Aggregation via Set Partitioning for Natural Language Generation

Regina Barzilay
Computer Science and Artificial Intelligence Laboratory
Massachusetts Institute of Technology
regina@csail.mit.edu

Mirella Lapata
School of Informatics
University of Edinburgh
mlap@inf.ed.ac.uk

Abstract

The role of aggregation in natural language generation is to combine two or more linguistic structures into a single sentence. The task is crucial for generating concise and readable texts. We present an efficient algorithm for automatically learning aggregation rules from a text and its related database. The algorithm treats aggregation as a set partitioning problem and uses a global inference procedure to find an optimal solution. Our experiments show that this approach yields substantial improvements over a clustering-based model which relies exclusively on local information.

1 Introduction

Aggregation is an essential component of many natural language generation systems (Reiter and Dale, 2000). The task captures a mechanism for merging together two or more linguistic structures into a single sentence. Aggregated texts tend to be more concise, coherent, and more readable overall (Dalianis, 1999; Cheng and Mellish, 2000). Compare, for example, sentence (2) in Table 1 and its non-aggregated counterpart in sentences (1a)–(1d). The difference between the fluent aggregated sentence and its abrupt and redundant alternative is striking.

The benefits of aggregation go beyond making texts less stilted and repetitive. Researchers in psycholinguistics have shown that by eliminating redundancy, aggregation facilitates text comprehension and recall (see Yeung (1999) and the references therein). Furthermore, Di Eugenio et al. (2005) demonstrate that aggregation can improve learning in the context of an intelligent tutoring application.

(1)	a.	Holocomb had an incompletion in the first quarter.
	b.	Holocomb had another incompletion in the first quarter.
	c.	Davis was among four San Francisco defenders.
	d.	Holocomb threw to Davis for a leaping catch.
(2)	**After two incompletions in the first quarter, Holcomb found Davis among four San Francisco defenders for a leaping catch.**	

Table 1: Aggregation example (in boldface) from a corpus of football summaries

In existing generation systems, aggregation typically comprises two processes: semantic grouping and sentence structuring (Wilkinson, 1995). The first process involves partitioning semantic content (usually the output of a content selection component) into disjoint sets, each corresponding to a single sentence. The second process is concerned with syntactic or lexical decisions that affect the realization of an aggregated sentence.

To date, this task has involved human analysis of a domain-relevant corpus and manual development of aggregation rules (Dalianis, 1999; Shaw, 1998). The corpus analysis and knowledge engineering work in such an approach is substantial, prohibitively so in

Proceedings of the Human Language Technology Conference of the North American Chapter of the ACL, pages 359–366,
New York, June 2006.

large domains. But since corpus data is already used in building aggregation components, an appealing alternative is to try and learn the rules of semantic grouping directly from the data. Clearly, this would greatly reduce the human effort involved and ease porting generation systems to new domains.

In this paper, we present an automatic method for performing the semantic grouping task. We address the following problem: given an aligned parallel corpus of sentences and their underlying semantic representations, how can we learn grouping constraints automatically? In our case the semantic content corresponds to entries from a database; however, our algorithm could be also applied to other representations such as propositions or sentence plans.

We formalize semantic grouping as a set partitioning problem, where each partition corresponds to a sentence. The strength of our approach lies in its ability to capture global partitioning constraints by performing collective inference over local pairwise assignments. This design allows us to integrate important constraints developed in symbolic approaches into an automatic aggregation framework. At a *local* level, pairwise constraints capture the semantic compatibility between pairs of database entries. For example, if two entries share multiple attributes, then they are likely to be aggregated. Local constraints are learned using a binary classifier that considers all pairwise combinations attested in our corpus. At a *global* level, we search for a semantic grouping that maximally agrees with the pairwise preferences while simultaneously satisfying constraints on the partitioning as a whole. Global constraints, for instance, could prevent the creation of overly long sentences, and, in general, control the compression rate achieved during aggregation. We encode the global inference task as an integer linear program (ILP) that can be solved using standard optimization tools.

We evaluate our approach in a sports domain represented by large real-world databases containing a wealth of interrelated facts. Our aggregation algorithm model achieves an 11% F-score increase on grouping entry pairs over a greedy clustering-based model which does not utilize global information for the partitioning task. Furthermore, these results demonstrate that aggregation is amenable to an automatic treatment that does not require human involvement.

