Instance-based Sentence Boundary Determination by Optimization for Natural Language Generation

Shimei Pan and **James C. Shaw**
IBM T. J. Watson Research Center
19 Skyline Drive
Hawthorne, NY 10532, USA
{shimei,shawjc}@us.ibm.com

Abstract

This paper describes a novel instance-based sentence boundary determination method for natural language generation that optimizes a set of criteria based on examples in a corpus. Compared to existing sentence boundary determination approaches, our work offers three significant contributions. First, our approach provides a general domain independent framework that effectively addresses sentence boundary determination by balancing a comprehensive set of sentence complexity and quality related constraints. Second, our approach can simulate the characteristics and the style of naturally occurring sentences in an application domain since our solutions are optimized based on their similarities to examples in a corpus. Third, our approach can adapt easily to suit a natural language generation system's capability by balancing the strengths and weaknesses of its sub-components (e.g. its aggregation and referring expression generation capability). Our final evaluation shows that the proposed method results in significantly better sentence generation outcomes than a widely adopted approach.

1 Introduction

The problem of sentence boundary determination in natural language generation exists when more than one sentence is needed to convey multiple concepts and propositions. In the classic natural language generation (NLG) architecture (Reiter, 1994), sentence boundary decisions are made during the sentence planning stage in which the syntactic structure and wording of sentences are decided. Sentence boundary determination is a complex process that directly impacts a sentence's readability (Gunning, 1952), its semantic cohesion, its syntactic and lexical realizability, and its smoothness between sentence transitions. Sentences that are too complex are hard to understand, so are sentences lacking semantic cohesion and cross-sentence coherence. Further more, bad sentence boundary decisions may even make sentences unrealizable.

To design a sentence boundary determination method that addresses these issues, we employ an instance-based approach (Varges and Mellish, 2001; Pan and Shaw, 2004). Because we optimize our solutions based on examples in a corpus, the output sentences can demonstrate properties, such as similar sentence length distribution and semantic grouping similar to those in the corpus. Our approach also avoids problematic sentence boundaries by optimizing the solutions using all the instances in the corpus. By taking a sentence's lexical and syntactic realizability into consideration, it can also avoid sentence realization failures caused by bad sentence boundary decisions. Moreover, since our solution can be adapted easily to suit the capability of a natural language generator, we can easily tune the algorithm to maximize the generation quality. To the best of our knowledge, there is no existing comprehensive solution that is domain-independent and possesses all the above qualities. In summary, our work offers three significant contributions:

1. It provides a general and flexible sentence

Proceedings of the 43rd Annual Meeting of the ACL, pages 565–572,
Ann Arbor, June 2005.

boundary determination framework which takes a comprehensive set of sentence complexity and quality related criteria into consideration and ensures that the proposed algorithm is sensitive to not only the complexity of the generated sentences, but also their semantic cohesion, multi-sentence coherence and syntactic and lexical realizability.

2. Since we employ an instance-based method, the proposed solution is sensitive to the style of the sentences in the application domain in which the corpus is collected.
3. Our approach can be adjusted easily to suit a sentence generation system's capability and avoid some of its known weaknesses.

Currently, our work is embodied in a multimodal conversation application in the real-estate domain in which potential home buyers interact with the system using multiple modalities, such as speech and gesture, to request residential real-estate information (Zhou and Pan, 2001; Zhou and Chen, 2003; Zhou and Aggarwal, 2004). After interpreting the request, the system formulates a multimedia presentation, including automatically generated speech and graphics, as the response (Zhou and Aggarwal, 2004). The proposed sentence boundary determination module takes a set of propositions selected by a content planner and passes the sentence boundary decisions to SEGUE (Pan and Shaw, 2004), an instance-based sentence generator, to formulate the final sentences. For example, our system is called upon to generate responses to a user's request: "Tell me more about this house." Even though not all of the main attributes of a house (more than 20) will be conveyed, it is clear that a good sentence boundary determination module can greatly ease the generation process and improve the quality of the output.

In the rest of the paper, we start with a discussion of related work, and then describe our instance-base approach to sentence boundary determination. Finally, we present our evaluation results.

