A Natural Language Generation System for Database Summarization

Abstract: This paper describes a technique for summarizing and expressing the data in a small database as English. It uses a set of relations among a set of normalized values stored in the database in conjunction with a set of schemas and a probabilistic context free grammar. The algorithm is demonstrated on a small, representative database.

I. Introduction

Natural Language Generation is the process of automatically converting data stored in a computer into a natural language like English. Many applications involve large databases consisting of similar items which are either too large or too difficult to comprehend for an average user to grasp easily. One possible solution is to use a natural language generation system to abstract the data in a meaningful way and then present it to the user as readable English rather than as a table. Natural language generation algorithms are usually rule-based, although there has been a recent rise in corpus-based work. This paper presents a rule-based algorithm, tested on a database containing a list of objects and properties of those objects.

A natural language generation system's operation can be decomposed into a series of smaller tasks. While there is disagreement on what precisely these tasks are, Reiter and Dale have composed a set of tasks most frequently agreed upon.

The first task is *Content Determination*. Content determination is the process of deciding which pieces of information contained in the database should be expressed to the user. These pieces of information can be referred to as messages, and often contain information about how the data in the database is related. For example, a database containing information on real estate might produce a message which relates pieces of data as an implication: "because the property is sea side, it is expensive."

The second task is *Discourse Plannning*. Discourse planning is the arranging of messages into a coherent order. It is usually represented as a tree structure. Content determination and discourse planning are often grouped together into a single task called *text planning*.

Next is *Sentence Aggregation*. This is the process of grouping messages together into sentences, and is usually highly dependent on the discourse plan. Grouping sentences into paragraphs and other higher structures is less well understood, but is possible for some tasks.

Lexicalization decides which words are used to express the messages and entities involved in each sentence.

Referring Expression Generation chooses words or phrases to express the entities involved in a discourse. It differs from lexicalization in that it must choose phrases which are sufficient to distinguish the entities being discussed. If the system is describing two real estate properties, referring to each as "the real estate" would not be enough information to distinguish them. Sentence aggregation, lexicalization, and referring expression generation are often grouped together into a single task called *sentence planning*.

Linguistic Realization converts the sentences into grammatically correct structures, such as noun phrases, prepositional phrases, etc.

II. Problem Statement

Convert a database containing a set of objects, associated properties, and relationships between properties into a natural language representation. The output should be easy for a user to understand and accurate.

III. Specification

The database consists of a set of objects, each of which has a proper name. Each object consists of a set of real valued properties, in the range [0.0, 1.0], each of which also has an associated adjective. In some cases, the associated word is a noun, in which case the property may be flagged to indicate that it must be expressed as part of a prepositional phrase. The database must be summarized to eliminate useless information, converted into a series of messages, and then rendered by a natural language generation system into English.

IV. Methods

This section describes the implementation of the natural language system. It follows the basic outline of tasks described in Reiter and Dale's paper.

Text Planning. All information contained in the database is presented to the user, although much of it is abstracted. The abstraction algorithm takes a set of objects and their properties and recognizes similarities which allow it to abstract those objects into classes of similar objects which may then be presented to the user as a group. Each group serves as a paragraph. This process is complex; the next several paragraphs describe it.

The process, as an abstraction, relies on *schemas*. Schemas describe a means of expressing data commonly encountered in the database. Schemas are useful for domain specific natural language generation systems like extracting information from a database. An alternate set of approaches are the *planning based* approaches. Planning based approaches are very general in that they discern the user's plan based on the request for data and present only the data important to that plan, organized by paralleling the steps of the plan. Unfortunately, the user's plan is often impossible to guess accurately, or the user may be querying the database as part of generating a plan. In either case, a planning based approach is not suitable for the task of generic database presentation, even if the query of the database contains a property by which the results are to be ranked.

This paper's schemata consist of a schema for expressing the database as paragraphs. Those paragraphs then have their own schema for translation into sentences. In the lexicalization and linguistic realization phases, a combination of schemas and a probabilistic context free grammar are used. However, the schemas for paragraphs and sentences are actually very complex, so that the term schema may be misleading. As a result, I will not use the term schema.

