in Arabic) ... slot 1
            :Mix
            If Y.PQUALIF (CONS = 1)....if y is liquid
            THEN IF EQUAL Y.LOC 1....if ingredients are on stove
                  THEN CREATE ACTION DISSOLVE
                        CA <- "Dissolve" (in Arabic)
                        PELT1 <- X
                        PELT2 <- Y
                        ELSE WAITFOR ACTION STIR......expect stirring
                              CA <- "Stir" (in Arabic)
                              PELT1 <- X
                              PELT2 <- Y
                              ENDIF
                        ENDIF
```

The entry PUTWITH will branch to the ELEMENT X (sugar) by the instruction MIX X Y to see whether it can find additional information that is specific to ingredient X.

This is one of our system's ways of handling lexical semantics and lexical choice. Before it decides to "use" a verb it waits to see what's around it. If the basic action is: PUTWITH PEPPER VINEGAR it will go to the entry PEPPER where it finds that it can choose the verb "to pepper". If we have PUTWITH SUGAR MILK it will branch to the entry SUGAR where it is told to test whether Y (here MILK) is a liquid and is hot, so it chooses the verb "to dissolve" or, if not hot, to wait for the action STIR before choosing "to dissolve".

In both cases (pepper or sugar) it recognises the possible equivalence between the verb "to add" and the newly found verbs, "to pepper" in one case and "to dissolve" in the other. It keeps all candidate words (verbs) and waits for the activation of the stylistic rules.

The entries in the word dictionary are words not concepts. It contains grammatical information necessary for the final realisation of the words and sentences. For example the word "mixture" (in Arabic) has the following entry:

MIXTURE Noun Masculine Singular Irregular

3.3 The grammars

Three modules concentrate upon natural language grammar.

a) The syntactic graph:

This graph is the Arabic or French version of Woods' ATNs. It is traversed in parallel with the semantic network where it recognises the ACTIONS and CHANGES-OF-STATE as verbs, the ELEMENTS and UTENSILS as nouns, and QUALIFS as properties associated with the elements.

The syntactic graph is read as data from a file and is used through its specialised interpreter. For example, one rewrite rule in the Arabic syntactic graph file is read literally as: S -> ?V <VP> | ?N <NP>. The syntactic graph interpreter checks for the presence of a verb when it "sees" ?V, and if that is the case it continues to a verb phrase when it "sees" <VP>. Otherwise it jumps after the sign |. There it sees ?N, and recognises that it should check for a noun in order to continue a noun phrase , i.e. <NP>. This rewrite rule states that a sentence is a verbal phrase when its head is a verb, or that it is a noun phrase when its head is a noun. The Arabic language accepts a noun phrase as a legitimate sentence though it has no explicit verb!.

The fact that the syntactic knowledge is "read" into the system facilitates shifting into another language. The system interpreter stays the same; one changes only the rewrite rules description in the syntactic graph file. For example, in the case of French, the syntactic graph file would contain literally the following line: S -> <NP> + <VP>; which means that a sentence is a noun phrase followed by (+) a verbal phrase.

The conditions associated with the arcs, however, are not sufficient to choose a unique path in the graph i.e. a unique string for the sentence structure. The choice of the desirable path is commanded by the stylistic rules (discussed later).

b) The morphology modules:

These modules are language dependent.

b.1) Arabic: It can simulate the Arabic grammar that is very systematic. One can construct nouns, adverbials, adjectives, actors, etc. from the form (or rhyme) of the radical of the verb. This is an important and a handy feature for sentence generation in Arabic.

Moreover, adjectives can be treated similarly to verbs. Actually in our program they stay as verbs, and when necessary they are transformed into adjectives. For example, "heat the milk" can be converted to "hot milk". The same assertion is true for adverbials. Another issue concerns classification: verbs in Arabic do not need to be assigned to classes according to allowable patterns of surface strings. They naturally belong to (or have) a rhyme class. Changing their rhyme automatically changes their syntactic (grammatical) function (which is reflected by a different path in the syntactic graph); hence a different surface string is generated.

b.2) French: The French language does not have a systematic way of constructing nouns, adverbs, adjectives, and actors according to the rhyme of the verb, as is the case in Arabic. For this reason, the morphology module is very different.

c) The dictionaries of surface string words:

They are consulted when the program is traversing the syntactic graph. They contain information necessary for the morphology module or for choosing paths along the syntactic graph.