In the following section, we provide an overview of existing work on aggregation. Then, we define the learning task and introduce our approach to content grouping. Next, we present our experimental framework and data. We conclude the paper by presenting and discussing our results.

2 Related Work

Due to its importance in producing coherent and fluent text, aggregation has been extensively studied in the text generation community.[1] Typically, semantic grouping and sentence structuring are interleaved in one step, thus enabling the aggregation component to operate over a rich feature space. The common assumption is that other parts of the generation system are already in place during aggregation, and thus the aggregation component has access to discourse, syntactic, and lexical constraints.

The interplay of different constraints is usually captured by a set of hand-crafted rules that guide the aggregation process (Scott and de Souza, 1990; Hovy, 1990; Dalianis, 1999; Shaw, 1998). Alternatively, these rules can be learned from a corpus. For instance, Walker et al. (2001) propose an overgenerate-and-rank approach to aggregation within the context of a spoken dialog application. Their system relies on a preference function for selecting an appropriate aggregation among multiple alternatives and assumes access to a large feature space expressing syntactic and pragmatic features of the input representations. The preference function is learned from a corpus of candidate aggregations marked with human ratings. Another approach is put forward by Cheng and Mellish (2000) who use a genetic algorithm in combination with a hand-crafted preference function to opportunistically find a text that satisfies aggregation and planning constraints.

Our approach differs from previous work in two important respects. First, our ultimate goal is a generation system which can be entirely induced from a parallel corpus of sentences and their corresponding database entries. This means that our generator will operate over more impoverished representations than are traditionally assumed. For example we do

[1] The approaches are too numerous to list; we refer the interested reader to Reiter and Dale (2000) and Reape and Mellish (1999) for comprehensive overviews.

Passing					
PLAYER	CP/AT	**YDS**	**AVG**	**TD**	INT
Cundiff	22/37	237	6.4	1	1
Carter	23/47	237	5.0	1	4
...	...	...	...	...	...

Rushing					
PLAYER	REC	**YDS**	**AVG**	LG	**TD**
Hambrick	13	33	2.5	10	1
...	...	...	...	...	...

1	(*Passing* (Cundiff 22/37 237 6.4 1 1))
	(*Passing* (Carter 23/47 237 5.0 1 4))
2	(*Interception* (Lindell 1 52 1))
	(*Kicking* (Lindell 3/3 100 38 1/1 10))
3	(*Passing* (Bledsoe 17/34 104 3.1 0 0))
4	(*Passing* (Carter 15/32 116 3.6 1 0))
5	(*Rushing* (Hambrick 13 33 2.5 10 1))
6	(*Fumbles* (Bledsoe 2 2 0 0 0))

Table 2: Excerpt of database and (simplified) example of aggregated entries taken from a football domain. This fragment will give rise to 6 sentences in the final text.

not presume to know all possible ways in which our database entries can be lexicalized, nor do we presume to know which semantic or discourse relations exist between different entries. In this framework, aggregation is the task of grouping semantic content without making any decisions about sentence structure or its surface realization. Second, we strive for an approach to the aggregation problem which is as domain- and representation-independent as possible.

3 Problem Formulation

We formulate aggregation as a *supervised partitioning task*, where the goal is to find a clustering of input items that maximizes a global utility function. The input to the model consists of a set E of database entries selected by a content planner. The output of the model is a partition $S = \{S_i\}$ of nonempty subsets such that each element of E appears in exactly one subset.[2] In the context of aggregation, each partition represents entries that should be verbalized in the same sentence. An example of a partitioning is illustrated in the right side of Table 2 where eight entries are partitioned into six clusters.

We assume access to a relational database where each entry has a type and a set of attributes associated with it. Table 2 (left) shows an excerpt of the database we used for our experiments. The aggregated text in Table 2 (right) contains entries of five types: `Passing`, `Interception`, `Kicking`, `Rushing`, and `Fumbles`. Entries of type `Passing` have six attributes — `PLAYER`, `CP/AT`, `YDS`, `AVG`, `TD`, `INT`, entries of type `Interception` have four attributes, and so on. We assume the existence of a non-empty set of attributes that we can use for meaningful comparison between entities of different types. In the example above, types `Passing` and `Rushing` share the attributes `PLAYER`, `AVG` (short for average), `TD` (short for touchdown) and `YDS` (short for yards). These are indicated in boldface in Table 2. In Section 4.1, we discuss how a set of shared attributes can be determined for a given database.