The similarities are recognized by abstracting each object's properties into a discrete set. The property values are initially real values in [0.0, 1.0]. These are converted into a set of abstract values consisting of {very poor, poor, decent, good, very good}. Very poor values are those less than 0.1. Poor values are between 0.1 and 0.25. Decent values are between 0.25 and 0.75. Good and very good values are the same as poor and very poor values except reflected

across 0.5. These distinctions are arbitrary--they depend on the database whose information is to be presented. However, if no other option were available, a number of abstract values could be guessed and then their bounds could be automatically generated to minimize the difference in the number of objects assigned to each category, thereby providing the maximum information about the objects in the database.

Once the values are abstracted, the objects are examined to form paragraphs. An object's *similarity coefficient* is the absolute difference of each of that object's properties to that of another object, normalized to one such that identical objects receive a similarity coefficient of 1.0, while exactly dissimilar objects receive a similarity coefficient of 0.0. For each object, it is compared to every other object in the database. If they have a similarity coefficient greater than some cutoff value, they are grouped together. The choice of cutoff value is again arbitrary and application dependent, but it should be designed to result in a relatively small number of groups to allow for appropriate summarization while not providing so few groups that too much of the information in the database is abstracted away. Multiple possible groupings are broken by choosing the grouping whose object has the lowest index. This is a fairly arbitrary decision, but it leads to a cleaner algorithm, and the groupings of objects are approximately equally good so long as the cutoff value is reasonable. An alternative would be a search over the space of possible groupings to find the set of groups which maxmimizes the sum of the similarity coefficients between each object, appropriately optimized or simplified for the task at hand.

Now that objects have been grouped into paragraphs, the *relations* which hold between properties are used to form sentences. Every property must have a relationship with some other property, although that relationship may be defined as having no relationship. The relationship types are *correlational*, *implicational*, and *none*. *Correlational relationships* assert that there is a correlation between the values of two properties. If onevalue is high and the other is low, it is a negative correlation. If both values are high or both values are low, it is a positive correlation. Correlations are used to link properties of objects which bear some relationship to one another. If an object's durability and current condition are two properties, these could be related by a correlation, indicating that high durability usually corresponds to being in good condition, while low durability corresponds to being in poor condition and any other combination is anomalous. *Implicational relationships* assert that one set of properties implies a second set of properties. For example, durability and condition could be related by an implicational relationship, allowing the natural language generator to form sentences such as "because object #1 is durable, it is in good condition." A *none relationship* is used for any properties which do not fall into the earlier types of relationships.

These relationships help organize the properties of each object into sentences. Any properties involved in implicational or correlational relationships belong in the same sentence. Furthermore, that sentence should express the relation which holds between those properties. Properties which have no relationship to other properties may be expressed in sentences effectively at random. However, for this application, it is best to express them in a similar order for each object or set of similar objects to provide a parallel structure throughout all of the paragraphs. For the same reason, the relationships which hold between objects are presented in order, with objects in a paragraph which are different from the other objects with respect to that relationship being expressed in the order in which the objects appear in the database.

For each relationship, the values of the properties of each object are examined. Those objects which have similar relationships between a set of properties are grouped into a single sentence. For example, if object 1 and object 2 both have that property 1 is very good and that it

implies properties 2 and 3, which are poor and very poor for both objects, respectively, then object 1 and object 2 may be combined as the subject of the sentence and that sentence expresses properties 1, 2, and 3. Some sentences may express one property of one object for paragraphs containing multiple objects because that one object was different from all of the others with respect to one property. Nevertheless, the results of this process are generally fairly good.

The output of the text planning phase is a set of arrays which record the number of collections of objects and the objects in each collection. This could be represented as a tree, but the array representation is sufficient for generating only paragraphs and sentences.

Next, lexicalization and linguistic realization are performed. Note that this is different from the outline proposed by Reiter and Dale. This is a result of the convenience of implementing referring expression generation as an independent process on the completed sentences, as described later. After text planning, the messages or relationships which each sentence should express have been determined.