2 Related Work

Existing approaches to sentence boundary determination typically employ one of the following strategies. The first strategy uses domain-specific heuristics to decide which propositions can be combined. For example, *Proteus* (Davey, 1979; Ritchie, 1984) produces game descriptions by employing domain-specific sentence scope heuristics. This approach can work well for a particular application, however, it is not readily reusable for new applications.

The second strategy is to employ syntactic, lexical, and sentence complexity constraints to control the aggregation of multiple propositions (Robin, 1994; Shaw, 1998). These strategies can generate fluent complex sentences, but they do not take other criteria into consideration, such as semantic cohesion. Further more, since these approaches do not employ global optimization as we do, the content of each sentence might not be distributed evenly. This may cause dangling sentence problem (Wilkinson, 1995).

Another strategy described in Mann and Moore(1981) guided the aggregation process by using an evaluation score that is sensitive to the structure and term usage of a sentence. Similar to our approach, they rely on search to find an optimal solution. The main difference between this approach and ours is that their evaluation score is computed based on preference heuristics. For example, all the semantic groups existing in a domain have to be coded specifically in order to handle semantic grouping. In contrast, in our framework, the score is computed based on a sentence's similarity to corpus instances, which takes advantage of the naturally occurring semantic grouping in the corpus.

Recently, Walker (2002) and Stent (2004) used statistical features derived from corpus to rank generated sentence plans. Because the plan ranker was trained with existing examples, it can choose a plan that is consistent with the examples. However, depending on the features used and the size of the training examples, it is unclear how well it can capture patterns like semantic grouping and avoid problems likes dangling sentences.

3 Examples

Before we describe our approach in detail, we start with a few examples from the real-estate domain to demonstrate the properties of the proposed approach.

First, sentence complexity impacts sentence boundary determination. As shown in Table 1, after receiving a user's request (U1) for the details of a house, the content planner asked the sentence planner to describe the house with a set of attributes including its asking price, style, number of bedrooms, number of bathrooms, square footage, garage, lot size, property tax, and its associated town and school

Example	Turn	Sentence
E1	U1	Tell me more about this house
	S1	This is a 1 million dollar 3 bedroom, 2 bathroom, 2000 square foot colonial with 2 acre of land, 2 car garage, annual taxes 8000 dollars in Armonk and in the Byram Hills school district.
	S2	This is a 1 million dollar house. This is a 3 bedroom house. This is a 2 bathroom house. This house has 2000 square feet. This house has 2 acres of land. This house has 2 car garage. This is a colonial house. The annual taxes are 8000 dollars. This house is in Armonk. This house is in the Byram Hills school district.
	S3	This is a 3 bedroom, 2 bathroom, 2000 square foot colonial located in Armonk with 2 acres of land. The asking price is 1 million dollar and the annual taxes are 8000 dollars. The house is located in the Byram Hills School District.
E2	S4	This is a 1 million dollar 3 bedroom house. This is a 2 bathroom house with annual taxes of 8000 dollars.
	S5	This is a 3 bedroom and 2 bathroom house. Its price is 1 million dollar and its annual taxes are 8000 dollars.
E3	S6	The tax rate of the house is 3 percent.
	S7	The house has an asphalt roof.
E4	S8	This is a 3 bedroom, 2 bathroom colonial with 2000 square feet and 2 acres of land.
	S9	The house has 2 bedrooms and 3 bathrooms. This house is a colonial. It has 2000 square feet. The house is on 2 acres of land.

Table 1: Examples

district name. Without proper sentence boundary determination, a sentence planner may formulate a single sentence to convey all the information, as in S1. Even though S1 is grammatically correct, it is too complex and too exhausting to read. Similarly, output like S2, despite its grammatical correctness, is choppy and too tedious to read. In contrast, our instance-based sentence boundary determination module will use examples in a corpus to partition those attributes into several sentences in a more balanced manner (S3).

Semantic cohesion also influences the quality of output sentences. For example, in the real-estate domain, the number of bedrooms and number of bathrooms are two closely related concepts. Based on our corpus, when both concepts appear, they almost always conveyed together in the same sentence. Given this, if the content planner wants to convey a house with the following attributes: price, number of bedrooms, number of bathrooms, and property tax, S4 is a less desirable solution than S5 because it splits these concepts into two separate sentences. Since we use instance-based sentence boundary determination, our method generates S5 to minimize the difference from the corpus instances.

