Combining Hierarchical Clustering and Machine Learning to Predict High-Level Discourse Structure

Caroline Sporleder and **Alex Lascarides**
School of Informatics
University of Edinburgh
2 Buccleuch Place,
Edinburgh EH8 9LW
{csporled, alex}@inf.ed.ac.uk

Abstract

We propose a novel method to predict the inter-paragraph discourse structure of text, i.e. to infer which paragraphs are related to each other and form larger segments on a higher level. Our method combines a clustering algorithm with a model of segment "relatedness" acquired in a machine learning step. The model integrates information from a variety of sources, such as word co-occurrence, lexical chains, cue phrases, punctuation, and tense. Our method outperforms an approach that relies on word co-occurrence alone.

1 Introduction

For the interpretation of texts, it is not enough to understand each sentence individually; one also needs to have an idea of how sentences relate to each other, i.e. one needs to know the *discourse structure* of the text. This knowledge is important for many NLP applications, e.g. text summarisation or question answering. Most discourse theories, such as *Rhetorical Structure Theory* (RST) (Mann and Thompson, 1987), assume that discourse structure can be represented as a tree whose leaves are the elementary discourse units (*edus*) of the text, e.g. sentences or clauses. Edus are linked to each other by *rhetorical relations*, such as *Contrast* or *Elaboration*, and then form larger text segments (represented by intermediate nodes in the tree), which in turn can be linked to other segments via rhetorical relations, giving rise to even larger segments.

Discourse parsing is concerned with inferring discourse structure automatically and can be viewed as consisting of three co-dependent subtasks: (i) identifying the edus, (ii) determining which discourse segments (edus or larger segments) relate to each other, i.e. finding the correct attachment site for each segment, and (iii) identifying how discourse segments are related to each other, i.e. inferring the rhetorical relation.

While these tasks have been dealt with quite well for small structures (i.e. on clause and sentence level) (Soricut and Marcu, 2003), many of these approaches cannot be applied directly to higher-level structures (e.g. on multi-sentence and inter-paragraph level) because they rely nearly exclusively on cue phrases, which are much less useful for large structures (Marcu, 2000, p. 129). In this paper, we focus exclusively on inferring high-level structure. In particular, we investigate ways to automatically determine the correct attachment site for paragraph and multi-paragraph segments.

Finding a good attachment site is a complex task; even if one requires the final structure to be a tree, the number of valid structures grows rapidly with the number of edus in a text. An exhaustive search is often not feasible, even for relatively small texts. One way to address this problem is by making an assumption that discourse structure correlates with higher-level text structure, i.e. that it obeys sentence, paragraph and section breaks (Marcu, 2000). Under this assumption a non-sentential edu cannot attach directly to another edu that is not part of the same sentence and a sentence cannot attach directly to a sentence in a preceding or following paragraph. This leads to a significant reduction in the number of valid trees by allowing for a "divide-and-conquer" approach which treats inter-paragraph structure as independent from intra-paragraph structure.

While this clearly is a simplifying assumption, it is likely that textual and discourse structure are related in some way. For example, psycholinguistic research has shown that paragraph boundaries are not arbitrary (humans can predict them with an accuracy that is higher than chance) and they are not largely determined by aesthetics either (humans do not apply a simple length criterion when deciding where to place a boundary) (Stark, 1988). This suggests that the placement of paragraph boundaries may be influenced by discourse structure. This is further supported by the observation that 79% of the paragraphs in the manually annotated data set we used (see Section 3) do correspond to discourse segments, even though the annotators were free to link text spans in any way they liked, provided it

did not result in crossing branches, i.e. the annotation instructions did not bias the annotators to ensure that paragraphs corresponded to discourse segments (Carlson and Marcu, 2001).

To determine the high-level tree-structure of a text in the absence of cue phrases, word co-occurrence measures have been suggested (Marcu, 2000; Yaari, 1997). In this paper we take a different approach: instead of relying on word co-occurrence alone, we use machine learning to build a complex model of segment relatedness that combines word co-occurrence with other information, such as lexical chains, cue phrases, segment length, punctuation and tense patterns. We then combine this model with a hierarchical clustering algorithm to derive the inter-paragraph tree structure of a text. The results show significant improvements over an approach based on word co-occurrence alone.