Our training data consists of entry sets with a known partitioning. During testing, our task is to infer a partitioning for an unseen set of entries.

4 Modeling

Our model is inspired by research on text aggregation in the natural language generation community (Cheng and Mellish, 2000; Shaw, 1998). A common theme across different approaches is the notion of similarity — content elements described in the same sentence should be related to each other in some meaningful way to achieve conciseness and coherence. Consider for instance the first cluster in Table 2. Here, we have two entries of the same type (i.e., `Passing`). Furthermore, the entries share the same values for the attributes `YDS` and `TD` (i.e., 237 and 1). On the other hand, clusters 5 and 6 have no attributes in common. This observation motivates modeling aggregation as a binary classification task: given a pair of entries, predict their aggregation status based on the similarity of their attributes. Assuming a perfect classifier, pairwise assignments

[2] By definition, a partitioning of a set defines an equivalence relation which is reflexive, symmetric, and transitive.

will be consistent with each other and will therefore yield a valid partitioning.

In reality, however, this approach may produce globally inconsistent decisions since it treats each pair of entries in isolation. Moreover, a pairwise classification model cannot express general constraints regarding the partitioning as a whole. For example, we may want to constrain the size of the generated partitions and the compression rate of the document, or the complexity of the generated sentences.

To address these requirements, our approach relies on global inference. Given the pairwise predictions of a *local* classifier, our model finds a *globally optimal* assignment that satisfies partitioning-level constraints. The computational challenge lies in the complexity of such a model: we need to find an optimal partition in an exponentially large search space. Our approach is based on an Integer Linear Programming (ILP) formulation which can be effectively solved using standard optimization tools. ILP models have been successfully applied in several natural language processing tasks, including relation extraction (Roth and Yih, 2004), semantic role labeling (Punyakanok et al., 2004) and the generation of route directions (Marciniak and Strube, 2005).

In the following section, we introduce our local pairwise model and afterward we present our global model for partitioning.

4.1 Learning Pairwise Similarity

Our goal is to determine whether two database entries should be aggregated given the similarity of their shared attributes. We generate the training data by considering all pairs $\langle e_i, e_j \rangle \in E \times E$, where E is the set of all entries attested in a given document. An entry pair forms a positive instance if its members belong to the same partition in the training data. For example, we will generate $\frac{8\times7}{2}$ unordered entry pairs for the eight entries from the document in Table 2. From these, only two pairs constitute positive instances, i.e., clusters 1 and 2. All other pairs form negative instances.

The computation of pairwise similarity is based on the attribute set $\mathcal{A} = \{A_i\}$ shared between the two entries in the pair. As discussed in Section 3, the same attributes can characterize multiple entry types, and thus form a valid basis for entry comparison. The shared attribute set $\mathcal{A}$ could be identified in many ways. For example, using domain knowledge or by selecting attributes that appear across multiple types. In our experiments, we follow the second approach: we order attributes by the number of entry types in which they appear, and select the top five[3].

A pair of entries is represented by a binary feature vector $\{x_i\}$ in which coordinate x_i indicates whether two entries have the same value for attribute i. The feature vector is further expanded by conjuctive features that explicitly represent overlap in values of multiple attributes up to size k. The parameter k controls the cardinality of the maximal conjunctive set and is optimized on the development set.

To illustrate our feature generation process, consider the pair (*Passing* (Quincy Carter 15/32 116 3.6 1 0)) and (*Rushing* (Troy Hambrick 13 33 2.5 10 1)) from Table 2. Assuming $\mathcal{A} = \{\texttt{Player}, \texttt{Yds}, \texttt{TD}\}$ and $k = 2$, the similarity between the two entries will be expressed by six features, three representing overlap in individual attributes and three representing overlap when considering pairs of attributes. The resulting feature vector has the form $\langle 0, 0, 1, 0, 0, 0 \rangle$.