4 Varying the output

The stylistic rules are the system's means of varying its output. They are specific to the generation module; they are read, also, as data and they are used through their specialised interpreters. These rules are language dependent and they command the choice of paths along the syntactic graph. Thus they enable the system to generate a variety of texts for the same recipe. Some of these stylistic rules are called aggregation rules by Dalianis and Hovy.

Rule 1: if HISTORY NOUN then ATTACH ADJECTIVE NOUN eif.

This rule allows adjectives to be attached to those elements (nouns) that were subjected to actions (verbs) with a durable effect. For example instead of saying "melt the butter then add the eggs to it", this rule allows the system to say "add the eggs to melted butter".

Rule 2: if REF NOUN then SUBSTIT NOUN PRONOUN eif.

This is the pronoun substitution rule. If a noun has just been referenced then the generator can use the pronoun. "Measure the sugar. Add it to the milk" (Arabic syntax).

Rule 3. if REF INDOBJ then INVERT DIROBJECT INDOBJECT eif.

This rule allows the inversion of the direct and indirect objects if the latter has just been referenced. For example instead of saying "beat the eggs and add the sugar to them" it can say "beat the eggs and to them add the sugar". (Acceptable Arabic word order). It is similar to the French "y ajouter le sucre".

Rule 4. if INORDER NOEND EXPEND then INSERT "until" eif.

This rule checks whether the first of two consecutive verbs reflects no end (such as 'to stir') and whether the second verb expresses some kind of an end (such as 'to thicken'). In this case it allows the insertion of 'until ', and says "Stir until thickened". The lexical semantics details indicating, for example that the verb "has no end" are found in the concept dictionary.

Rule 5. if EQUAL GENERAL 1 then SUBSTIT EQUIV VERB eif.

This rule allows the system to use a more "general" word instead of the more specialised one. For example it can say "add the pepper to the vinegar" instead of saying "pepper the vinegar".

Rule 6 & 7. if SAME VERB then FACTORIZE VERB eif

if SAME INDOBJ then FACTORIZE INDOBJ eif

This rule allows the system to remove redundant verbs and redundant indirect objects. For example "add sugar milk, add chocolate milk" would be realised as "add the sugar and the chocolate to the milk".

Scott and de Souza call these rules 'embedding', Kempen calls them 'forward or backward conjunction', Dalianis calls them 'compacting', and Horacek calls them 'grouping motivated by structural reasons'.

Rule 8. if INORDER EXPEND QUICK then INVERT VERB1 VERB2

INSERT "when" eif.

This rule checks whether the first of two consecutive verbs expresses some ending event (boil) and whether the second is quick (add). In this case it inverts them and inserts the word "when". For example, instead of saying "boil the water, add the flour to the water" it prefers saying "Add the flour to the water when it boils".

Rule 9. if AMBIGUOUS NOUN then ATTACH ADJECTIVE NOUN eif.

This rule checks whether a noun (ingredient or utensil) is ambiguous and then adds an adjective in order to eliminate this ambiguity. For example in a recipe where half of the eggs are beaten and half are not, the system is forced into saying "the beaten eggs" in one case and "the unbeaten eggs" in the second.

Rule 10. if INORDER EXPLOC STATECHANGE then INSERT LOCPREP eif

This rule checks whether the first of two consecutive verbs expresses a location like 'put in chocolate double boiler' and the second one is a change of state like 'melt the chocolate', therefore it decides to say "melt the chocolate in a double boiler".

All of the above stylistic rules are associated with activators that can be toggled on or off. The most concise text is generated when they are all in action.

5 Using this generator in an MT system

The system as has just been presented respects only the interlingua paradigm. It is able to remember the meaning and then generate a variety of texts upon varying the activation of the stylistic rules. But somewhere during the analysis of the French text and then the regeneration of the Arabic recipe, the number of sentences and their form is lost. The system is able to "produce" many forms, but is unable to "remember" which one was the original form.

Some alternatives have been tried for the sake of more "fidelity" to the original French text. These are:

a. Using the semantic network produced during the analysis phase as an interlingua rather than the list of basic actions that is completely blind to original sentence structure.

b. Grouping the basic actions, produced during the analysis phase, according to the original French sentence that produced them. Thus the interlingua passed between the two phases is a number of lists of actions (corresponding respectively to the sentences) rather than just one long list of actions.

Both of these two trials produced sentence to sentence translations.

