Building a Large-scale Commercial NLG System for an EMR

Mary Dee Harris
Catalis, Inc.
7801 N. Capital of Texas Hwy., Ste. 260
Austin, TX 78731
mdharris@thecatalis.com

Abstract

Natural language generation technology is mature enough for implementing an NLG system in a commercial environment, but the circumstances differ significantly from building a research system. This paper describes the challenges and rewards of building a commercial NLG component for an electronic medical records system. While the resulting NLG system has been successfully completed, the path to that success could have been somewhat smoother knowing the issues in advance.

1 Introduction

In 2002 I was hired by a small start-up company to add narrative generation to their electronic medical records (EMR) system under development. After six months, we had a first-cut system producing narrative based on a doctor's selection of items from the graphical interface during an encounter with a patient. This paper describes the rewards and challenges of building such a system in a commercial environment, in hopes the lessons I learned can contribute to successful future commercial systems for natural language generation.

The company has always been funded by investment money with some recent revenue income. The founders were both medical doctors, with little corporate experience and even less knowledge of technology. They had the vision; the rest of us did the work. The company's product is a general-purpose EMR system on a tablet PC with handwriting recognition and extensive graphical representation of human anatomy. Its foundation is an elaborate database of medical content, outlining specific requirements for information collection. Much of this medical information is arranged in templates, one for each complaint. When a patient comes into the doctor's office complaining of chest pains, the template for Chest Pain provides the appropriate selections for the doctor to record pertinent information related to that condition. Other parts of the system deal with physical examinations, procedures, prescription of medication, orders for lab tests and procedures, and so on.

My mission was to implement a narrative generation system to record the doctor/patient encounter, following the traditional narrative created by the doctor's dictation, which is then transcribed into a narrative. These narratives serve as a legal record of the encounter and are used in court in malpractice suits. Thus the narrative is an extremely important part of the patient record and must be complete and accurate. Otherwise the doctor – and our company – could be liable for malpractice.

2 Challenges

The challenges of designing the narrative system were many. The narrative must be completely accurate to avoid liability. While the initial targets were small practices and clinics, the system would expand into larger clinics and hospitals. So the system had to be scalable. The scope of the project also had to be scalable. Beginning with hundreds of medical templates often with multiple names,

there are now thousands for many specialties with different requirements for format and style.

Another challenge was the naïveté of the company and its staff. The CEO had a grand vision, but little concept of the technology for language generation. He believed the automated narrative was possible, but there was little understanding of the extent of commitment of staff, time, and money for building such a system.

One less obvious challenge is the difference between research and commercial applications. Our limited finances allowed us few available commercial products. However, the freely available resources that academics rely on were usually available *only* for research. In our field of health care technology, the UMLS was the primary resource available to us. The Unified Medical Library System[1] developed at the National Library of Medicine has resources which include a medically-oriented dictionary of English called SPECIALIST and tools to access it, a semantic network related to health care, and a Metathesaurus -- "a very large, multi-purpose, and multi-lingual vocabulary database that contains information about biomedical and health related concepts, their various names, and the relationships among them." This lack of resources was a mixed blessing: all our tools and program components belong to the company with no financial or licensing strings attached.

One usual way to start any NLP project is to acquire a large sample of the texts to be processed. However because of privacy issues, we had no "live" data representing medical narratives. The two doctors wrote some imaginary scenarios to serve as samples and provided feedback on our guesses of what the medical language should be. So the project started with no outside resources, little support, no samples, but a lot of enthusiasm.

3 Plan of Attack

Despite the challenges, I knew NLG technology would be able to fulfill the requirements for this application so the planning began. The original idea was to follow a standard development model: proof-of-concept (POC) system, prototype, and production system. The POC would have simple sentences and a restricted vocabulary, but enough to convince the company that the technology could provide a feasible solution. The prototype would extend the capability, adding a grammar and a more extensive vocabulary as well as being robust enough to handle more data. The POC was built in about six months with a Java developer assisting me with the design – it sounded like a second grader had written the narrative, but it was accurate and proved that we could do it.