Sentence boundary placement is also sensitive to the syntactic and lexical realizability of grouped items. For example, if the sentence planner asks the surface realizer to convey two propositions S6 and S7 together in a sentence, a realization failure will be triggered because both S6 and S7 only exist in the corpus as independent sentences. Since neither of them can be transformed into a modifier based on the corpus, S6 and S7 cannot be aggregated in our system. Our method takes a sentence's lexical and syntactic realizability into consideration in order to avoid making such aggregation request to the surface realizer in the first place.

A generation system's own capability may also influence sentence boundary determination. Good sentence boundary decisions will balance a system's strengths and weaknesses. In contrast, bad decisions will expose a system's venerability. For example, if a sentence generator is good at performing aggregations and weak on referring expressions, we may avoid incoherence between sentences by preferring aggregating more attributes in one sentence (like in S8) rather than by splitting them into multiple sentences (like in S9).

In the following, we will demonstrate how our approach can achieve all the above goals in a unified instance-based framework.

4 Instance-based boundary determination

Instance-based generation automatically creates sentences that are similar to those generated by humans, including their way of grouping semantic content, their wording and their style. Previously, Pan and Shaw (2004) have demonstrated that instance-based learning can be applied successfully in generating new sentences by piecing together existing words and segments in a corpus. Here, we want to demonstrate that by applying the same principle, we can make better sentence boundary decisions.

The key idea behind the new approach is to find a sentence boundary solution that minimizes the expected difference between the sentences resulting from these boundary decisions and the examples in the corpus. Here we measure the expected difference based a set of cost functions.

4.1 Optimization Criteria

We use three sentence complexity and quality related cost functions as the optimization criteria: sentence boundary cost, insertion cost and deletion cost.

Sentence boundary cost (SBC): Assuming P is a set of propositions to be conveyed and S is a collection of example sentences selected from the corpus to convey P. Then we say P can be realized by S with a sentence boundary cost that is equal to $(|S|-1)*SBC$ in which $|S|$ is the number of sentences and SBC is the sentence boundary cost. To use a specific example from the real-estate domain, the input P has three propositions:

p_1. House1 has-attr (style=colonial).

p_2. House1 has-attr(bedroom=3).

p_3. House1 has-attr(bathroom=2).

One solution, S, contains 2 sentences:

s_1. This is a 3 bedroom, 2 bathroom house.

s_2. This is a colonial house.

Since only one sentence boundary is involved, S is a solution containing one boundary cost. In the above example, even though both s_1 and s_2 are grammatical sentences, the transition from s_1 to s_2 is not quite smooth. They sound choppy and disjointed. To penalize this, whenever there is a sentence break, there is a SBC. In general, the SBC is a parameter that is sensitive to a generation system's capability such as its competence in reference expression generation. If a generation system does not have a robust approach for tracking the focus across sentences, it is likely to be weak in referring expression generation and adding sentence boundaries are likely to cause fluency problems. In contrast, if a generation system is very capable in maintaining the coherence between sentences, the proper sentence boundary cost would be lower.

Insertion cost: Assume P is the set of propositions to be conveyed, and C_i is an instance in the corpus that can be used to realize P by inserting a missing proposition p_j to C_i, then we say P can be realized using C_i with an insertion cost of $icost(C_H, p_j)$, in which C_H is the host sentence in the corpus containing proposition p_j. Using an example from our real-estate domain, assume the input P=(p_2, p_3, p_4), where

p_4. House1 has-attr (square footage=2000).

Assume C_i is a sentence selected from the corpus to realize P: "This is 3 bedroom 2 bathroom house". Since C_i does not contain p_4, p_4 needs to be added. We say that P can be realized using C_i by inserting a proposition p_4 with an insertion cost of $icost(C_H, p_4)$, in which C_H is a sentence in the corpus such as "This is a house with 2000 square feet."