c. Every time the analysis module passes through a path on the French syntactic graph that corresponds to a stylistic rule, it memorises the rule type to be used with the corresponding concept. Suppose it analysed "y ajouter le sucre", it activates the inversion rule for this sentence.

This is a way of peaking back to the sentence form. It is a form of transfer between the syntactic graph of the French recipe and the final Arabic text.

A system enriched with such a capability would not lose any of its "meaning" processing. It only calls upon more syntactic "history" of the original text.

Reconsidering our first example where the original sentence says "ajouter le sucre au lait chaud", the following rules would be activated:

- "Add" is a more general verb than any other "PUTWITH" verb so rule #5 is activated.

- "chaud" forces the analyser to pass through the adjective path on the syntactic graph so rule #1 is activated.

With these two rules activated the Arabic generator is "forced" into the corresponding wording. Therefore the first and last forms that were possible, in our earliest example, are frozen and the system would generate in Arabic: "Add the sugar to hot milk".

In conclusion, this paper proposed a number of ways for parametering the choice of words in text generation. It explained how a variety of languages, a variety of sentence lengths and structures, and a variety of more or less concise wording can be achieved. In a computer assisted MT environment, the user can be given the choice of which realisation he/she considers best. On the other hand, in a fully automatic MT system, more meta-rules should be included that can measure the "degree of expressiveness" of a realisation or also that can give a "measure of distance" between the wanted text and the produced one.

References

Bouillon, P. & Clas, A., (1993) "La traductique, Etudes et recherche de traduction par ordinateur", Presses de l'Université de Montréal.

Cordier, M. O. et C. Moghrabi (1982) "Towards a more efficient automatic translation", Proceedings of the 1982 European Conference on Artificial Intelligence, Orsay, France, 228-231.

Dahl, V. & H. Abramson (1984) "On Gapping Grammars", Proceedings of the Second International Conference on Logic, Université d'Upsala.

Dalianis, H. & Hovy, E. (1993) "Aggregation in Natural Language Generation", 4th European Workshop on Natural Language Generation, Pisa, Italy.

de Finney, J. et C. Moghrabi (1988) "L'enseignement structuré de l'argumentation écrite", Comptes rendus de la conférence conjointe IBM-Université Laval sur les technologies de l'information.

de Finney, J., C.Moghrabi et P.Tarau (1987) "PARDA: a Parser and Text Generation System for Argumentative Discourse", Proceedings of the International Conference on Computer Assisted Learning in Post-Secondary Education, Calgary, 85-91.

Horacek, H. (1992) "An integrated view of text planning" in Aspects of Automated Natural Language Generation, Dale, R. et al (eds), Springer Verlag Lecture Notes in Artificial Intelligence, pp. 193-227.

Hutchins, W. J. & Somers, H. L. (1992) "An Introduction to Machine Translation, London, Academic Press.

Kempen, G. (1991) "Conjunction reduction and gapping in clause-level co-ordination: An inheritance based approach" in Computational Intelligence, vol. 7, no. 4, pp. 357-360.

McKeown, K. R., (1985) "Discourse Strategies for Generating Natural Language Text", Artificial Intelligence, 27, 1-41.

Moghrabi, C. (1985) "A Multi-Language Generation System", in Conference Proceedings of the APICS Computer Science, 127-130.

Moghrabi, C. (1980) "Un programme de génération conceptuelle de l'Arabe: applications aux recettes de cuisine", thèse de doctorat, Université Pierre et Marie Curie, Paris.

Rousselot, F. (1985) "Realisation d'un programme comprenant des textes en utilisant un formalisme unique pour représenter toutes les connaissances nécessaires", Thèse de doctorat, Université Pierre et Marie Curie, Paris.

Schank, R. (1975) "Conceptual Information Processing", Fundamental studies in Computer science, vol. 3, North-Holland.

Simonin, N. (1985) "Utilisation d'une expertise pour engendrer des textes structurés en français", Thèse de doctorat, Université Pierre et Marie Curie, Paris.

Scott, D. & de Souza, C.S. (1990) "Getting the message across in RST-based text generation" in Current Research in Natural Language Generation, Dale, R. et al (eds), Academic Press Limited, pp. 47-73.

Van Beek, P. et Cohen, R. (1986) "Towards User Specific explanations from Expert Systems", Actes de la sixième conférence canadienne sur l'intelligence artificielle, Ecole polytechnique de Montréal, Montréal.

Woods, W. A. (1970) " Transition Network Grammars for Natural Language Analysis", Communications of the ACM, vol. 13.