2 Combining Clustering and Machine Learning

We use Hierarchical Agglomerative Clustering (see e.g. Everitt (1993)) as our clustering method. The algorithm starts with the set of elementary segments, in our case the paragraphs of a text, and then iteratively chooses a pair of adjacent segments to merge until only one segment (corresponding to the whole text) is left. The result is a binary tree whose leaves correspond to paragraphs and whose intermediate nodes correspond to intermediate discourse segments. Most discourse theories allow non-binary structures. For example, the *List* relation in RST can be non-binary. But the majority of structures are binary. In our data set more than 95% of the inter-paragraph structures were binary. Hence, binary trees seem a good approximation.

In our use of clustering we draw on an idea from Yaari (1997) who uses this technique as a first step in his hierarchical segmentation algorithm. However, where Yaari uses a similarity measure based on word co-occurrence to decide which segments should be merged, we use a machine learnt model of segment relatedness. Since we will compare our model against Yaari's measure, we describe the latter in more detail here.

Yaari uses a cosine measure to define segment similarity (cf. Salton and Buckley, (1994)):

$$similarity(S_i, S_j) = \frac{\sum_t w_{t,S_i} w_{t,S_j}}{\sqrt{\sum_t w_{t,S_i}^2 w_{t,S_j}^2}} \quad (1)$$

Here t ranges over the set of terms in the text. Terms are extracted by removing closed-class words from the text and then stemming it using Porter's stemmer (Porter, 1980). Each term is weighted, where w_{t,S_i} is the weight assigned to term t in segment S_i, defined as the product of three factors: the frequency of the term in the segment, the relative frequency of the term in the text, and the general significance of the term in a corpus ($Gsig_t$):

$$w_{t,S_i} = f_{t,S_i} \times \frac{f_t}{f_{max}} \times Gsig_t \quad (2)$$

$$Gsig_t = log \frac{N}{N_t} \quad (3)$$

In the definition of $Gsig_t$, N is the number of files in the corpus (Yaari uses the British National Corpus[1]) and N_t is the number of files containing the term t.

We take a different approach. Instead of basing the decision to merge two segments on word co-occurrence alone, we use supervised machine learning to combine several contextual features (including word co-occurrence, see Section 4) into a model that assesses how likely two segments are to be related. The segment pair that scores highest is merged. We used a maximum entropy learner (Ratnaparkhi, 1998) to train our model, but any machine learnt classifier that returns a probability distribution over possible outcome (e.g. *merge*, *don't merge*) or at least ranks them would be suitable.

3 Data

The RST Discourse Treebank (RST-DT) (Carlson et al., 2002) was used for training and testing. It contains 385 Wall Street Journal articles from the Penn Treebank, which are manually annotated with discourse structure in the framework of Rhetorical Structure Theory (RST) (Mann and Thompson, 1987). The set is divided into a training set (342 texts) and a test set (43 texts). 52 texts, selected from both sets, were annotated twice. We use these to estimate human agreement on the task.

Since we focus only on inter-paragraph structure, intra-paragraph structure was discarded. In most cases the discourse structure of a text obeyed paragraph boundaries, but about 21% of the paragraphs did not correspond to a discourse segment. One way to deal with such cases is by removing them from the training set but since the training set is already relatively small, we decided instead to replace them by the inter-paragraph tree which comes closest to the original structure.

In most cases where discourse structure does not follow paragraph structure the deviation is relatively minor. For example, Figure 1 shows 10 edus (numbered 1 to 10) in 3 paragraphs (A to C, indicated

[1] `http://www.hcu.ox.ac.uk/BNC/`

by boxes). There is no discourse segment corresponding to paragraph B because a subsegment of the paragraph (consisting of edus 4 to 6) merges with the previous paragraph and only then is the resulting segment merged with the last edu of B (i.e. number 7). However, there is a discourse segment corresponding to the two paragraphs A and B, i.e. the structure in Figure 1 maps relatively easily to the inter-paragraph structure ((AB)C) (as opposed to (A(BC))).