As discussed earlier, each sentence is a relationship. Since the relationship can be only correlational, implicational, or none, the sentences formed will show a number of similarities. These similarities are captured in the schemas for each of the relationships. Furthermore, these schemas correspond to parts of a PCFG.

Before discussing each of the three schemas, the schemas which underlie each of them must be explained. This schema represents the simplest possible sentence: a noun phrase followed by a verb phrase. The noun phrase expresses the subjects of the sentence, the verb is a form of ”to be,“ and the remainder of the verb phrase is a set of properties associated with the subjects. The only verbs, then, are ”is“ and ”are.“ However, adding additional verbs is a straight forward programming task, and these two verbs are sufficient for the task of summarizing the database.

Each set of properties is also defined by a schema. This schema simply describes how the properties should be separated by punctuation like commas or conjunctions like and. For example, the schema for one property is simply ”property 1.“ For two properties, ”property 1 and property 2.“ For three properties, it becomes ”property 1, property 2, and property 3,“ and so on. Each property is then rendered into text by a property schema. The same basic set of properties schema applies to the set of subjects of the sentence.

The simplest schema is for each property. This schema adds any appropriate adjectives to a schema and is responsible for recognizing the difference between a normal property and a property which must be expressed as a prepositional phrase. For example, it can express ”very large“ and ”in good condition.“

The implicational schema is the following: the word ”because,“ to indicate the implication; a noun phrase, corresponding to a subjects schema; and a verb phrase, consisting of the verb and a second noun phrase. The second noun phrase is a properties schema which contains the first set of properties. This is followed by a sentence schema which uses the same subjects (although referring expression generation results in the subjects being replaced by ”it,“ or ”they“) and the second set of properties. An example, then, is ”Because The Stately Estate is beautiful and seaside, it is very expensive.“ Another example is ”Because the old woman and the young man are very fashionable, they are very poor.“

The correlational schema is a sentence schema in which the noun phrase for the properties becomes two noun phrases joined by a conjunction. The conjuntion is either ”and“ or ”but“ depending on whether the correlation is positive or negative. So sentences like ”the

medallion is shiny but ugly“ are possible.

The schema for no relationship is the simplest. It is a sentence schema. An example is ”the porter is burly, surly, and curlish.“

Each of these schema corresponds to part of a PCFG which can be used to vary the wording used. For example, property schema do not always include an adjective, or the adjective may be different each time. Furthermore, each schema builds the appropriate part of the tree, such that the result of lexicalization and linguistic realization is a tree structure which obeys the grammar of the PCFG.

Each property contains a flag to indicate if it needs to be expressed as a prepositional phrase. While this could be detected using a corpus, it is generally not necessary given the small size of the list of properties for most applications, and it is better to avoid any possible inaccuracies by requiring the system designer to choose.

Finally, referring expression generation is performed on the trees representing each sentence. Each subject is assumed to be a proper name. Since the task of referring expression generation is to choose a subject (e.g. ”they,“ ”it,“ ”all three“) which is unambiguous but natural, some hypothesis regarding how ambiguity arises in referring expressions is advisable. Clearly, a preceeding sentence with a singular subject identical to the sentence under examination results in an unambiguous option: ”it.“ However, when plural subjects are involved, it becomes more difficult. A conservative formulation of when a referring expression is ambiguous is that it is ambiguous if the union over the subjects of all preceeding sentences in the paragraph is not equal to the set of subjects of the current paragraph. That is, assuming that each paragraph deals with a different set of objects, as is the case for this algorithm, ambiguity only involves the subjects of the present paragraph. For brevity, I will denote the union of all preceeding subjects as ‘A’ and the set of subjects in the current sentence as ‘B.’

If a sentence contains more subjects than are in A, or subjects not in A, then using ”they“ as the subject will lose the information about the subjects contained in the intersection of A and B. The following paragraph illustrates this difficulty with the last sentence:

Jake is a swell guy. Jill is a swell girl. Jake, Jill, and Tommy are all swell.

If the subject of the third sentence is replaced with ”they,“ the reader can no longer determine that Tommy is a swell boy.

