Proseminar: Natural Language Generation for Virtual Environments. Lecturer: Michael Roth <mroth@cl.uni-heidelberg.de>

Content Selection in Natural Language Generation Systems

Michael Haas <haas@cl.uni-heidelberg.de>

April 16, 2010

Part I.

Introduction

In this seminar report, I will start with an introduction which defines some common concepts. Then I will introduce two papers on the issue of *Content Selectio*n in NLG systems. A third approach to *Content Selection* is then described which is used in our contribution to the GIVE-2 challenge. In the conclusion, I'll compare this system against the two other systems.

Michael Roth's seminar on "Natural Language Generation for Virtual Environments" introduced the students to the domain of Natural Language Generation. Presentations were given and a NLG system for participation in the GIVE-2 challenge [Koller et al., 2009] was built.

This report deals with the issue of *Content Selection* in Natural Language Generation systems. The task of Natural Language Generation is defined by [Reiter and Dale, 1997] as follows:

> Natural language generation (NLG) is the subfield of artificial intelligence and computational linguistics that is concerned with the construction of computer systems that can produce understandable texts in English or other human languages from some underlying non-linguistic representation of information.

Typically, some kind of information needs to be communicated to the user. This can be routing directions, a train schedule or the weather forecast. NLG systems can also act as authoring aids, e.g. for documentation writers.

NLG can be decomposed into the following six subtasks according to [Reiter and Dale, 1997][1]:

Content Determination (aka Content Selection)
Appropriate content needs to be selected from the input data. Sometimes, all input data needs to be communicated. In many cases however, it is the system's responsibility to filter and summarize the data into a set of messages. The user needs to know only what is relevant for them.

Discourse Planning
The selected set of messages lacks structure. *Discourse Planning* orders the messages according to an underlying discourse structure. Consider real life examples of text, e.g. newspaper articles, which always have some kind of internal structure. Good ordering makes a text more coherent and easier to read.

Sentence Aggregation
The ordered messages may now be grouped together into sentences or paragraphs. This can improve readability.

Lexicalization
For the selected abstract concepts, appropriate words need to be chosen. This can be as simple as providing a manually generated mapping from concepts to words. Using a set

[1]For a comparison of the de facto standard architecture with psycholinguistic theories, see [Reiter, 1994]

of synonymous words for variation can improve fluency. *Lexicalization* is also important if the system needs to to generate output in multiple languages.

Referring Expression Generation

Generation of Referring Expressions is closely related to *Lexicalization.* Both tasks are concerned with finding appropriate surface realization expressions. However, referring expressions are used to discriminate between domain entities (e.g. two different trains). In case of our GIVE system, referring expressions are used to refer to objects in the virtual world which the user needs to manipulate. Simply providing a hand-generated mapping proved insufficient as ambiguity needed to be resolved by relating the object in question to other objects in the virtual world.

Linguistic Realisation

The *Linguistic Realisation* applies the rules of grammar and orthography of the target language to construct correct natural language text. Morphological rules need to be applied to the words. Syntactical rules are applied to order the inflected words.

Part II.

Systems

1. Using Discourse Strategies to produce draft scientific summaries

This chapter describes [Rino and Scott, 1994]. Their work is concerned with the automatic generation of draft summaries from scientific papers. Based on an analysis of both existing abstracts and newly produced summaries by domain experts, they develop heuristics based on a discourse model which allows them to filter out unwanted information.

N.B.: The context of their work is text summarization. Traditionally, text summarization uses Information Retrieval techniques to gather relevant information from text [Reiter et al., 2005]. The definition given by Reiter and Dale [1997]requires using abstract data as input as required by the definition shown above. However, this paper describes an interesting approach for content selection based on linguistic theories which can be adopted for traditional NLG systems.

Basically, this paper shows a method of transforming an indicative text into an informative one. According to [Hutchins, 1987], there are three types of summaries:

Indicative summaries: these have a referential function, as they only record essential topics without going into details

Informative summaries: these can act as substitutes for the original document as they contain relevant details

Evaluative summaries: assessment of the original document, taking into account other work in the field

Evaluative summaries are not considered here.

1.1. Their Analysis

Rino and Scott performed two experiments. Both existing abstracts and newly produced summaries by domain experts were analysed with respect to three perspectives: macro structure, relevant content information and the relation between abstracts and their source texts. Their findings suggest that the abstracts follow the *Introduction-Methodology-Results-Discussion-Conclusion* [Weissberg and Buker [1990], cited according to Rino and Scott [1994][1]] macro structure. As this structure is also employed in the

[1]Unfortunately, the original texts for the citations in this part could not be sourced so I need to rely on Rino and Scott.

papers themselves, they conclude that the macro structure of the abstracts mirrors the macro structure of the papers.

Analysis of the summaries created by the domain experts show a similar pattern. Rino and Scott adopt the Problem-Solution pattern [Hoey [1979, 1983], Hutchins [1977], all cited according to Rino and Scott [1994]] which, at a macro-structural level, is described as *Situation-Problem-Solution/Response-Results-Evaluation*. This is analogous to the *Introduction-Methodology-Results-Discussion-Conclusion* pattern described above. The individual discourse components themselves can be composed from other discourse components. In such a recursive structure, the *Evaluation* can give rise to another *Problem*, which can in turn require *Elaboration* and a *Solution*.

