Domain-oriented two-stage aggregation: generating baseball play-by-play narratives

James Baldwin[‡] and Songsak Channarukul[†]
Department of Computer Science
Assumption University
Bangkok, Thailand
Email: [‡]bravamicus@yahoo.com, [†]songsak@scitech.au.edu

***Abstract*— This paper presents an end-to-end natural language generation system that performs aggregation in two stages: the first takes advantage of the information implicit in the source knowledge base in order to aggregate event components into complex sentences. The second stage examines the developing context of the text in order to aggregate similar adjacent events into more fluent text. The source knowledge base is the Retrosheet collection of play-by-play baseball scoresheets encoded in machine-readable form. The output is reasonably fluent and natural, human-readable play-by-play narratives of historical baseball games. The system was tested against all regular season major league games played from 1950 to 1969, taking less than a second to produce three to five pages of text for each game. The aggregation achieved resulted in a substantial improvement in native speaker judgments of fluency and readability.**

Keywords - natural language generation; aggregation

I. INTRODUCTION

In natural language generation, aggregation is a catch-all term that attempts to encompass such linguistic processes as coordination and subordination, along with various forms of ellipsis to eliminate redundancies. The goal of aggregation is variously seen as improving the fluency, conciseness, coherence and cohesion of the output text [1].

Where and how these processes can best be performed in the generation process remains uncertain. Aggregation has been implemented or proposed in each of the three Reiter and Dale consensus architecture stages: document planning, microplanning and realization [2]. Furthermore, within their central microplanning stage, aggregation can be found before, after and between lexicalization and the generation of referring expressions. Nor is it clear that a single monotonic module is desirable. In order to provide a mechanism for more complex feedback or revision based approaches, the RAGs implementation architecture [3] gives aggregation its own component that may be visited (repeatedly if desired) via the Objects and Arrows System whiteboard interface [4].

Likewise, approaches to how aggregation should be performed vary widely. On the one hand, theoretically oriented researchers have employed Discourse Representation Theory [5] or Centering Theory [6] to inform their approaches, which results in a close linkage between aggregation and text planning. At the other extreme, generation has been approached as a discrete optimization problem [7], which avoids theoretical commitment altogether in favor of conciseness. Between these extremes, there is a large body of research literature, but little consensus has emerged. In recent years, corpus statistical and machine approaches to generation have become popular. However, such "data-driven" systems rely on the existence of extensive corpora of model outputs, which may or may not be available. For the domain to be discussed here, such model outputs do not exist.

This uncertainty about how and where to perform aggregation arises in part from the fact that coordination and subordination are both constrained and motivated by a tangle of syntactic, semantic and pragmatic factors. Where one elects to deal with aggregation depends on which factors are deemed more important, based largely on one's theoretical predisposition and what shortcomings one is willing to overlook. Definitions of aggregation are usually couched in terms of messages or propositions, revealing an assumption that the underlying knowledge base provides its information as some kind of atomic logical formulae that will need to be combined appropriately. David McDonald has recently expressed skepticism that this kind of definition is coherent, noting that "it might turn out that this is an artifact of the architecture of today's popular text planners and not at all a natural kind, that is, something that is handled with the same procedures and at the same points in the processing for all the different instances of it that we see in real texts." [8].

This paper presents the baseball play-by-play narrative generation system, named "BABBAGE" (for Babbage's Automatic BaseBall Aggregated Generation Engine). In our system, we have retreated from a theoretical, "domain independent" approach to aggregation. Instead, we have turned to the resources available within the source knowledge base, and to the stylistic desiderata for the output text, to guide our approach to aggregation. For the first of these, it became clear that it was necessary not only to decode the densely coded events surrounding each batter's turn at the plate, but also to maintain a record of the on-going state of the game. For example, the score, the number of outs, and the identities of the

978-1-4799-6049-1/15/$31.00 ©2015 IEEE