The prototype never got built due to management decisions and some bad luck. We had no internal staff to devote to the task. To make matters more complicated, Steve Shipman, the original Java developer who knew some computational linguistics, was replaced by a developer with no real knack for linguistics whose English was a second language. I had to teach him the linguistic terminology and the language structures before he could write the code to handle them.

The next problem arose when the management saw the narrative output -- simple as it was -- and immediately started adding templates for us to handle. Despite my protests that it wasn't ready for deployment, we had to add additional features such as aggregation and negation to this simple-minded version. It took several years before we got the go-ahead to write the full-blown system, by which time we had several thousand templates in the system. We finally spent six months on the new system, followed by nearly a year of testing. Because the POC had been put into production, we had to establish a dual model that ran both old and new versions. We are still trying to get all the original parts converted to work on the full-blown production system so we can eliminate the POC section.

4 System Architecture

The architecture of the Narrative Engine followed the basic design described in Reiter and Dale (2000) for an NLG system, with adaptations to fit our data model. Because the narrative output had to be so accurate and the style so sophisticated to satisfy the physician client base, I doubted that completely automated generation would be suffi-

[1] http://www.nlm.nih.gov/research/umls/

cient. So following the lead of machine translation, I chose to implement human-assisted computer generation. That seemed the only appropriate approach, used similarly by CoGenTex in their Exemplars method (White and Caldwell, 1998). We considered using Exemplars, but they are Java-based which was not appropriate for our situation. Most of our NLG staff didn't know Java since we hired liberal arts and linguistics majors.

We developed a plan language called Linguo, after Lisa Simpson's robot by that name. Linguo helps us write plans to describe the translation from medical findings for a particular patient into appropriate medical language for an encounter. The plan writers select the predicate best suited for each finding in a template. That predicate then determines the semantic structure, following Jackendoff (1990; 1991). These plans are generalized to handle many similar findings, rather than being a one-to-one translation. The basic design for the Narrative Engine held up well through the various implementations, with only minor adjustments required. [2]

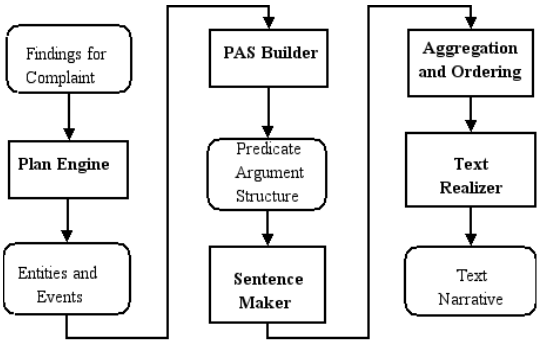

Figure 1 Architecture of Narrative Engine

5 Proof of Concept

The initial POC was string-based, for speed of development – a decision I would come to regret, but probably necessary to get the project underway. The POC system had no separate grammar, but handled syntactic and morphological issues in the code. We created two XML files for the translation into English: predicate templates and clause templates. The predicate templates define the semantic roles for each predicate in our restricted vocabulary, while the clause templates match the semantic forms to syntactic structures. The final stage of processing was the Sentence Realizer that converted the syntactic structures into English. The Narrative Engine was a separate module in the application that received medical findings (the individual items that the doctor selected) and that output English text to the application for the note.

5 Commercial Development

The commercial version of the Narrative Engine was called Component-Based Processing (CBP) to distinguish it from the string-based POC. We now had two lexicons: a semantic one containing the finding names for all the medical templates and the UMLS SPECIALIST syntactic lexicon. We added bracketed forms to specify language structures for each finding name, to go with the basic syntactic information. For example, the phrase "the right shoulder" would be represented as [np [det the][adj right][n shoulder]]. We hired a computational lexicographer, Ken Litkowski, to help produce the bracketed forms for the 60,000 plus entries. Creating these forms was not trivial since the finding phrases vary from a single word to a complex noun phrase to a complete clause.