If a sentence contains fewer subjects than are in A, then using ”they“ may mislead the reader into believing that all subjects in the paragraph are referred to by ”they,“ when only some subset were intended.

Jake is a swell guy. Jill is a swell girl. Tommy is a nasty guy. Jake and Jill are swell.

The fourth sentence, were the subject to be replaced with ”they,“ could imply that Tommy was a swell guy as well as Jake and Jill.

If the sentence contains exactly those subjects in A, then ”they“ may be used with only one ambiguity. That is, if ”they“ is meant to refer only to a subset of the subjects discussed in the paragraph, it may be interpreted to include all subjects in A. In most cases which this algorithm encounters, this ambiguity is allowable. If it were not, it could be easily fixed by adding the word ”all“ after the verb of the verb phrase. Then, for example,

Jake is a swell guy. Jill is a swell girl. Tommy is also a swell guy. They are swell.

would be disambiguated as

Jake is a swell guy. Jill is a swell girl. Tommy is also a swell guy. They are all swell.

However, in all cases for this algorithm, ”they“ is used to refer to all preceeding subjects in the (usually short) paragraph, such that a reader will most likely not be disturbed by any ambiguities which arise. The only difficulty with adding ”all“ to a sentence is that ”all“ usually implies a large number of subjects, so that it can sound awkward if ”they“ refers to only two subjects. This is an area in which statistical methods might be applicable to determine the average human being’s disambiguation capabilities with respect to referring expressions and to then use knowledge of those capabilities to allow ambiguities which humans can resolve while eliminating ambiguities which humans cannot.

Subjects occuring within sentences which are not the primary subjects are handled by an internal referring expression generator. In these cases, the use of ”they“ is rarely ambiguous because any second set of subjects appearing in a sentence is most likely associated with the first set of subjects of the sentence.

The following paragraphs are a series of further implementation notes which would disrupt the normal discoure.

Care must be exercised in the choice of adjectives which can be used to express the abstracted values of properties. If a value is low, such that an adjective ”poor“ is used to describe it in a prepositional phrase and the pseudo-adjective ”not very“ is used to describe it in the main sentence, the result is describing the object as ”not very poor,“ which is a form of double negation. This difficulty only arises when a property is expressed as a preposition, and the algorithm avoids it by having an additional rule which prevents the use of such adjectival constructs.

The grammar used to generate sentences is formally a PCFG, but the probabilistic elements of the grammar are restricted to subtrees rooted at a property node since ”schemas“ are used to generate the paragraphs, sentences, and most of the sentence structure. For example, the adjectives used to describe a high attribute might be ”good“ and ”excellent.“ Beyond these simple applications, PCFGs have little use for domain-specific natural language generation. An element of randomness could also be added to the words emitted by other nodes in the sentences to increase the variety of the sentences produced, however. For example, the conjunction in the grammar, represented by a node, could probabilistically emit ”and“ or ”in addition to,“ perhaps biased toward using ”and“ for short lists and ”in addition to“ for longer lists.

These random words or phrases could be extracted from a corpus of tagged text. However, for domain specific applications it is easier and more accurate to provide a pseudo-dictionary of words and phrases for each purpose. The primary difficulty is in providing a sufficient number of rules to prevent these probabilistic elements from conflicting. While providing complex rules is possible for this algorithm, in a more general algorithm it would be better to extract these rules from a tagged text corpus. More precisely, one could examine the corpus for all instances of two specified tags with a specific relationship between them and then pick one of the pairs of words resulting from that analysis according to the corpus’ probability

distribution when the natural language generator encounters that arrangement of tags in building the sentence trees. Augmenting this approach with word classes would allow more general rules to be derived, at the cost of introducing potentially incorrect word arrangements not exhibited by the corpus.

There is also a small ”peep-hole“ optimizations which helps improve the fluidity of the algorithm. In an implicational relation, if the first property is greater than the second property, the conjunction ”but“ is replaced with ”but only.“ These peep-hole optimizations are useful, but they are tedious to implement and only slightly improve the output.