Once we define a mapping from database entries to features, we employ a machine learning algorithm to induce a classifier on the feature vectors generated from the training documents. In our experiments, we used a publicly available maximum entropy classifier[4] for this task.

4.2 Partitioning with ILP

Given the pairwise predictions of the local classifier, we wish to find a valid global partitioning for the entries in a single document. We thus model the interaction between *all* pairwise aggregation decisions as an optimization problem.

Let $c_{\langle e_i, e_j \rangle}$ be the probability of seeing entry pair $\langle e_i, e_j \rangle$ aggregated (as computed by the pairwise classifier). Our goal is to find an assignment that maximizes the sum of pairwise scores and forms a valid partitioning. We represent an assignment using a set of indicator variables $x_{\langle e_i, e_j \rangle}$ that are set

[3] Selecting a larger number of attributes for representing similarity would result in considerably sparser feature vectors.

[4] The software can be downloaded from `http://www.isi.edu/~hdaume/megam/index.html`.

to 1 if $\langle e_i, e_j \rangle$ is aggregated, and 0 otherwise. The score of a global assignment is the sum of its pairwise scores:

$$\sum_{\langle e_i,e_j\rangle \in E \times E} c_{\langle e_i,e_j\rangle} x_{\langle e_i,e_j\rangle} + (1 - c_{\langle e_i,e_j\rangle})(1 - x_{\langle e_i,e_j\rangle}) \quad (1)$$

Our inference task is solved by maximizing the overall score of pairs in a given document:

$$\text{argmax} \sum_{\langle e_i,e_j\rangle \in E \times E} c_{\langle e_i,e_j\rangle} x_{\langle e_i,e_j\rangle} + (1 - c_{\langle e_i,e_j\rangle})(1 - x_{\langle e_i,e_j\rangle}) \quad (2)$$

subject to:

$$x_{\langle e_i,e_j\rangle} \in \{0, 1\} \; \forall \, e_i, e_j \in E \times E \quad (3)$$

We augment this basic formulation with two types of constraints. The first type of constraint ensures that pairwise assignments lead to a consistent partitioning, while the second type expresses global constraints on partitioning.

Transitivity Constraints We place constraints that enforce transitivity in the label assignment: if $x_{\langle e_i,e_j\rangle} = 1$ and $x_{\langle e_j,e_k\rangle} = 1$, then $x_{\langle e_i,e_k\rangle} = 1$. A pairwise assignment that satisfies this constraint defines an equivalence relation, and thus yields a unique partitioning of input entries (Cormen et al., 1992).

We implement transitivity constraints by introducing for every triple e_i, e_j, e_k $(i \neq j \neq k)$ an inequality of the following form:

$$x_{\langle e_i,e_k\rangle} \geq x_{\langle e_i,e_j\rangle} + x_{\langle e_j,e_k\rangle} - 1 \quad (4)$$

If both $x_{\langle e_i,e_j\rangle}$ and $x_{\langle e_j,e_k\rangle}$ are set to one, then $x_{\langle e_i,e_k\rangle}$ also has to be one. Otherwise, $x_{\langle e_i,e_k\rangle}$ can be either 1 or 0.

Global Constraints We also want to consider global document properties that influence aggregation. For example, documents with many database entries are likely to exhibit different compression rates during aggregation when compared to documents that contain only a few.

Our first global constraint controls the number of aggregated sentences in the document. This is achieved by limiting the number of entry pairs with positive labels for each document:

$$\sum_{\langle e_i,e_j\rangle \in E \times E} x_{\langle e_i,e_j\rangle} \leq m \quad (5)$$

Notice that the number m is not known in advance. However, we can estimate this parameter from our development data by considering documents of similar size (as measured by the number of corresponding entry pairs.) For example, texts with thousand entry pairs contain on average 70 positive labels, while documents with 200 pairs have around 20 positive labels. Therefore, we set m separately for every document by taking the average number of positive labels observed in the development data for the document size in question.

The second set of constraints controls the length of the generated sentences. We expect that there is an upper limit on the number of pairs that can be clustered together. This restriction can be expressed in the following form:

$$\forall \, e_i \sum_{e_j \in E} x_{\langle e_i,e_j\rangle} \leq k \quad (6)$$

This constraint ensures that there can be at most k positively labeled pairs for any entry e_i. In our corpus, for instance, at most nine entries can be aggregated in a sentence. Again k is estimated from the development data by taking into account the average number of positively labeled pairs for every entry type (see Table 2). We therefore indirectly capture the fact that some entry types (e.g., `Passing`) are more likely to be aggregated than others (e.g., `Kicking`).