The insertion cost is influenced by two main factors: the syntactic and lexical insertability of the proposition p_j and a system's capability in aggregating propositions. For example, if in the corpus, the proposition p_j is always realized as an independent sentence and never as a modifier, $icost(*, p_j)$ should be extremely high, which effectively prohibit p_j from becoming a part of another sentence. $icost(*, p_j)$ is defined as the minimum insertion cost among all the $icost(C_H, p_j)$. Currently $icost(C_H, p_j)$ is computed dynamically based on properties of corpus instances. In addition, since whether a proposition is insertable depends on how capable an aggregation module can combine propositions correctly into a sentence, the insertion cost should be assigned high or low accordingly.

Deletion cost: Assume P is a set of input propositions to be conveyed and C_i is an instance in the corpus that can be used to convey P by deleting an unneeded proposition p_j in C_i. Then, we say P can be realized using C_i with a deletion cost $dcost(C_i, p_j)$. As a specific example, assuming the input is P=(p_2, p_3, p_4), C_i is an instance in the corpus "This is a 3 bedroom, 2 bathroom, 2000 square foot colonial house." In addition to the propositions p_2, p_3 and p_4, C_i also conveys a proposition p_1. Since p_1 is not needed when conveying P, we say that P can be realized using C_i by deleting proposition p_1 with a deletion cost of $dcost(C_i, p_1)$.

The deletion cost is affected by two main factors as well: first the syntactic relation between p_j and its host sentence. Given a new instance C_i, "This 2000 square foot 3 bedroom, 2 bathroom house is a colonial", deleting p_1, the main object

of the verb, will make the rest of the sentence incomplete. As a result, $dcost(C_i, p_1)$ is very expensive. In contrast, $dcost(C_i, p_4)$ is low because the resulting sentence is still grammatically sound. Currently $dcost(C_i, p_j)$ is computed dynamically based on properties of corpus instances. Second, the expected performance of a generation system in deletion also impacts the deletion cost. Depending on the sophistication of the generator to handle various deletion situations, the expected deletion cost can be high if the method employed is naive and error prone, or is low if the system can handle most cases accurately.

Overall cost: Assume P is the set of propositions to be conveyed and S is the set of instances in the corpus that are chosen to realize P by applying a set of insertion, deletion and sentence breaking operations, the overall cost of the solution

$$\begin{aligned} Cost(P) &= \sum_{C_i}(W_i * \sum_j icost(C_{Hj}, p_j) \\ &\quad + W_d * \sum_k dcost(C_i, p_k)) \\ &\quad + (N_b - 1) * SBC \end{aligned}$$

in which W_i, W_d and SBC are the insertion weight, deletion weight and sentence boundary cost; N_b is the number of sentences in the solution, C_i is a corpus instance been selected to construct the solution and C_{Hj} is the host sentence that proposition p_j belongs.

4.2 Algorithm: Optimization based on overall cost

We model the sentence boundary determination process as a branch and bound tree search problem. Before we explain the algorithm itself, first a few notations. The input P is a set of input propositions chosen by the content planner to be realized. Σ is the set of all possible propositions in an application domain. Each instance C_i in the corpus C is represented as a subset of Σ. Assume S is a solution to P, then it can be represented as the overall cost plus a list of pairs like (C_is, O_is), in which C_is is one of the instances selected to be used in that solution, O_is is a set of deletion, insertion operations that can be applied to C_is to transform it to a subsolution S_i. To explain this representation further, we use a specific example in which P=(a, d, e, f), Σ=(a, b, c, d, e, f g, h, i). One of the boundary solution S can be represented as

$$\begin{aligned} S &= (Cost(S), (S1, S2)) \\ S_1 &= (C_1 = (a, b, d, i), delete(b, i)), \\ S_2 &= (C_2 = (e), insert(f \text{ as in } C_3 = (f, g))) \\ Cost(S) &= W_d * (dcost(C_1, b) + dcost(C_1, i)) + \\ &\quad W_i * icost(C_3, f) + 1 * SBC \end{aligned}$$

in which C_1 and C_2 are two corpus instances selected as the bases to formulate the solution and C_3 is the host sentence containing proposition f.

The general idea behind the instance-based branch and bound tree search algorithm is that given an input, P, for each corpus instance C_i, we construct a search branch, representing all possible ways to realize the input using the instance plus deletions, insertions and sentence breaks. Since each sentence break triggers a recursive call to our sentence boundary determination algorithm, the complexity of the algorithm is NP-hard. To speed up the process, for each iteration, we prune unproductive branches using an upper bound derived by several greedy algorithms. The details of our sentence boundary determination algorithm, $sbd(P)$, are described below. P is the set of input propositions.