V. Results

Presenting results for a nondeterministic natural language generation system is difficult. Since the nondeterminism arises from sentence aggregation and structure as well as diction, it would be very difficult to produce a database and a comprehensive set of all acceptable outputs by hand. As a result, I will present a series of database values and the algorithm’s output for comparison.

These examples all use an implicational relationship that the price of an object implies its weight and defense rating. While this is an unrealistic implication, it suffices for testing. They also assume a correlational relationship between the condition of the weapon and its durability. Finally, the remaining properties have no relationship to one another.

The first example demonstrates a pair of swords being described in the same paragraph due to their similarity, followed by two dissimilar descriptions of swords which each appear in their own paragraph. I will omit the database contents, as they are provided in the text.

Jake's Longsword is moderately fast and extremely deadly. Bo's Longsword is extremely fast and exceptionally deadly. They are in extremely good condition but only a little durable. Because they are extremely expensive, they are decently heavy and extremely useful for defense.

Eljay's Rapier is extremely fast and decently deadly. It is in poor condition and a little durable. Because it is exceptionally expensive, it is moderately heavy and extremely useful for defense.

The Goblin's Cudgel is not very fast but extremely deadly. It is in below average condition and extremely durable. Because it is decently expensive, it is extremely heavy and not very useful for defense.

Here is a second example. The first paragraph demonstrates a flaw in the similarity coefficient approach to identifying similar objects. While The Goblin’s ‘lil Cudgel and The Centaur’s Axe are similar objects, those similarities are not exploited because they are used in relationships. Unfortunately, it is difficult to design an algorithm which could efficiently determine the optimal aggregation of objects into paragraphs for arbitrary sets of relationships and properties without actually attempting the aggregation for some subset of all possible combinations and computing the average number of sentences per paragraph to determine the best combination.

The Goblin's 'lil Cudgel is moderately fast and moderately deadly. The Centaur's Axe is not very fast but exceptionally deadly. The Goblin's 'lil Cudgel is in poor condition and exceptionally durable. The Centaur's Axe is in extremely good condition and unusually durable.

Because The Goblin's 'lil Cudgel is moderately expensive, it is moderately heavy and a little useful for defense. Because The Centaur's Axe is extremely expensive, it is exceptionally heavy and a little useful for defense.

The Simple Man's Sword is moderately fast and moderately deadly. The Short Man's Sword is not very fast and not very deadly. They are in below average condition and moderately durable. Because they are decently expensive, they are moderately heavy and moderately useful for defense.

These two examples are also beginning to reveal another problem. This algorithm produces readable text, but when a user sees text produced by this algorithm many times, the lack of different sentence structures and larger dictionaries of words will reveal the output's underlying similarity. This problem can be fixed fairly easily through the addition of new schemas and expanding the dictionaries, but it also suggests that, to save labor, large databases should not be summarized using this algorithm without first augmenting it with the corpus-based enhancements mentioned earlier.

Finally, careful analysis of a much larger output would suggest the addition of new schemas which would express relationships between objects' properties as spectrums. That is, a paragraph might describe ten similar objects which happen to be very similar in that some large subset of their properties are monotonically increasing under some ordered sequence of the objects. This would allow a new schema in which an introductory sentence presents the ordering of the objects and the set of properties which follow that order, while the remainder of the paragraph notes the properties of each object which did not obey the pattern. Sufficiently large patterns of this sort could be used to form sections which are further subdivided into paragraphs. Unfortunately, the generation of sections and paragraphs is very poorly understood in natural language generation, and it is precisely the recognition and expression of that structure which is most important to summarizing large databases in a helpful way.

VI. Conclusion

The algorithm described in this paper respectably expresses a small database as readable English. However, extending it for use with large databases requires a large amount of highly efficient pattern recognition code or statistical code to break the database down into patterns, and further advances in the field of natural language generation to render those patterns into English. The algorithm would also need several techniques to increase the variety of its output so that a user who was going to see a lot of this algorithm's output would not become bored with the sentence structure and diction. Despite these flaws, the algorithm generates grammatically correct English while using only rule-based approaches, which makes it suitable for many of the tasks to which natural language generation algorithms are applied today.