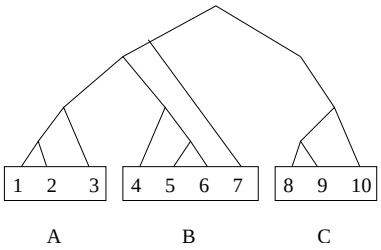


Figure 1: Unambiguous inter-paragraph structure

However, for 8% of paragraphs the mapping was less straightforward. For example, in the tree in Figure 2 some of B's edus attach to the left and some to the right. Hence it is not immediately clear whether one should map to ((AB)C) or (A(BC)). In these cases we used majority voting to resolve the ambiguity, i.e. if most of the edus of a paragraph attached to the left (as in Figure 2) the paragraph was merged with its left neighbour otherwise it was merged with its right neighbour. Hence, the tree in Figure 2 is assumed to have the structure ((AB)C).

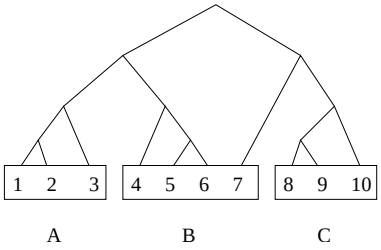


Figure 2: Ambiguous inter-paragraph structure

The few non-binary structures in the training set were binarised by replacing them with left-branching binary structures.

Since we want to predict the likelihood of merging two segments, each pair of adjacent segments (of any size) can be treated as a training example. Segment pairs that are contained in a discourse tree are positive examples and segment pairs not contained in the tree are added as negative examples. For instance, the tree in Figure 3 contains 3 positive training examples (A+B, C+D, and AB+BC) and 7 negative examples (B+C, AB+C, A+BC, BC+D, B+CD, ABC+D, and A+BCD). Pairs of non-adjacent segments, e.g. A+D, were ignored because they are not permitted under the assumption that discourse structure is a tree with non-crossing branches (i.e. their probability is 0).

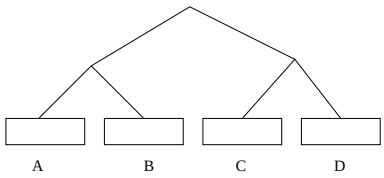


Figure 3: Tree showing inter-paragraph structure

The 342 texts in the RST-DT training set gave rise to 1,830 positive and 185,691 negative training examples.

4 Feature Set

Each training example is described by a set of features. The features were deliberately kept fairly shallow, i.e. they make use only of tokenisation, part-of-speech and sentence boundary information (all of which were taken from the original Penn Treebank mark-up). They do not require any deep processing, such as parsing.

The model uses features from 7 areas: segment position, segment length, term overlap, punctuation, tense, cue phrases, and lexical chains.

Segment position This set comprises 3 features, indicating whether the left (right) segment of the pair is the first (last) in the text and whether the merged segment would be in the beginning, middle or end of the text. The motivation for these features is that the beginning and end of a text often have a special discourse role (at least in this domain), e.g. the first paragraph frequently leads into the text, while the last often provides a summary.

Segment length This set consists of 6 features: the number of words, sentences, and paragraphs of the left and right segment. Segment length can often be a clue as to whether two segments should be merged. For example, very long segments are not normally merged with very short segments unless the short segment has a special position, e.g. is the first or last of the text.

Term overlap We use the formulae in Section 2 to calculate term overlap. This yields a real-valued score between 0 and 1, which was quantised by breaking the range into 10 equal intervals.

Punctuation This set comprises 7 features: the final punctuation mark of the left segment and whether the left (right) segment contains, starts with, or ends with a quotation mark. The presence of quotations in both segments may indicate that they are related and so should increase their merging

probability. Likewise, the final punctuation mark can sometimes be an important clue, e.g. if the left segment ends with a question mark, the next segment might provide an answer to the question and this should increase the merging probability.

Tense We use 6 tense features: the first, last, and majority tense of the left (right) segment. Tense information was obtained by using regular expressions to extract verbal complexes from the part-of-speech tagged text and then determine their tense. Tense often serves as a cue for discourse structure (Lascarides and Asher, 1993; Webber, 1988b). A shift from simple past to past perfect, for instance, can indicate the start of an embedded segment.