1. Set the current upper bound, UB, to the minimum cost of solutions derived by greedy algorithms, which we will describe later. This value is used to prune unneeded branches to make the search more efficient.

2. For each instance C_i in corpus C in which $(C_i \cap P) \neq \emptyset$, loop from step 3 to 9. The goal here is to identify all the useful corpus instances for realizing P.

3. Delete all the propositions $p_j \in D$ in which $D = C_i - P$ (D contains propositions in C_i but not exist in P) with cost $Cost_d(P) = W_d * \sum_{P_j \in D} dcost(C_i, p_j)$. This step computes the deletion operators and their associated costs.

4. Let $I = P - C_i$ (I contains propositions in P but not in C_i). For each subset $E_j \subseteq I$ (E_j includes $\emptyset$ and I itself), iterate through step 5 to 9. These steps figure out all the possible ways to add the missing propositions, including inserting into the instance C_i and separating the rest as independent sentence(s).

5. Generate a solution in which $\forall p_k \in E_j$, insert p_k to C_i. All the propositions in $Q = I - E_j$ will be realized in different sentences, thus incurring a SBC.

6. We update the cost $Cost(P)$ to

$$Cost_d(P) + W_i * \sum_{p_k \in E_j} icost(*, p_k) + SBC + Cost(Q)$$

in which $Cost(Q)$ is the cost of sbd(Q) which recursively computes the best solution for input Q and $Q \subset P$. To facilitate dynamic programming, we remember the best solution for Q derived by sbd(Q) in case Q is used to formulate other solutions.

7. If the lower bound for Cost(P) is greater than the established upper bound UB, prune this branch.

8. Using the notation described in the beginning of Sec. 4.2, we update the current solution to

$$sbd(P) = (Cost(P), (C_i, delete_{\forall p_j \in D}(p_j), insert_{\forall p_k \in E_j}(p_k))) \bigoplus sbd(Q)$$

in which $\bigoplus$ is an operator that composes two partial solutions.

9. If sbd(P) is a complete solution (either Q is empty or have a known best solution) and $Cost(P) < UB$, update the upper bound $UB = Cost(P)$.

10. Output the solution with the lowest overall cost.

To establish the initial UB for pruning, we use the minimum of the following three bounds. In general, the tighter the UB is, the more effective the pruning is.

Greedy set partition: we employ a greedy set partition algorithm in which we first match the set $S \subset P$ with the largest $|S|$. Repeat the same process for P' where $P' = P - S$. The solution cost is $Cost(P) = (N - 1) * SBC$, and N is the number of sentences in the solution. The complexity of this computation is $O(|P|)$, where $|P|$ is the number of propositions in P.

Revised minimum set covering: we employ a greedy minimum set covering algorithm in which we first find the set S in the corpus that maximizes the overlapping of propositions in the input P. The unwanted propositions in $S - P$ are deleted. Assume $P' = P - S$, repeat the same process to P' until P' is empty. The only difference between this and the previous approach is that S here might not be a subset of P. The complexity of this computation is $O(|P|)$.

One maximum overlapping sentence: we first identify the instance C_i in corpus that covers the maximum number of propositions in P. To arrive at a solution for P, the rest of the propositions not covered by C_i are inserted into C_i and all the unwanted propositions in C_i are deleted. The cost of this solution is

$$W_d * \sum_{p_j \in D} dcost(C_i, p_j) + W_i * \sum_{p_k \in I} icost(*, p_k)$$

in which D includes proposition in C_i but not in P, and I includes propositions in P but not in C_i.

Currently, we update UB only after a complete solution is found. It is possible to derive better UB by establishing the upper bound for each partial solution, but the computational overhead might not justify doing so.

4.3 Approximation Algorithm

Even with pruning and dynamic programming, the exact solution still is very expensive computationally. Computing exact solution for an input size of 12 propositions has over 1.6 millions states and takes more than 30 minutes (see Figure 1). To make the search more efficient for tasks with a large number of propositions in the input, we naturally seek a greedy strategy in which at every iteration the algorithm myopically chooses the next best step without regard for its implications on future moves. One greedy search policy we implemented explores the branch that uses the instance with maximum overlapping propositions with the input and ignores all branches exploring other corpus instances. The intuition behind this policy is that the more overlap an instance has with the input, the less insertions or sentence breaks are needed.