1.2. Discourse relations between structure elements

The heuristics for content selection are based on discourse relations (cf.[Hobbs, 1978]). Some discourse relations given by Hobbs [1978] include:

Cause if a causal chain can be found from the event or state described in the first discourse segment to an event or state given in the second discourse segment

Explanation used to describe an unusual event further by relating it to a normal situation or event

Generalization when moving from an specific statement on a class instance to a statement applying to the class as a whole

Exemplification the inverse relation of Generalization. When moving from a generic statement about a class to a specific class instance.

Elaboration a statement A elaborates on a statement B if a common proposition can be inferred from both statements, with statement A adding more information to the proposition.

This list only serves an example, Rino and Scott [1994] describe even more relations.

When comparing the discourse relations given above with the macro-structural patterns, it is obvious that many of these relations hold between discourse segments. For example, in the *Problem-Solution* sub-pattern, we can find the *Cause* relation as the statement of a problem in a scientific causes the search for a solution. I hypothesize that the *Exemplification* or *Generalisatio*n relations are employed e.g. when comparing results of an experiment to existing theories and that *Explanation* is used when, say, trying to relate unusual experimental results.

1.3. Using discourse relations for text compression

As indicated above, Rino and Scott's goal is to transform informative texts with lots of details into indicate texts where only essential topics are referenced. They call this process "compression". Rino and Scott do not give a concrete definition for this phenomenon;

however, it is easy to adapt Knight and Marcu [2000]'s definition on sentence-level compression:

> Determining what is important in a sentence and determining how to convey the important information grammatically [..]

Now that we have a working definition for the term "compression", we can look at some of the assumptions they adopted for compression in summary generation.

- Some units of information can be omitted from the discourse structure as they are not essential.
- When removing all non-essential units of information from a discourse structure, the resulting summary will be highly indicative.
- When omitting (non-essential) units of information, the text remains understandable even though it may increase the inference burden placed on the user.

Rino and Scott suggest two compression strategies. The first one involves the removal of macro-components, the second one the removal of details by taking into account discourse relations such as the *Elaboration* relation. The first approach is not described any further in their paper. I hypothesize that it is not suited to the task at hand as elimination of macro-components might disrupt the discourse structure in a way which makes the resulting summaries unsuitable for scientific consumption. In their analysis, Rino and Scott find that the macro structure of the summaries mirrors the macro structure of the corresponding documents. Completely eliminating macro components such as *Evaluation* will break this convention. However, my hypothesis is unverified. It is likely that there are discourse structures where removal of macro elements will not hurt, especially when considering different information needs depending on the intended audience.

For the second approach, Rino and Scott show several heuristic rules based on discourse relations.

- Remove examples from *Exemplification* relations. An example sentence:

> This experiment does not include trained personnel, such as graduate students.

becomes

> This experiment does not include trained personnel.

- Remove any optional discourse elements linked to the *Problem-Solution* sequence. This leaves a minimum summary.
- Remove *Elaborations*. If a discourse segment elaborates on another discourse segment, the elaborating discourse segment may be removed.
- Remove *Justifications* while leaving the elements which they were justifying in place.

1.4. Discussion

Rino and Scott show promising heuristics based on a linguistic analysis of a sufficiently large corpus on the one hand and the application of existing discourse theories on the other hand. Their theories are linguistically sound and produce good results according to their evaluation. Within the scope of their paper, their work can be considered successful.

However, some questions remain regarding the applications of these heuristics. The heuristics themselves are simple rules which can easily applied to a data set, given the right data structure. As their work is concerned with text summarization, it is still necessary to transform natural language text into a discourse representation where the rules can indeed be applied. Newer work by Pardo and Rino [2002] utilizes the approach described in this paper; however, their input data is abstract information units as opposed to normal text. Martins et al. [2001] build on Rino and Scott's work as well, but they work on abstract input in UNL (Universal Network Language) format. According to Boguslavsky et al. [2000], an "enconverter" is used to create an UNL representation from Natural Language text. This process is not fully automatic; it was designed to be done with human assistance. A combination of the work by Martins et al. [2001] and the UNL component in Boguslavsky et al. [2000] along with the rest of their ETAP-3 environment in a pipeline architecture might be a good starting point for an actual implementation of the content selection heuristics described by Rino and Scott [1994]. However, the semi-automatic nature of UNL enconversion does not allow a fully automated solution.

Bridging the gap to data-to-text NLG

As noted above, Rino and Scott deal with text summarization, where natural language text is both input and output format. Conversely, the NLG definition given by Reiter and Dale [1997] explicitly demands "non-linguistic representation of information". It can be shown that these heuristics can be beneficial to traditional NLG as well, thus motivating the inclusion of the Rino and Scott [1994] paper in this report.

It might be argued that UNL is a sufficiently "non-linguistic representation" so that existing work by Martins et al. [2001] shows that Rino and Scott's heuristics can be applied successfully in data-to-text tasks (while the actual text generation is outside the scope of Martins et al., ETAP-3 can be used here). If "non-linguistic" is interpreted to refer to anything but natural languages, thus including artificial languages, then the definition of UNL as given by Martins [2010] supports this argumentation:

> The Universal Networking Language (UNL) is an artificial language for representing, describing, summarizing, refining, storing and disseminating information in a natural-language-independent format.