Using the bracketed forms allowed us to extend our aggregation capability to a linguistically solid method of analyzing the component structure to identify corresponding parts to coordinate. We also added a means for asymmetric aggregation, known as *hypotactic aggregation* (Shaw, 2002). Besides being able to coordinate similar items, as

The patient described the pain as sharp and throbbing,

we can now combine dissimilar findings, as in

The skin was closed with 14 2-0 monofilament sutures using continuous stitch.

One major addition to the Narrative Engine was the syntactic grammar set up as a properties file allowing modification of the grammar without code changes. We can test new features easily and

[2] This paper does not detail the technology specifically as it is proprietary. The company has patent applications pending for much of the design. This is another important contrast with the research community where sharing ideas is the norm.

try extensions to the language with no impact on the overall system. The clause templates used in the POC were replaced with verb templates since much of that work was now handled by the grammar. Verb templates describe the alternation patterns (Levin, 1993) and include the irregular forms.

Another change was the integration of the narrative process into the application more completely. This integration was not simply a code change, but a change in perception of the project as a whole. The company came to understand that adding the narrative capability had increased the value of the EMR in the marketplace. At this time, ours is the only EMR with real natural language generation, not handled by templates or canned text.

6 Recommendations

Many of the choices made during development of this system would be changed, if I had the luxury of starting over. I would like to offer up some suggestions for others to avoid the difficulties I faced. Consider these ideas before you start.

- Educate your clients. Your clients are mostly in your own company. Not everyone is going to understand the importance of the work and the need for resources or have the basic linguistic knowledge to comprehend the requirements. I gave many tutorials to help our staff understand what we were doing.
- Be clear about the costs of building the NLG system. Your estimates will be wrong, almost by definition, but you have to start somewhere. Since the uninitiated cannot imagine the potential until they see it, they will have many more ideas of how to apply the technology once they see it, thus extending the requirements. Here again, educate the company regarding the staff requirements (developers, linguists, quality assurance, marketing) and the training needed to make them productive. Consider the data development requirements as well as the coding.
- Be careful how you plan out the development stages. A proof-of-concept system is a great idea to demonstrate that the technology is feasible, but it is tempting to take it and run with it. You need to build the system in stages, but make sure the staging is spelled out in advance with an understanding from management of the process.
- Deployment and customer acceptance is the goal, not completion of the code. We found that the customers were gratified by the ability to eliminate the dictation and transcription process, but they do have opinions about the wording sometimes. We work with specialists to develop the medical templates and the narrative before we implement.
- Remember that a commercial system is not cutting-edge technology, no matter what management thinks. A commercial system should use time-proven, reliable methods robust enough for inevitable modifications. Some features will be untested, but the basic foundation of the system must be reliable.
- Make sure you have the funding to complete the project. A champion within the company can help fight your battles.

None of these recommendations should preclude anyone from trying to build a large-scale commercial product, but knowing in advance where the pitfalls lie can ease the process. It takes more than a good idea and a knowledge of the technology to make it work, but the effort can be worthwhile in the end. The language component of our EMR system has helped doctors increase their ability to see more patients by reducing the time required to take notes, dictate them, and pay for their transcription. So the doctors appreciate the automated narrative capability even though they have no idea how it is accomplished.

References

Ray Jackendoff. 1990. *Semantic Structures*. MIT Press.

Ray Jackendoff. 1991. Parts and Boundaries. *Lexical and Conceptual Structures*. Blackwell.

Beth Levin. 1993. *English Verb Classes and Alternations: A Preliminary Investigation*. University of Chicago Press.

Ehud Reiter and Robert Dale. 2000. *Building Natural Language Generation Systems*. Cambridge University Press.

James C. Shaw. *Clause Aggregation*. Dissertation, Columbia University. 2002.

Michael White and Ted Caldwell. 1998. Exemplars: Practical, Extensible Framework for Dynamic Text Generation. *Proceedings of the Ninth International Workshop on Natural Language Generation*.