Figure 1 and Figure 2 demonstrate the tradeoff between computation efficiency and accuracy. In this graph, we use instances from the real-estate corpus with size 250, we vary the input sentence length from one to twenty and the results shown in the graphs are average value over several typical weight configurations ((W_d,W_i,SBC)=

(1,3,5),(1,3,7),(1,5,3),(1,7,3),(1,1,1)). Figure 2 compares the quality of the solutions when using exact solutions versus approximation. In our interactive multimedia system, we currently use exact solution for input size of 7 propositions or less and switch to greedy for any larger input size to ensure sub-second performance for the NLG component.

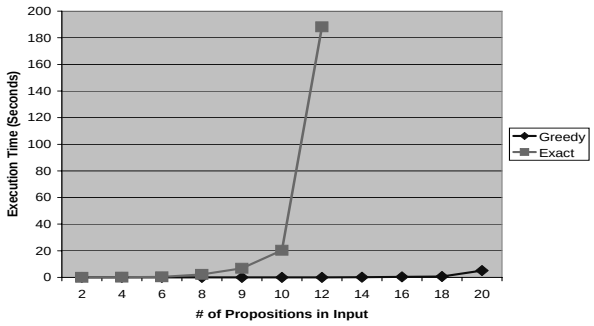

Figure 1: Speed difference between exact solutions and approximations

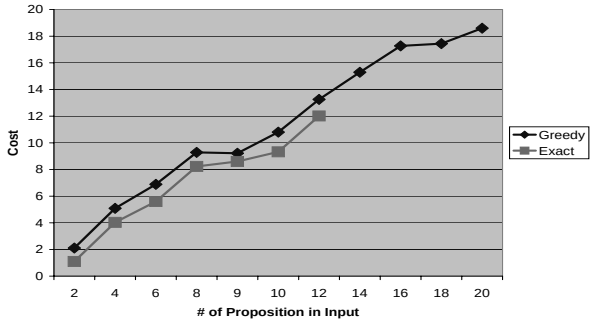

Figure 2: Cost difference between exact solutions and approximations

Measures	Ours	B-3	B-6
Dangling sentence (7)	0	100%	100%
Split Semantic Group	1%	61%	21%
Realization Failure	0	56%	72%
Fluency	59%	4%	8%

Table 2: Comparisons

5 Evaluations

To evaluate the quality of our sentence boundary decisions, we implemented a baseline system in which boundary determination of the aggregation module is based on a threshold of the maximum number of propositions allowed in a sentence (a simplified version of the second strategy in Section 2. We have tested two threshold values, the average (3) and maximum (6) number of propositions among corpus instances. Other sentence complexity measures, such as the number of words and depth of embedding are not easily applicable for our comparison because they require the propositions to be realized first before the boundary decisions can be made.

We tune the relative weight of our approach to best fit our system's capability. Currently, the weights are empirically established to $W_d = 1$, $W_i = 3$ and $SBC = 3$. Based on the output generated from both systems, we derive four evaluation metrics:

1. **Dangling sentences**: We define dangling sentences as the short sentences with only one proposition that follow long sentences. This measure is used to verify our claim that because we use global instead of local optimization, we can avoid generating dangling sentences by making more balanced sentence boundary decisions. In contrast, the baseline approaches have dangling sentence problem when the input proposition is 1 over the multiple of the threshold values. The first row of Table 2 shows that when the input proposition length is set to 7, a pathological case, among the 200 input proposition sets randomly generated, the baseline approach always produce dangling sentences (100%). In contrast, our approach always generates more balanced sentences (0%).

2. **Semantic group splitting**. Since we use an instance-based approach, we can maintain the semantic cohesion better. To test this, we randomly generated 200 inputs with up to 10 propositions containing semantic grouping of both the number of bedrooms and number of bathrooms. The second row, Split Semantic Group, in Table 2 shows that our algorithm can maintain semantic group much better than the baseline approach. Only in 1% of the output sentences, our algorithm generated number of bedrooms and number of bathrooms in separate sentences. In contrast, the baseline approaches did much worse (61% and 21%).