Other applications are also possible. All we need is for the input data of an NLG system to contain the discourse relations used by Rino and Scott [1994]. These need to be available either explicitly as part of the data or easily derivable in a pre-processing module of the NLG system. For demonstration purposes, I will focus on two relations: *Cause* and *Elaboration*.

For the *Cause* relation, we can hypothesize that a weather forecast generation System as described by Reiter et al. [2005] might receive causality information as part of its input data. Weather forecasts may include information as to why a particular weather phenomenon, such as a sudden temperature drop, is happening. Such a hypothetical NLG system could then include or exclude the cause for a particular weather event, depending on the desired informativeness of the resulting message.

Another case where the *Cause* relation is abundant are proof explanation systems such as the one described by Fiedler [1998]. In a theorem prover, deductions are performed based on a set of rules. I argue that a deduction step is taking place because of the existence of a rule. We might exclude such a rule during the content selection step according to the heuristics described by Rino and Scott. *Exemplification* and, to a lesser extent, *Generalisation*, are also common deduction rules and provide a direct equivalent of these discourse relations to link two deduction steps. Whether it is useful to leave out deduction steps when generating an explanation for a proof is debatable, however.

For the *Elaboration* Relation, we can hypothesize a system in the area of Technical Documentation Authoring. In a knowledge base, there might be links between units of information where one unit of information elaborates on the other and where this link is even marked as such, e.g. for term definitions. The unit of information which elaborates on the other unit can then be discarded.

2. SUMTIME: Content Determination for Time-series data

This chapter focuses on two papers originating from the SUMTIME project. The first paper, Sripada and Reiter [2001], describes the two-stage content determination model worked out during the knowledge acquisition phase. The second paper, Sripada et al. [2002], describes the actual content selection algorithms they used and partially builds on the first paper.

2.1. SUMTIME?

SUMTIME "is a research project aimed at developing a generic computational model for producing textual summaries of time series data" [Sripada et al., 2001]. Time-series Data (TSD) is defined as "a collection of values of a set of parameters over time" [Sripada and Reiter, 2001].

The project consists of several parts. SUMTIME-MOUSAM (Sripada et al. [2003a]) is concerned with TSD from Numerical Weather Predictions (NWP); that is, the automatic generation of weather forecasts. The other, SUMTIME-TURBINE (Yu et al. [2003]), deals with TSD derived from readings from gas turbine sensors. I will only report on SUMTIME-MOUSAM due to space constraints. SUMTIME-TURBINE employs pattern recognition techniques based on KBTA (knowledge-based temporal abstraction) to detect anomalies in the incoming sensor data. It is more complicated and less mature than SUMTIME-MOUSAM, which is actually used in production at a weather forecast company (Sripada et al. [2004]).

There is also a third system, SUMTIME-NEONATE Sripada et al. [2003b], which is concerned with report generation from TSD in neo-natal care units. However, it's not as mature as SUMTIME-MOUSAM.

2.2. Theory: two-stage model for content determination

This part is about Sripada and Reiter [2001].

The paper describes a two-stage approach to content determination. The first step is to build an overview of the input data set. The second step uses this overview to create summaries from the TSD which can then be used as input for the rest of the NLG pipeline.

This model was developed during the knowledge acquisition phase of the SUMTIME project. Several methods were employed to gain insight into the domain, e.g. corpus analysis as well as think-aloud sessions and discussions with domain experts. During their discussions with those experts, the authors noticed that they first build a mental overview of the data in question. In this stage, the experts do not consider the user's needs or any informative goal, they simply reason about the data. The authors call this stage *Domain Problem Solving*.

After the expert has gotten an overview and understanding of the data to be summarized, they create the summary using their domain knowledge and personal experience while taking into account a user's needs and interests. This step is called *Communication Reasoning*. For example, a hiker will expect different information than an oil-rig worker in a weather forecast. The authors show an example where the weather simulation can be corrected by the forecaster's experience: if warm wind flows over cold sea, then the wind speed predictions need to be increased to be accurate as the estimates given by the model tend to be too low for this particular scenario.

The authors cite evidence from other projects which supports their theory. In the STOP project, the involved health professionals state that they first create a "mental image" of the subject at hand[Sripada and Reiter, 2001].

Sripada et al. do not give a domain-independent definition of what constitutes an overview. Instead, they consider an overview to be the result of inferences drawn from the input data which in turn trigger content determination rules.

It is also noted that overviews and the texts created from them can be different regarding the underlying ontologies. A meteorologist, for example, uses his domain knowledge to reason about the data, which includes concepts such as atmospheric stability. The resulting forecasts usually do not mention these concepts but instead focus on information which is useful to the end user.

2.3. Application: Segmenting time series for Weather Forecasting

This part is about Sripada et al. [2002].

The paper describes the actual content selection methods for the SUMTIME-MOUSAM project. Two different methods to summarize TSD are described and evaluated. The first method was suggested to the authors by a domain expert and was named *step model*, the second approach uses *segmentation* techniques.