Cue phrases This set comprises 4 features. The first three features are reserved for potential cue phrases in the first sentence of the right segment. Cue phrases are identified by scanning a sentence (or the first 100 characters of it, whichever is shorter) for an occurrence of one of the cue phrases listed in Knott (1996). We have three features to be able to deal with multiple cue phrases (e.g. *But because...*). In this case, the feature `first cue phrase` will be assigned the first cue word (*but*), `second cue phrase` the second cue word (*because*) and so on. Cue phrases are often ambiguous between syntactic and discourse use, as well as among different rhetorical relations. While our algorithm does not attempt proper disambiguation between syntactic and discourse usage, some non-discourse usages are filtered out on the basis of part-of-speech information. For example, *second* can be an adverb (as in Example 4) as well as an adjective (as in Example 5) but when used as a discourse marker it is usually an adverb.

(4) Second, the extra savings would spur so much extra economic growth that the Treasury wouldn't suffer.

(5) It was announced yesterday that the profits have fallen for the second year in a row.

The fourth cue phrase feature encodes whether the first sentence of the right segment contains a discourse anaphor, i.e. an anaphor which refers to a discourse segment rather than a real world entity, and if so which it is. An example is *that* in Example 6 (cf. Webber (1988a)). We do not attempt proper anaphora resolution, instead we treat first sentence occurrences of *this* and *that* as discourse anaphors if they seem to be complete NPs, e.g. are directly followed by a verb. This method potentially overgenerates as these expressions could still refer to a preceding NP and it potentially undergenerates as *it* can sometimes also refer to discourse segments. However, previous research has found that demonstrative anaphors rarely refer to NPs, while *it* rarely refers to discourse segments (Webber (1988a)).

(6) It's always been presumed that when the glaciers receded, the area got very hot. The Folsum men couldn't adapt, and they died out. That's what is supposed to have happened.

Lexical chains This set comprises 28 features. The idea of using lexical chains as indicators of lexical cohesion goes back to Morris and Hirst (1991). A lexical chain is a sequence of semantically related words and can indicate the presence and extent of subtopics in a text. We use our own implementation to compute chains.

A distinction is made between common noun chains, which are built on the basis of semantic relatedness using WordNet (Miller et al., 1990), and proper noun chains, which contain nouns not found in WordNet and are based on co-reference rather than semantic relatedness. As a first step, nouns are extracted and lemmatised using the *Morpha* analyser (Minnen et al., 2001) and then looked up in WordNet. If no entry can be found and the noun is a compound noun, the first lexeme is removed and the remaining string is looked up until an entry is found or only one lexeme remains. For example, if *chief executive officer* could not be found in WordNet, our algorithm would try *executive officer* and then *officer*. Each term that can be found in WordNet is treated as a potential element of a common noun chain, even if it is strictly speaking a proper noun. This allows chains like *Mexico – country – Chile*. If a noun cannot be found in WordNet it is treated as a potential member of a proper noun chain.

A potential problem for lexical chains is that words can have more than one sense and semantic relatedness depends on the sense rather than the word itself. We take a greedy approach to word sense disambiguation: while a noun is in a chain on its own, the algorithm is agnostic about its sense but this changes when another noun is added. A new noun n is added by comparing each of its senses to the senses of the members of existing chains and a score is calculated for each sense pair depending on the WordNet distance between them. Only distances up to an empirically set cut-off point count as a match, where the cut-off point depends on whether the term is a proper noun and on the nature of the semantic relation (only hypernym, hyponym and synonym relations are considered). If there are one or more matches, the noun is added with the sense that achieved the highest score to the chain c with which this score was achieved. If c contains only one noun n', all senses of n' are removed apart from the sense with which the match was achieved. Repeated occurrences of the same noun in a text are placed in the same chain, i.e. it is assumed that a word keeps

its sense throughout the text.

When all common noun chains have been built, the significance of each chain is assessed and chains that are not considered significant are deleted. To be considered significant a chain has to contain at least two nouns (or two occurrences of the same noun) and the *Gsig* (see equation 3) averaged over all its elements either has to be relatively high or the chain has to be relatively long compared to the overall length of all other chains, where length is measured as the number of "hits" a chain has in the text.[2] For example, Wall Street Journal articles frequently contain expressions of date, such as *December*, *month*, *Tuesday*, but these do not normally make interesting chains as they are high frequency expressions and the appearance of various date expressions throughout the text does not normally indicate a subtopic, i.e. it does not mean that the text is "about" time and date expressions. However, if time and date expression are *very* frequent in the text this may be an indicator that these do indeed form a subtopic and that the chain should be retained.