Solving the ILP In general, solving an integer linear program is NP-hard (Cormen et al., 1992). Fortunately, there exist several strategies for solving ILPs. In our study, we employed *lp_solve*, an efficient Mixed Integer Programming solver[5] which implements the Branch-and-Bound algorithm. We generate and solve an ILP for every document we wish to aggregate. Documents of average size (approximately 350 entry pairs) take under 30 minutes on a 450 MHz Pentium III machine.

[5]The software is available from `http://www.geocities.com/lpsolve/`

5 Evaluation Set-up

The model presented in the previous section was evaluated in the context of generating summary reports for American football games. In this section we describe the corpus used in our experiments, our procedure for estimating the parameters of our models, and the baseline method used for comparison with our approach.

Data For training and testing our algorithm, we employed a corpus of football game summaries collected by Barzilay and Lapata (2005). The corpus contains 468 game summaries from the official site of the American National Football League[6] (NFL). Each summary has an associated database containing statistics about individual players and events. In total, the corpus contains 73,400 database entries, 7.1% of which are verbalized; each entry is characterized by a type and a set of attributes (see Table 2). Database entries are automatically aligned with their corresponding sentences in the game summaries by a procedure that considers anchor overlap between entity attributes and sentence tokens. Although the alignment procedure is relatively accurate, there is unavoidably some noise in the data.

The distribution of database entries per sentence is shown in Figure 1. As can be seen, most aggregated sentences correspond to two or three database entries. Each game summary contained 14.3 entries and 9.1 sentences on average. The training and test data were generated as described in Section 4.1. We used 96,434 instances (300 summaries) for training, 59,082 instances (68 summaries) for testing, and 53,776 instances (100 summaries) for development purposes.

Parameter Estimation As explained in Section 4, we infer a partitioning over a set of database entries in a two-stage process. We first determine how likely all entry pairs are to be aggregated using a local classifier, and then infer a valid global partitioning for all entries. The set of shared attributes $\mathcal{A}$ consists of five features that capture overlap in players, time (measured by game quarters), action type, outcome type, and number of yards. The maximum cardinality of the set of conjunctive features is five.

[6] See `http://www.nfl.com/scores`.

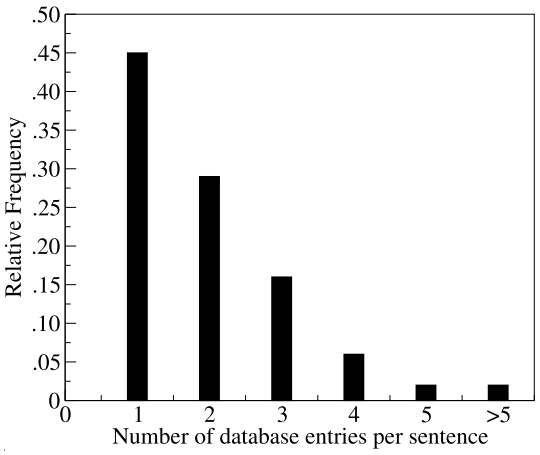

Figure 1: Distribution of aggregated sentences in the NFL corpus

Overall, our local classifier used 28 features, including 23 conjunctive ones. The maximum entropy classifier was trained for 100 iterations. The global constraints for our ILP models are parametrized (see equations (5) and (6)) by m and k which are estimated separately for every test document. The values of m ranged from 2 to 130 and for k from 2 to 9.

Baseline Clustering is a natural baseline model for our partitioning problem. In our experiments, we a employ a single-link agglomerative clustering algorithm that uses the scores returned by the maximum entropy classifier as a pairwise distance measure. Initially, the algorithm creates a separate cluster for each sentence. During each iteration, the two closest clusters are merged. Again, we do not know in advance the appropriate number of clusters for a given document. This number is estimated from the training data by averaging the number of sentences in documents of the same size.