Step Model

The *step model* uses a step value to select individual data points from a time series. The algorithm selects the first data point as anchor point and then goes through the successive data points and checks whether the difference between the data point and the last anchor point exceeds the given step value. If the data point exceeds the step value, then this data point is selected as new anchor point and then the process is repeated. All anchor points are included in the summary while the rest of the points is discarded.

Segmentation

As defined by Sripada et al. [2002], *segmentation* means that a time series is approximated by a number of straight lines, with the number of straight lines being much smaller than the original number of data points in the TSD. Three types of segmentation algorithms are available:

top-down the series is iteratively broken down into increasingly smaller parts

bottom-up neighbouring data points are combined into increasingly larger parts

sliding window extending the segments left-to-right by increasing window size

The authors choose the bottom-up algorithm as it fits the overview stage of their model better. When using the sliding window approach, not the complete data set is considered and thus the overview stage is omitted.

Regarding the number of segments, there are three different methods to formulate a stopping criterion:

- specify the desired number of segments
- ensure that the maximum error for any segment does not exceed a given threshold
- ensure that the maximum error for all segments does not exceed a given threshold

The *Balance of Error* algorithm described by Keogh [1997] will create an “optimum” number of segments. However, the resulting segmentation will only by ideal when considering internal criteria such as the state of the input data set itself. External stopping criteria may impose different requirements. When looking at wind direction, a higher rate of error can be acceptable when wind speed is low; that is, direction changes can be significant or insignificant depending on wind speed. A related issue is that channels are not independent and need coordination. For example, wind text is generated from both wind direction and wind speed. Finally, the user’s preferences and needs must be taken into account.

The researchers create a table where the stopping criteria were realized as thresholds. For wind speed, the threshold is 5 (units are unspecified). The threshold for wind direction depends on the threshold for wind speed, with speeds from 0-15 allowing up to 44 in direction changes before the data point is selected for inclusion while the threshold for higher speeds is 22.

Evaluation

For the evaluation, Sripada et al. [2002] use two metrics. As a measure of information loss, the error introduced by the analysis method is used. The other metric is the size of the resulting summaries, given by the number of wind states in a summary. These metrics are calculated for both the step method and the segmentation method.

The results show that the segmentation method performs better than the step method. Concerning size and error measure combined, segmentation performs better than step in 16.5% of the cases. Conversely, in only 0.56% of the cases the step method is better than segmentation. In 32% of the cases, segmentation is better at size but worse regarding information loss. In 31% of the cases, segmentation loses less information while being on par with the step method concerning size.

2.4. Discussion

The proposed model in Sripada and Reiter [2001] seems solid as it is based on extensive knowledge acquisition work. However, the authors do not cite existing linguistic theories which could serve as an explanation or foundation for their observations. Also, no psycholinguistic approaches or methods from other cognitive sciences have been used to verify their hypothesis. Of course, their work does not claim to be an explanation for language understanding.

At least some models of text understanding support the overview step. Britton and Graesser [1996] describe 12 models of text understanding from different researchers. In their summary (page 243), they describe five metaphors underlying the different models. Three of these five metaphors (1,2 and 5) in particular support the overview process described by Sripada and Reiter:

- "Understanding is the assembly of a multileveled representation"
- "Understanding is the construction of a coherent representation"
- "Understanding is inference generation"

Alas, a quick search did not reveal any previous work which explicitly supports the distinction between *Domain Problem Solving* and *Communicative Reasoning*.

Within the scope of the work - describing a model for content selection - their observations and the derived model are plausible and they seem like a solid working hypothesis.

The second paper, Sripada et al. [2002], partially builds on the first paper. The authors describe their approach of selecting interesting content from time-series data using segmentation techniques. They compare their approach with an approach suggested by a domain expert and achieve notable improvements.

Sripada and Reiter [2001] state in their evaluation that they are building a testbed system called SUMTIME-MOUSAM. In Sripada et al. [2002], we are presented with SUMTIME-MOUSAM. In their work, they do build on the two-stage model introduced earlier. However, the distinction between overview generation and summary generation is not as prominent. In their 2001 paper, they state that *Domain Problem Solving* - the overview construction - is a process different from *Communicative Reasoning*. In SUMTIME-MOUSAM however, both tasks are solved by the same segmentation algorithm. Sripada et al however let their two-stage model guide their algorithm selection. The choice of the bottom-up algorithm is inspired by the overview stage of the model; a sliding-window algorithm does not have an overview on the data. The *Communicative Reasoning* is then done using the threshold values discovered during the knowledge acquisition phase.

3. Content Selection in GIVE - Our approach

3.1. GIVE?

GIVE stands for "Generating Instructions in Virtual Environments" (Koller et al. [2009]). It is a research challenge where researchers submit their NLG systems which are then evaluated in experiments organised by the GIVE team. The evaluation is internet-based; that is, people try out the submitted systems over the internet and their feedback is recorded.

GIVE is a treasure-hunt scenario. The user is put in a virtual 3D environment consisting of multiple rooms. He has to navigate the world and manipulate world state using buttons in order to retrieve a trophy from a safe. The NLG systems created by researchers assist the user by generating instructions which tell the user how to get the trophy.