3. **Sentence realization failure**. This measure is used to verify that since we also take a sentence's lexical and syntactical realizability into consideration, our sentence boundary decisions will result in less sentence realization failures.

An realization failure occurs when the aggregation module failed to realize one sentence for all the propositions grouped by the sentence boundary determination module. The third row in Table 2, Realization Failure, indicates that given 200 randomly generated input proposition sets with length from 1 to 10, how many realization happened in the output. Our approach did not have any realization failure while for the baseline approaches, there are 56% and 72% outputs have one or more realization failures.

4. **Fluency**. This measure is used to verify our claim that since we also optimize our solutions based on boundary cost, we can reduce incoherence across multiple sentences. Given 200 randomly generated input propositions with length from 1 to 10, we did a blind test and presented pairs of generated sentences to two human subjects randomly and asked them to rate which output is more coherent. The last row, Fluency, in Table 2 shows how often the human subjects believe that a particular algorithm generated better sentences. The output of our algorithm is preferred for more than 59% of the cases, while the baseline approaches are preferred 4% and 8%, respectively. The other percentages not accounted for are cases where the human subject felt there is no significant difference in fluency between the two given choices. The result from this evaluation clearly demonstrates the superiority of our approach in generating coherent sentences.

6 Conclusion

In the paper, we proposed a novel domain independent instance-based sentence boundary determination algorithm that is capable of balancing a comprehensive set of generation capability, sentence complexity, and quality related constraints. This is the first domain-independent algorithm that possesses many desirable properties, including balancing a system's generation capabilities, maintaining semantic cohesion and cross sentence coherence, and preventing severe syntactic and lexical realization failures. Our evaluation results also demonstrate the superiority of the approach over a representative domain independent sentence boundary solution.

References

Anthony C. Davey. 1979. *Discourse Production*. Edinburgh University Press, Edinburgh.

Robert Gunning. 1952. *The Technique of Clear Writing*. McGraw-Hill.

William C. Mann and James A. Moore. 1981. Computer generation of multiparagraph English text. *American Journal of Computational Linguistics*, 7(1):17–29.

Shimei Pan and James Shaw. 2004. SEGUE: A hybrid case-based surface natural language generator. In *Proc. of ICNLG*, Brockenhurst, U.K.

Ehud Reiter. 1994. Has a consensus NL generation architecture appeared, and is it psycholinguistically plausible? In *Proc. of INLG*, Kennebunkport, Maine.

Graeme D. Ritchie. 1984. A rational reconstruction of the Proteus sentence planner. In *Proc. of the COLING and the ACL*, Stanford, CA.

Jacques Robin. 1994. Automatic generation and revision of natural language summaries providing historical background. In *Proc. of the Brazilian Symposium on Artificial Intelligence*, Fortaleza, CE, Brazil.

James Shaw. 1998. Segregatory coordination and ellipsis in text generation. In *Proc. of the COLING and the ACL.*, Montreal, Canada.

Amanda Stent, Rashmi Prasad, and Marilyn Walker. 2004. Trainable sentence planning for complex information presentation in spoken dialog systems. In *Proc. of the ACL*, Barcelona, Spain.

Sebastian Varges and Chris Mellish. 2001. Instance-based natural language generation. In *Proc. of the NAACL*, Pittsburgh, PA.

Marilyn Walker, Owen Rambow, and Monica Rogati. 2002. Training a sentence planner for spoken dialogue using boosting. *Computer Speech and Language*.

John Wilkinson. 1995. Aggregation in natural language generation: Another look. Co-op work term report, Dept. of Computer Science, University of Waterloo.

Michelle Zhou and Vikram Aggarwal. 2004. An optimization-based approach to dynamic data content selection in intelligent multimedia interfaces. In *Proc. of the UIST*, Santa Fe, NM.

Michelle X. Zhou and Min Chen. 2003. Automated generation of graphic sketches by example. In *IJCAI*, Acapulco, Mexico.

Michelle X. Zhou and Shimei Pan. 2001. Automated authoring of coherent multimedia discourse in conversation systems. In *ACM Multimedia*, Ottawa, Canada.