Evaluation Measures We evaluate the performance of the ILP and clustering models by measuring F-score over pairwise label assignments. We compute F-score individually for each document and report the average. In addition, we compute partition accuracy in order to determine how many sentence-level aggregations our model predicts correctly.

Clustering	Precision	Recall	F-score
Mean	57.7	66.9	58.4
Min	0.0	0.0	0.0
Max	100.0	100.0	100.0
StDev	28.2	23.9	23.1

ILP Model	Precision	Recall	F-score
Mean	82.2	65.4	70.3
Min	37.5	28.6	40.0
Max	100.0	100.0	100.0
StDev	19.2	20.3	16.6

Table 3: Results on pairwise label assignment (precision, recall, and F-score are averaged over documents); comparison between clustering and ILP models

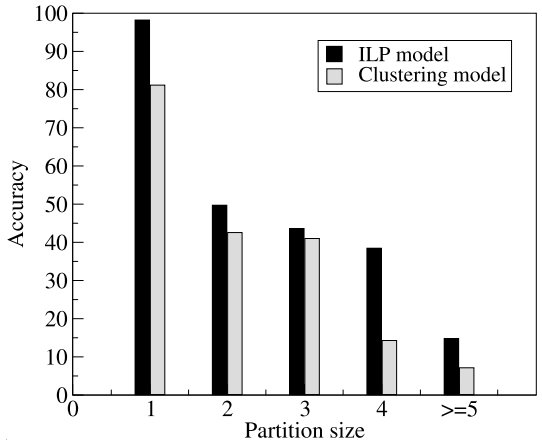

Figure 2: Partition accuracy for sentences of different size

6 Results

Our results are summarized in Table 3. As can be seen, the ILP model outperforms the clustering model by a wide margin (11.9% F-score). The two methods yield comparable recall; however, the clustering model lags considerably behind as far as precision is concerned (the difference is 24.5 %).[7]

Precision is more important than recall in the context of our aggregation application. Incorrect aggregations may have detrimental effects on the coherence of the generated text. Choosing not to aggregate may result in somewhat repetitive texts; however, the semantic content of the underlying text remains intact. In the case of wrong aggregations, we may group together facts that are not compatible, and even introduce implications that are false.

We also consider how well our model performs when evaluated on total partition accuracy. Here, we are examining the partitioning as a whole and ask the following question: how many clusters of size 1, 2 ... n did the algorithm get right? This evaluation measure is stricter than F-score which is computed over pairwise label assignments. The partition accuracy for entry groups of varying size is shown in Figure 2. As can be seen, in all cases the ILP outperforms the clustering baseline. Both models are fairly accurate at identifying singletons, i.e., database entries which are not aggregated. Performance is naturally worse when considering larger clusters. Interestingly, the difference between the two models becomes more pronounced for partition sizes 4 and 5 (see Figure 2). The ILP's accuracy increases by 24% for size 4 and 8% for size 5.

These results empirically validate the importance of global inference for the partitioning task. Our formulation allows us to incorporate important document-level constraints as well as consistency constraints which cannot be easily represented in a vanilla clustering model.

7 Conclusions and Future Work

In this paper we have presented a novel data-driven method for aggregation in the context of natural language generation. A key aspect of our approach is the use of global inference for finding aggregations that are maximally consistent and coherent. We have formulated our inference problem as an integer linear program and shown experimentally that it outperforms a baseline clustering model by a wide margin. Beyond generation, the approach holds promise for other NLP tasks requiring the accurate partitioning of items into equivalence classes (e.g., coreference resolution).

[7] Unfortunately we cannot apply standard statistical tests such as the t-test on F-scores since they violate assumptions about underlying normal distributions. It is not possible to use an assumptions-free test like χ^2 either, since F-score is not a frequency-based measure. We can, however, use χ^2 on precision and recall, since these measures are estimated from frequency data. We thus find that the ILP model is significantly better than the clustering model on precision (χ^2 = 16.39, $p < 0.01$); the two models are not significantly different in terms of recall (χ^2 = 0.02, $p < 0.89$).

Currently, semantic grouping is carried out in our model sequentially. First, a local classifier learns the similarity of entity pairs and then ILP is employed to infer a valid partitioning. Although such a model has advantages in the face of sparse data (recall that we used a relatively small training corpus of 300 documents) and delivers good performance, it effectively decouples learning from inference. An appealing future direction lies in integrating learning and inference in a unified global framework. Such a framework would allow us to incorporate global constraints directly into the learning process.