3.2. GIVE API

The GIVE team provides an abstract class called *NlgSystem* which must be implemented. From my experience, the most important callback methods are *handleStatusInformation*, which is called regularly to provide new information to the NLG system, *handleAction,* which is called when the user performs an action such as manipulating a button, and *handleDidNotUnderstand,* which is called when the user signals to the system that he is lost or did not understand the instructions.

3.3. Our architecture

Our *NlgSystem* implementation roughly follows the three-component pipeline architecture described by Reiter and Dale [1997]. In our seminar, we implemented a *Discourse Planner*, a *Text Planner* and a *Surface Realizer* component. In the following sections, I will describe the *Discourse Planner* component implemented by Duy Trinh and myself and compare our approach to the approaches described above.

3.4. Content Selection (Discourse Planning) in GIVE

The content selection step of an NLG system is always dependent on the structure of the input data. In case of GIVE, the input data is a plan consisting of first-order logic formulas which describes the actions the user needs to perform to obtain the trophy. As given by Koller [2009a], the actions employed by the planner are:

move(p1, p2) The player needs to move from position p1 to position p2.

manipulate(a, s1, s2, p) Click on object a (which is usually a button) to change its state from state s1 to state s2. The user needs to stand at position p.

manipulate-stateless(a, b, s1, s2, p) Click on a stateless object a (which is usually a stateless button) to change the state of object b from state s1 to state s2. The user needs to stand at position p.

manipulate-stateless-a(p) Click on stateless object a. The user needs to stand at position p. This is a special case of the previously mentioned action where the stateless button does not control another object.

take-a(p) Take trophy a while standing at position p.

We can see that there are two types of actions: movement instructions and manipulation instructions. Regarding the positions used as arguments for the predicates: these are not numerical positions as found in a Cartesian coordinate systems. Instead, a discretizer is employed to assign a name to a region which is then used as position argument.

A plan is a series of actions. Usually, there will be a multitude of *move* instructions to transition the user from one region to another until he reaches the correct location to execute a *manipulate* statement. From my experience, there are significantly less *manipulate* actions than *move* actions.

Our Algorithm

Our algorithm is quite simple. Output data structure is the same as input data structure; that is, the text planner is given a plan as described above as input. We employ a two-step strategy for content selection: filtering and aggregation. In agreement with the text planning team, the resulting plan must not contain more than two items. Only one *manipulate* atom is allowed.

Plan filtering

In the plan filtering stage, only the first *manipulate* instruction and all *move* instructions prior to that *manipulate* instruction are retained. The reasoning behind this design decision is that a user is to be led towards the goal in small steps, with several messages being generated per minute.

Plan aggregation

After the plan is pruned to contain only one *manipulate* instruction, there are still several *move* instructions left which means that two-item limit is violated. To further reduce the number of atoms, the move atoms are aggregated. As per the definition given by Koller [2009a], two regions used as argument for *move*, e.g. move(a, b), with a and b being the regions, are always adjacent. A first attempt at aggregation involved computing the transitive closure over the move instructions in a plan. A plan like "move(a, b); move(b, c); move(c, d); manipulate(a, s1, s2, d)" is compressed to "move(a, d); manipulate(a, s1, s2, d)". However, this approach leads to problems for complicated room shapes or for room transitions. Consider that our system usually employs referring expressions to refer to locations. If the user is told "go to the blue button" with the blue button being in another room, they might be confused.

As a solution, we reason about the world. As we know the user's position and the target location for each *move* instruction as well as the shape of the room, we can easily compute whether there is a direct line of sight between these two positions. This line of sight test is then used to check if the user could potentially view the target location of a move statement. The transitive-closure approach was modified to use this check. If the view is obstructed, then the aggregation is aborted. As a result, the user is guided to the closest location for the target of the next *manipulate* statement they can still see. A second *move* action can be added if the target is still not visible.

This approach assumes that further *move* statements in the plan refer to locations which cannot be seen by the user. It might be possible to construct edge cases where *move* atoms even closer to the next *manipulate* statement are visible where the current algorithm will prefer an earlier atom, thus not achieving maximum compression. However, this goes against the intuition gained when implementing the content selection algorithm. In any case, it is easy to change the algorithm to consider all move atoms while still remaining in O(n) for aggregation.

3.5. Timing: when to talk to the user

While timing is outside the scope of the simple input/output model described above, the issue of whether to communicate with the user or not may as well be considered within the scope of *Content Selection*. In our system, this part is realized in our *NlgSystem* implementation. We run the NLG pipeline if the user has moved from one region to another (with the regions being determined by the discretizer provided by the GIVE API) or if the user has turned. If the user has moved, then it is advisable to re-run the GIVE planner module to obtain a new plan because new or different *move* atoms may have been introduced. If the user has turned, then it is usually not necessary to generate a new plan. However, the NLG pipeline is still invoked as we might want to signal the user in which direction they have to turn. Generation of *turn* instructions is done by the text planner component by doing additional reasoning over the world state.

Another case where the planner is invoked and the NLG pipeline is run is when the user performs an action as this changes the world state. Additionally, the user receives canned text which congratulates them if they pushed the right button.

