An ATMS Approach to Systemic Sentence Generation

Hasan Kamal and Chris Mellish

1 Department of Computer Science, University of Bahrain, Bahrain
hasank@itc.uob.bh
2 Department of Computing Science, University of Aberdeen King’s College,
Aberdeen AB24 3UE, United Kingdom
cmellish@csd.abdn.ac.uk

Abstract. This paper introduces a new NLG architecture that can be sensitive to surface stylistic requirements. It brings together a well-founded linguistic theory that has been used in many successful NLG systems (Systemic Functional Linguistics, SFL) and an existing AI search mechanism (the Assumption-based Truth Maintenance System, ATMS) which caches important search information and avoids work duplication. It describes a technique for converting systemic grammar networks to dependency networks that an ATMS can reason with. The generator then uses the translated networks to generate natural language texts. The paper also describes how surface constraints can be incorporated within the new architecture. We then evaluate the efficiency of our system.

1 Introduction

Given the wide range of applications NLG systems might be part of, it is very important that they master their job in an aesthetic and sophisticated manner. Some applications may require that the generation component produces text with certain rhyme, alliteration or even poetic aspects. Such tasks may require redoing syntactic and lexical choices under constraints from different levels.

In this work, surface stylistic constraints (SSC) are those stylistic requirements that are known beforehand but cannot be tested until after the utterance or (in some lucky cases) a proper linearised part of it has been generated. For example, the French pronouns le, la cannot precede words starting with e. When that happens, both are abbreviated to l’. Now, if we want to generate (in French) an unambiguous utterance, the choice between the feminine pronoun la and Sarah depends on the next word. Although simple, this example shows that there are cases where generators cannot make a final decision on lexical choice until after the surface form has been linearised and its words inflected.

Another example is text size limits. Some lexical choices result in longer utterances because of the way in which each word packages information. The cumulative effect of such verbose choices can be longer texts. However, the exact length of text is not known until after the text is generated and only then can it be compared to the size limit it is allowed to occupy. In the STOP project,
Reiter discusses how even things like punctuation, inflection, and font type can play a role in keeping the text within the allowed limit [10].

Current systemic generation algorithms are prone to surface stylistic problems because they need to make decisions at different choice points before the surface form or part of it has been built. When there is not enough information to make a decision, current generators resort to one of two strategies: selection of a default or selection of a random alternative. [5] show that neither strategy guarantees problem-free surface forms, as “the default choices frequently are not the optimal ones” and the alternative of randomized decisions entails “the risk of producing some non-fluent expressions”.

This paper approaches the crucial notion of choice differently to previous systems using SFL. It relaxes the choice process in that choosers are not obliged to deterministically choose a single alternative allowing SSC to influence the final lexical and syntactic decisions. On the other hand, instead of introducing into the system the well-known problems of backtracking [4], we use a Truth Maintenance System (TMS) to efficiently manage the choices and their dependencies.

2 The ATMS Framework

A Truth Maintenance System (TMS) is attached to a problem solver so that it can focus on the particulars of the given task and the TMS on the bookkeeping of beliefs and assumptions [4]. The problem solver passes the TMS “reasoning information”. In the case of the Assumption-based Truth Maintenance System (ATMS), this reasoning information comes in three kinds: nodes, justifications and no-goods. A node is associated with an instance of a data structure which is being manipulated by the problem solver: a problem-solver datum. The actual content of this problem-solver datum is of no interest to the ATMS. It is the problem solver which requests the ATMS to create an autonomous node, thus informing the ATMS it is reasoning with the associated data.

A justification is a statement indicating that the truth of a conjunction of nodes is sufficient to conclude the truth of a node. Or, a justification can be defined as an implication equivalent to: \( n_1 \land \ldots \land n_k \rightarrow n_x \), where each \( n_i \) is a node. Some nodes, as decided by the problem solver, are called assumptions. The assumptions are the nodes on which any datum ultimately depends. They are considered true until proven false.

Evidence against the presumed truth of assumptions comes in the form of no-goods. A no-good is a set of nodes which cannot all be true at the same time. More precisely, it is a conjunction of nodes which is impossible: \( n_1 \land \ldots \land n_k \rightarrow \bot \), where each \( n_i \) is a node. Often, falsity is represented by a specially constructed node \( \bot \).

As nodes, justifications and no-goods are added, the ATMS maintains a label for each node. A label is the set of environments, representing the disjunction of those environments, which supports the associated node. An environment is a set of assumptions representing the conjunction of these assumptions. If an environment \( E = \{ a_1, \ldots, a_m \} \) is in the label of a node \( n \), the ATMS has deduced