Another important issue, not addressed in this work, is the interaction of our aggregation method with content selection and surface realization. Using an ILP formulation may be an advantage here since we could use feedback (in the form of constraints) from other components and knowlegde sources (e.g., discourse relations) to improve aggregation or indeed the generation pipeline as a whole (Marciniak and Strube, 2005).

Acknowledgments

The authors acknowledge the support of the National Science Foundation (Barzilay; CAREER grant IIS-0448168 and grant IIS-0415865) and EPSRC (Lapata; grant GR/T04540/01). Thanks to Eli Barzilay, Michael Collins, David Karger, Frank Keller, Yoong Keok Lee, Igor Malioutov, Johanna Moore, Kevin Simler, Ben Snyder, Bonnie Webber and the anonymous reviewers for helpful comments and suggestions. Any opinions, findings, and conclusions or recommendations expressed above are those of the authors and do not necessarily reflect the views of the NSF or EPSRC.

References

R. Barzilay, M. Lapata. 2005. Collective content selection for concept-to-text generation. In *Proceedings of the Human Language Technology Conference and the Conference on Empirical Methods in Natural Language Processing*, 331–338, Vancouver.

H. Cheng, C. Mellish. 2000. Capturing the interaction between aggregation and text planning in two generation systems. In *Proceedings of the 1st International Natural Language Generation Conference*, 186–193, Mitzpe Ramon, Israel.

T. H. Cormen, C. E. Leiserson, R. L. Rivest. 1992. *Intoduction to Algorithms*. The MIT Press.

H. Dalianis. 1999. Aggregation in natural language generation. *Computational Intelligence*, 15(4):384–414.

B. Di Eugenio, D. Fossati, D. Yu. 2005. Aggregation improves learning: Experiments in natural language generation for intelligent tutoring systems. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics*, 50–57, Ann Arbor, MI.

E. H. Hovy. 1990. Unresolved issues in paragraph planning. In R. Dale, C. Mellish, M. Zock, eds., *Current Research in Natural Language Generation*, 17–41. Academic Press, New York.

T. Marciniak, M. Strube. 2005. Beyond the pipeline: Discrete optimization in NLP. In *Proceedings of the Annual Conference on Computational Natural Language Learning*, 136–143, Ann Arbor, MI.

V. Punyakanok, D. Roth, W. Yih, D. Zimak. 2004. Semantic role labeling via integer linear programming inference. In *Proceedings of the International Conference on Computational Linguistics*, 1346–1352, Geneva, Switzerland.

M. Reape, C. Mellish. 1999. Just what is aggregation anyway? In *Proceedings of the 7th European Workshop on Natural Language Generation*, 20–29, Toulouse, France.

E. Reiter, R. Dale. 2000. *Building Natural Language Generation Systems*. Cambridge University Press, Cambridge.

D. Roth, W. Yih. 2004. A linear programming formulation for global inference in natural language tasks. In *Proceedings of the Annual Conference on Computational Natural Language Learning*, 1–8, Boston, MA.

D. Scott, C. S. de Souza. 1990. Getting the message across in RST-based text generation. In R. Dale, C. Mellish, M. Zock, eds., *Current Research in Natural Language Generation*, 47–73. Academic Press, New York.

J. Shaw. 1998. Clause aggregation using linguistic knowledge. In *Proceedings of 9th International Workshop on Natural Language Generation*, 138–147, Niagara-on-the-Lake, Ontario, Canada.

M. A. Walker, O. Rambow, M. Rogati. 2001. Spot: A trainable sentence planner. In *Proceedings of the 2nd Annual Meeting of the North American Chapter of the Association for Computational Linguistics*, 17–24, Pittsburgh, PA.

J. Wilkinson. 1995. Aggregation in natural language generation: Another look. Technical report, Computer Science Department, University of Waterloo, 1995.

A. S. Yeung. 1999. Cognitive load and learner expertise: Split-attention and redundancy effects in reading comprehension tasks with vocabulary definitions. *Journal of Experimental Education*, 67(3):197–218.