Proper noun chains are built for words not in WordNet. Chain membership is determined on the basis of identity, i.e. a chain contains repeated occurrences of the same noun. Some proper noun phrase matching is done. For example, the expressions *U.S. District Judge Peter Smith*, *Judge Smith*, and *Mr. Smith* are treated as referring to the same entity and can therefore be placed in the same chain. When all proper noun chains have been built, those that contain only one element (i.e. one occurrence of a term) are removed. All other chains are retained.

Note, unlike most approaches that make use of lexical chains, we do not break a chain in two if too many sentences intervene between the individual chain elements; chains are continued as long as new elements can be found. However, the algorithm keeps track of where in the text chain elements were found. If a chain skips one or two paragraphs this is actually an important clue because it can indicate that the two paragraphs form an embedded segments. This is especially true if there are also chains which start in the left paragraph and end in the right. For example, Figure 4 shows a text with 5 paragraphs (A to E) and two lexical chains. Chain 1 spans the whole text but skips paragraphs B and C, while chain 2 only spans paragraphs B and C. A situation like this makes it likely that B and C should be merged before either of them is merged with another paragraph. Hence Tree 1 in Figure 5 should be more likely than Tree 2. For this analysis it is crucial that chain 1 is not broken into two. Obviously for very long texts the situation will be slightly different and there will be circumstances where a chain should be broken.

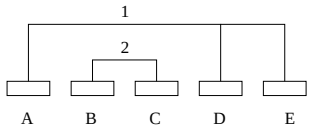
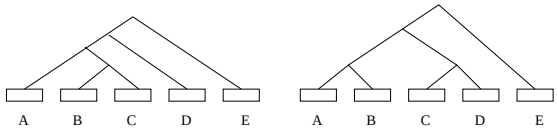


Figure 4: A chain skipping two segments



(a) Tree 1 (b) Tree 2

Figure 5: Possible tree structures

The individual chain features distinguish between proper and common noun chains. The reason for this is that the former are likely to be more reliable as they are based on term identity rather than semantic relatedness. For both types the features encode whether and how many chains:

- span the two segments
- exclusively span the two segment (i.e. start in the left segment and end in the right)
- start or end in the left (right) segment
- skip both of the segments
- exclusively skip the two segments (i.e. skip both segments but none of the neighbouring segments)
- skip one of the two segments
- exclusively skip the left (right) segment

To combine all features, we trained a maximum entropy model (see e.g. Ratnaparkhi (1998)) on the training set. Each feature is automatically assigned a weight reflecting its usefulness. Once trained the model outputs a probability distribution over the classes *merge* and *don't merge* for each pair of segments, based on the weighted features for the pair. To prevent the model from overfitting we used a feature cut-off of 10, i.e. feature-value pairs that occur less than 10 times in the training set were discarded.

5 Experiments

As described in Section 2, the trained model was combined with the clustering method to build trees for the test set. These were evaluated against the manually built discourse trees. Precision (P) and recall (R) were defined in accordance with the PARSEVAL measures (Black et al., 1991), i.e. precision is

[2] Both thresholds were empirically set.

	random	RB	TO	LB	ME	ME−LC	ME−TO	ME−LCTO	human*
P	44.37%	36.76%	49.98%	53.52%	58.06%	55.86%	57.12%	55.26%	64.37%
R	46.71%	40.35%	52.42%	56.23%	60.78%	58.29%	59.70%	57.69%	64.60%
F	45.05%	37.58%	50.79%	54.27%	59.00%	56.66%	58.00%	56.07%	64.34%

Table 1: Results on RST-DT test set (* on doubly annotated set)

defined as the number of correct nodes (i.e. matching brackets) divided by the number of nodes in the automatically built tree and recall as the number of correct nodes divided by the number of nodes in the manually built tree. Precision and recall are combined in the f-score (F), defined as $\frac{2PR}{P+R}$.