3.6. Discussion

Let me start by stating that the described content selection method has its shortcomings. Due to time constraints - the Heidelberg team started rather late - we were not able to test and improve as much as we would have liked.

Lack of background research before coming up with our algorithm is one of the issues. While our initial testing shows that it works reasonably well, it'd be nice to have some backing from linguistics or the cognitive sciences in general with regard to content selection rules. For example, we would like to know how many instructions a typical user can process in such a task. As noted above, we choose to only select two pieces of information with a third piece, the *turn* direction, being possibly added later. This seemed like a safe assumption.

Other potential issues arise as well, namely the interaction of the *Content Selection* with the *NlgSystem* with regards to timing. Up to three pieces of information, namely *move*, *manipulate*, and *turn*, need to be processed per message by the user. However, our *NlgSystem* will occasionally produce too many messages. For example, if the user turns around, then the plan will still be valid as the world state is unchanged. In our implementation, a new plan is generated and run through the NLG pipeline. The user is then presented with almost the same message again, overriding the previous message.

Similarly, if the user moves from one region to another, a new message is sent to the user. The regions do not have a fixed size; the size is determined by the discretizer according to room shape and items located in the room [Koller, 2009b]. In some cases, the user will cross many different regions in a short period of time, leading to an abundance of messages. We can conclude that our heuristics for timing are insufficient and need to consider other options as well, perhaps looking at the the distance walked by the user or a rate limit for messages. Another conclusion is that the message rate and message display time needs to be correlated to the amount of information conveyed by the message. Psycholinguistic theories on reading comprehension can provide hints here. Keeping track of previous plans and world state (which we already do, to some extent, to determine whether the user has pushed the right button or whether they have moved) to measure how much new information (the difference between the last message and the new message) will be conveyed. Low-information messages can then be either skipped, displayed for a shorter amount of time or merged with the next message.

To adapt the system to the user, user modelling techniques could be used. The GIVE API already provides the *NlgSystem* with the results of a questionnaire presented to the user. The messages can then be adapted to the user's language proficiency or their computer skills. Regarding content selection, the important tunables will be the amount of filtering and aggregation. Younger users, for example, might prefer more guidance, i.e. more messages with more details. This can be achieved by limiting the amount of *move* steps skipped in the aggregation step.

Another, easier to implement, user modelling approach involves the *handleDidNotUnderstand* method. If the user relies on this functionality often, one possible reason is that the messages are too complicated. In this case, the maximum number of atoms in the output of the content selection module can be further reduced.

To sum up this section: initial testing has shown that our *content selection* approach consisting of filtering and aggregation works well. The problem with our system mostly lies in the message timing and, from a scientific point of view, the lack of a theoretical foundation.

A real evaluation of the complete system is currently done as part of the GIVE project.

Part III.

Discussion

So far, we have seen three different content determination approaches.

Using a Discourse Model to generate Draft Scientific Summaries

Rino and Scott [1994] use a discourse model to generate indicative summaries from an informative text. Their content selection rules are based on discourse relations. The intuition is that if a discourse relation such as *exemplification* holds between two units of discourse, then the example can be dropped without losing relevant information.

Content Determination for Time Series Data

Sripada et al. [2002] use data segmentation techniques adapted from the data mining community to find salient data points in time series data. Their work recognizes the need for user adaptive summaries as well as the phenomenon of cross-channel co-ordination which is realized by threshold values acquired through extensive knowledge acquisition work with domain experts. To sum up, their content determination approach relies on segmentation algorithms as well as on manually obtained configuration data to realize user requirements.

Content Determination for AI plans

The content determination algorithm developed by us for the GIVE-2 challenge uses filtering and aggregation techniques to create messages containing up to three relevant instructions which guide the user on their way to the trophy. The system uses some information about the virtual world, such as unobstructed line of sight between two positions. The user is never told the whole plan, just the parts relevant to his current position and the current state of the world.

4. Comparison

As the systems are quite different in their input data, a comparison is not easy. The most versatile system seems to be Rino and Scott. While their work is originally concerned with text-to-text summarization, the discourse relations on which their content selection rules are based can also be found in other areas, such as theorem provers, knowledge bases and weather forecast generation. However, their approach is also limited by this requirement. Not all data will contain such relations, and not all irrelevant data can be discarded by their heuristics. Compare to, say, the action plan used in GIVE, the notion of indicative versus informative facts does not apply at all. In fact, even if the discourse relations are present, they simply might not be relevant to the user's interest. Consider a case where only a subset of input data is relevant to the user's interests, but for this subset, even data linked with relations such as *exemplification* must be considered.

The algorithms shown for Time Series Data by Sripada et al. are less versatile, but not necessarily less useful. As shown by their SUMTIME-MOUSAM, SUMTIME-TURBINE and SUMTIME-NEONATE, there are lots of real-world applications where TSD is produced and where a textual summary of the data is useful. Particularly impressive is the large amount of knowledge acquisition work to ensure that their system produces results which are satisfying to the end user as well as to create an underlying model of data summarization (Sripada and Reiter [2001]).

Common to both works is the usage of parallel corpora during KA activities. Both Sripada et al. and Rino and Scott compare manually created texts with the source data. This linguistic work helps them identify the salient properties of the target text genre and implement it in their system. Our content determination system, on the other hand, is mostly based on the developer's intuition. Although existing GIVE systems were tried out and example instructions were worked out during seminar meetings, no domain experts were consulted and no corpora were analyzed. This shows in the ad-hoc nature of the implementation.

Interchangeability of the content determination approaches is low as the nature of input data just is too different. We have AI plans in first-order logic, time-series data and discourse models. This observation is also made by Reiter and Dale [1997] as they state: "Both the message-creation process and the form and content of the messages created are highly application-dependent".

4.1. The two-stage model proposed by Sripada and Reiter (2001)

The relation between the two-stage model proposed by Sripada and Reiter [2001] and the content selection algorithm used by Sripada et al. [2002] has already been discussed above with the result that the decision when selecting their algorithm was influenced by their proposed model.

Can the two-stage model retroactively be applied to the two other systems? To recapitulate, the two-stage model divides the content task into two stages: *Domain Problem Solving* (where the overview is built) and *Communicative Reasoning* (where selection rules are applied).

The two-stage model and GIVE

For our GIVE system, I argue that this model is not found in the current implementation. In the filtering stage, all elements following the first manipulate actions are discarded without looking at them. The following aggregation stage then only works with the pruned plan. Therefore, no complete overview of the input data is built. As the overview is an essential part of the two-stage model, the model does not apply here.

Can we benefit from working with a complete overview? Yes. Considering the complete plan together with the world state lets us convey additional information to the user, such as the result of an action. This allows us to establish causal relationships which might motivate the system users better. Looking at the complete plan can also help us with the timing issue and help plan messages better. Consider a scenario where two buttons are located next to each other. Both are to be pushed. In our current implementation, two messages will be generated. Due to the physical closeness of the buttons, these messages may be generated too fast, thus confusing the user. Workarounds such as rate limits can leave the user without instructions for a while. Properly combining both instructions into one message is the better thing to do here.

Once the overview is established, the *Communicative Reasoning* step then employs filtering and merging to select the desired content. Thus, the two-stage model is easily applicable to content selection in the GIVE scenario.

Do we need a complete overview? Not necessarily. Currently, the biggest problem we're facing is the timing issue. While the overview can help here, simpler solutions such as rate limiting, optimization of the message queue, tuning of display time and using different heuristics to work out when it's appropriate to actually communicate with the user might work as well. The two-button scenario described above can also be improved by considering the distance between locations or by matching patterns against the plan (such as "if there are only two *move* actions between two *manipulate* actions, then send both *manipulate* actions in one message"). Of course, following the two-stage model would be a more elegant solution.

The two-stage model and the Discourse Model

Concerning the heuristics proposed by Rino and Scott [1994], I argue that the model can in fact be applied. Their rules are applied to a discourse model which can be regarded as the overview. Rino and Scott do not need to draw inferences in their data, so the overview trivially consists of the input data. Another point of evidence is the fact that meronymic relationships between discourse components are considered as indicated by one of their assumptions: “The deletion of complex discourse segments [..] implies the deletion of the related sub-structures”. Their emphasis on the production of coherent text indicates that they might benefit from using an overview, although their rules do not explicitly need it. It seems the rules inherently are constructed not to violate coherence constraints. As indicated by one of the assumptions, anaphora are a problem when omitting discourse components are referents need to be resolved. In their assumption, they simply state that all referents have been resolved, apparently to simplify their work. The task of anaphora resolution can also be aided by having a global overview of the data.

Thoughts

It would be beneficial to have a more concrete definition of the term “overview” than what Sripada and Reiter [2001] are able to give. It can be debatable whether a system builds a complete overview or solely relies on a portion of the input data as often some kind of filtering is used on the data before. In the case of our GIVE system, one might argue that the plan already is the complete overview and no further action is needed on part of the content selection module. However, I do not support this position as some of the problems described above clearly are a result of not considering the whole plan when selecting content.

Conclusion

For the two additional systems reviewed, one can be fit into the two-stage model while the other one lacks the overview. In the discussion, I have argued that the two-stage model with its global overview can help with content selection. It is possible to determine which content to communicate without building a complete overview, but this approach can bring problems as shown by our GIVE system.

4.2. Conclusion

I have reviewed and described three different systems for content selection. Different input data requires different methods which means that the systems are not interchangeable and thus application-specific. Two of these systems are the result of extensive knowledge acquisition activities which help to replicate the performance and selection strategies found in existing human-produced data. The third system, GIVE, relies on the developers’ intuition and discussions with fellow students.

A two-stage model for content selection has been discussed as well and its usefulness for the reviewed systems has been provisionally examined with favorable results.

Bibliography

Igor Boguslavsky, Nadezhda Frid, Leonid Iomdin, Leonid Kreidlin, Irina Sagalova, and Victor Sizov. Creating a universal networking language module within an advanced nlp system. In *Proceedings of the 18th conference on Computational linguistics*, pages 83–89, Morristown, NJ, USA, 2000. Association for Computational Linguistics. ISBN 1-55860-717-X. doi: http://dx.doi.org/10.3115/990820.990833.