Table 1 shows the results. We compared the performance of our model (ME) to Yaari's (1997) method of building trees based on term overlap (TO). In addition, three baselines were used: merging segments randomly (results averaged over 100 runs), producing a right-branching tree by always merging the last two segments (RB) and producing a left-branching tree by always merging the first two segments (LB). Finally, an upper bound was calculated by comparing the trees for the doubly annotated text files in the RST-DT. Note that the doubly annotated data set is slightly different from the test set, hence the upper bound can only give an indication of the human performance on this task.

The maximum entropy model outperforms all other methods on precision, recall and f-score. The difference in correct discourse segments (true positives) between our method and the next best (i.e. left-branching) is statistically significant (one-tailed paired t-test, t=1.72, df= 37, $p < 0.05$).

Interestingly, Yaari's word co-occurrence based method (TO) is outperformed by left-branching trees (LB). Furthermore, while Marcu (2000) argues that right-skewed structures should be considered better than left-skewed structures, in our experiments, the latter actually outperform the former, i.e. inter-paragraph structure in the RST-DT is predominantly left-branching. Predictably, human performance is better than any of the automatic methods.

To investigate the contribution of our different feature sets we re-trained the model after removing lexical chains (ME−LC), term overlap (ME−TO) and lexical chains and term overlap (ME−LCTO). The results are also shown in Table 1. As can be seen, removing lexical chain features results in more performance loss than removing term-overlap features. Thus it seems that lexical chains are more useful for the task than term-overlap. However, the performance difference between ME−LC and ME−TO is not statistically significant (t=0.96, df=37, $p > 0.05$). Removing both feature sets (ME−LCTO) still leads to a better performance than is achieved by left-skewed clustering (LB), which indicates that other features, such as tense and cue word features, are able to compensate to some degree for the absence of chain and term overlap features. But the difference between LB and ME−LCTO is again not statistically significant (t=1.24, df=37, $p > 0.05$).

So far we have not said much about the rhetorical relations that hold between larger discourse segments. In fact, assigning relations to higher-level structures is easier than doing so for inter-sentence structures. One reason for this is that there is much less variation on inter-paragraph level. For example, the RST-DT contains 111 different relations but only 64 of these are used at inter-paragraph level. Furthermore, the most frequent relation on inter-paragraph level (*Elaboration-additional*) accounts for a much larger percentage (37%) of all relations used at this level than does the most frequent relation on intra-paragraph level (*List*, 13%). Hence, always predicting *Elaboration-additional* would already achieve 37% accuracy. Being able to reliably distinguish between *Elaboration-additional* and the second most frequent inter-paragraph relation, *List*, would guarantee 53% accuracy. In contrast, correctly predicting the two most frequent relations on intra-paragraph level would only achieve 26% accuracy. We plan to address the prediction of rhetorical relations between larger discourse segments in future work.

6 Conclusion

In this paper, we proposed a machine learning approach for predicting inter-paragraph structure. Inferring inter-paragraph structure can be seen as a subtask of discourse parsing. While low-level discourse parsing relies to a large extent on cue phrases as predictors for rhetorical structure, these are less useful for high-level structure. As an alternative, word co-occurrence measures have been suggested. In this paper, we took a different approach and employed a machine learning approach to build a complex model of segment relatedness which was then combined with a clustering algorithm. The use of machine learning enabled us to combine contextual cues from several areas, such as word co-

occurrence, lexical chains, changes in tense patterns, punctuation etc. Our model outperformed a word co-occurrence measure as well as left- or right-branching trees.

In future work, we plan to extend our approach to predict rhetorical relations between paragraphs. While an empirical analysis revealed that one can achieve a relatively high accuracy by just predicting the most frequent relation, it is still worthwhile to investigate how much better one can do with more sophisticated methods. There is also clearly a relationship between structure and relation. For example, non-binary structures are more likely to be joined by a *List* relation than by an *Explanation* relation. Hence, inferring structure and predicting relations should be interleaved.

It would also be interesting to investigate, whether it would be useful to relax the constraint that inter-paragraph structure is a tree with non-crossing branches. Some researchers have suggested that higher level discourse structure may be better represented if one allows crossing branches (Knott et al., 2001). In principle, the approach suggested here could be used to generate such structures if one removed the constraint that only adjacent segments can be merged.

Finally, it remains to be seen to what extent our results carry over to other domains. So far, the RST-DT remains the only publicly available data set annotated with discourse structure but a larger corpus is currently annotated as part of the Penn Discourse Treebank project.[3] It would be interesting to apply our methods to this data set as well.

References

E. Black, S. Abney, D. Flickinger, C. Gdaniec, R. Grishman, P. Harrison, D. Hindle, R. Ingria, F. Jelinek, J. Klavans, M. Liberman, M. Marcus, S. Roukos, B. Santorini, T. Strzalkowski. 1991. A procedure for quantitatively comparing the syntactic coverage of English grammars. In *Proceedings of the 4th DARPA Workshop on Speech and Natural Language*, 306–311.

L. Carlson, D. Marcu. 2001. Discourse tagging manual. Technical Report ISI-TR-545, Information Sciences Institute, Los Angeles, CA, 2001.

L. Carlson, D. Marcu, M. E. Okurowski. 2002. RST Discourse Treebank. Linguistic Data Consortium. `http://www.ldc.upenn.edu/Catalog/CatalogEntry.jsp?catalogId=LDC2002T07`, 2002.

B. Everitt. 1993. *Cluster Analysis*. Edward Arnold, London, 3rd edition.

A. Knott, J. Oberlander, M. O'Donnell, C. Mellish. 2001. Beyond elaboration: The interaction of relations and focus in coherent text. In T. Sanders, J. Schilperoord, W. Spooren, eds., *Text Representation: Linguistic and Psycholinguistic Aspects*, 181–196. Benjamins, Amsterdam.

A. Knott. 1996. *A Data-Driven Methodology for Motivating a Set of Coherence Relations*. Ph.D. thesis, Department of Artificial Intelligence, University of Edinburgh.

A. Lascarides, N. Asher. 1993. Temporal interpretation, discourse relations and common sense entailment. *Linguistics and Philosophy*, 16(5):437–493.

W. C. Mann, S. A. Thompson. 1987. Rhetorical structure theory: A theory of text organization. Technical Report ISI/RS-87-190, Information Sciences Institute, Los Angeles, CA, 1987.

D. Marcu. 2000. *The Theory and Practice of Discourse Parsing and Summarization*. MIT Press, Cambridge, MA.

G. A. Miller, R. Beckwith, C. Fellbaum, D. Gross, K. J. Miller. 1990. Introduction to WordNet: An on-line lexical database. *International Journal of Lexicography*, 3(4):235–312.

G. Minnen, J. Carroll, D. Pearce. 2001. Applied morphological processing of English. *Natural Language Engineering*, 7(3):207–223.

J. Morris, G. Hirst. 1991. Lexical cohesion computed by thesaural relations as an indicator of the structure of text. *Computational Linguistics*, 17(1):21–48.

M. F. Porter. 1980. An algorithm for suffix stripping. *Program*, 14:130–137.

A. Ratnaparkhi. 1998. *Maximum Entropy Models for Natural Language Ambiguity Resolution*. Ph.D. thesis, Computer and Information Science, University of Pennsylvania.

G. Salton, C. Buckley. 1994. Term-weighting approaches in automatic text retrieval. *Information Processing and Management*, 24(5):513–617.

R. Soricut, D. Marcu. 2003. Sentence level discourse parsing using syntactic and lexical information. In *Proceedings of the 2003 Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*.

H. A. Stark. 1988. What do paragraph markings do? *Discourse Processes*, 11:275–303.

B. L. Webber. 1988a. Discourse deixis: Reference to discourse segments. In *Proceedings of the 26th Annual Meeting of the Association for Computational Linguistics*, 113–122.

B. L. Webber. 1988b. Tense as discourse anaphor. *Computational Linguistics*, 14(2):61–73.

Y. Yaari. 1997. Segmentation of expository texts by hierarchical agglomerative clustering. In *Proceedings of the 2nd International Conference on Recent Advance in Natural Language Processing*, 59–65.

[3] `http://www.cis.upenn.edu/~pdtb/`