Bruce K. Britton and Arthur C. Graesser. *Models of understanding text.* Lawrence Erlbaum Associates, Inc., Mahwah, New Jersey, 1996. ISBN 0-8058-1848-0.

Armin Fiedler. Macroplanning with a cognitive architecture for the adaptive explanation of proofs. In *Proceedings of the 9th International Workshop on Natural Language Generation*, pages 88–97, 1998.

Jerry R. Hobbs. Why is discourse coherent? *SRI Technical Note*, 176, November 1978.

M. Hoey. Signalling in discourse. In *Discourse Analysis Monographs, English Language Research.* University of Birmingham, 1979.

M. Hoey. *On the Surface of Discourse.* George Allen & Unwin (Publishers) Ltd., 1983.

J. Hutchins. On the structure of scientific discourse. In *UEA Papers in Linguistics 5*, pages 18–39. University of East Anglia, 1977.

John Hutchins. Summarization: Some problems and methods. In *Meaning: The Frontier of Informatics*, pages 151–173. Aslib, 1987.

Eamonn Keogh. A fast and robust method for pattern matching in time series databases. In *Proc. of the 9th Int. Conf. on Tools with Artificial Intelligence (ICTAI-97*, pages 578–584, 1997.

Kevin Knight and Daniel Marcu. Statistics-based summarization - step one: Sentence compression. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence and Twelfth Conference on Innovative Applications of Artificial Intelligence*, pages 703–710. AAAI Press / The MIT Press, 2000. ISBN 0-262-51112-6.

Alexander Koller. Give challenge: Wiki: Givegameworlds, 2009a. URL `http://kenai.com/projects/give-challenge/pages/GIVEGameWorlds`. This is an electronic document. Date last modified: September 18, 2009.

Alexander Koller. Give-2: Participantśmanual: The discretizer, 2009b. URL `http://kenai.com/projects/give-challenge/pages/Manual`. This is an electronic document. Date last modified: November 15, 2009.

Alexander Koller, Donna Byron, Justine Cassell, Robert Dale, Johanna Moore, Jon Oberlander, and Kristina Striegnitz. The software architecture for the first challenge on generating instructions in virtual environments. In *EACL '09: Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics: Demonstrations Session*, pages 33–36, Morristown, NJ, USA, 2009. Association for Computational Linguistics.

Camilla Brandel Martins, Lucia Helena, and Machado Rino. Pruning unl texts for summarizing purposes, 2001.

Ronaldo Martins. Unl, 2010. URL `http://www.unlweb.net/unlweb/index.php?option=com\_content&view=article&id=58:unl&catid=54:unlweb`. This is an electronic document. Date of publication: February 4, 2010. Date retrieved: March 18, 2010.

Thiago Alexandre Salgueiro Pardo and Lucia Helena Machado Rino. Dmsumm: Review and assessment. In *PorTAL '02: Proceedings of the Third International Conference on Advances in Natural Language Processing*, pages 263–274, London, UK, 2002. Springer-Verlag. ISBN 3-540-43829-7.

Ehud Reiter. Has a consensus nl generation architecture appeared, and is it psycholinguistically plausible?, 1994.

Ehud Reiter and Robert Dale. Building applied natural language generation systems, 1997.

Ehud Reiter, Somayajulu Sripada, Jim Hunter, and Ian Davy. Choosing words in computer-generated weather forecasts. *Artificial Intelligence*, 167:137–169, 2005.

Lucia Helena Machado Rino and Donia Scott. Automatic generation of draft summaries: Heuristics for content selection, 1994.

Somayajulu Sripada, Ehud Reiter, Jim Hunter, and Jin Yu. Sumtime: Observations from ka for weather domain, 2001.

Somayajulu Sripada, Ehud Reiter, Jim Hunter, and Jin Yu. Segmenting time series for weather forecasting. In *In Applications and Innovations in Intelligent Systems X*, pages 193–206. Springer-Verlag, 2002.

Somayajulu G. Sripada and Ehud Reiter. A two-stage model for content determination. In *In Proceedings of ENLGW-2001*, pages 3–10, 2001.

Somayajulu G. Sripada, Ehud Reiter, and Ian Davy. Sumtime-mousam: Configurable marine weather forecast generator, 2003a.

Somayajulu G. Sripada, Ehud Reiter, Jim Hunter, and Jin Yu. Summarizing neonatal time series data. In *10th Conference of the European Chapter of the Association for Computational Linguistics*, pages 167–170, 2003b.

Somayajulu G. Sripada, Ehud Reiter, Ian Davy, and Kristian Nilssen. Lessons from deploying nlg technology for marine weather forecast text generation. In *Proceedings of PAIS-2004, 2004*, pages 760–764, 2004.

Robert Weissberg and Suzanne Buker. *Experimental Research Report Writing for Students of English.* Prentice Hall, Inc., 1990. ISBN 0139708316.

Jin Yu, Ehud Reiter, Jim Hunter, and Somayajulu Sripada. Sumtime-turbine: A knowledge-based system to communicate gas turbine time-series data. In *The 16th International Conference on Industrial & Engineering Applications of Artificial Intelligence and Expert Systems*, pages 23–26, 2003.