Learning to Identify Fragmented Words in Spoken Discourse

Piroska Lendvai
ILK Research Group
Tilburg University
The Netherlands
p.lendvai@uvt.nl

Abstract

Disfluent speech adds to the difficulty of processing spoken language utterances. In this paper we concentrate on identifying one disfluency phenomenon: fragmented words. Our data, from the Spoken Dutch Corpus, samples nearly 45,000 sentences of human discourse, ranging from spontaneous chat to media broadcasts. We classify each lexical item in a sentence either as a completely or an incompletely uttered, i.e. fragmented, word. The task is carried out both by the IB1 and RIPPER machine learning algorithms, trained on a variety of features with an extensive optimization strategy. Our best classifier has a 74.9% F-score, which is a significant improvement over the baseline. We discuss why memory-based learning has more success than rule induction in correctly classifying fragmented words.

1 Introduction

Although human listeners are good at handling disfluent items (self-corrections, repetitions, hesitations, incompletely uttered words and the like, cf. Shriberg (1994)) in spoken language utterances, these are likely to cause confusion when used as input to automatic natural language processing (NLP) systems, resulting in poor human-computer interaction (Nakatani and Hirschberg, 1994; Eklund and Shriberg, 1998). Detecting disfluent passages can help clean the spoken input and improve further processing such as parsing.

By treating fragments we cover a considerable portion of the occurring disfluencies as incompletely uttered words often occur as part of a speaker's self-repair (Bear et al., 1992; Nakatani and Hirschberg, 1994). Moreover, if an incompletely pronounced item is identified, we thereby determine the interruption point, a central phenomenon in disfluencies (Bear et al., 1992; Heeman, 1999; Shriberg et al., 2001). The surroundings of this disfluency element are to be treated with greater care, as before an interruption point there might be word(s) meant to be erased (called the reparandum), whereas the word(s) that follow it (the repair) might be intended to replace the erased part, cf. the following example:

> het veilig gebruik **van interne–***[1] |[2] sorry **van electronic commerce**[3]
>
> (the safe usage **of interne–*** sorry **of electronic commerce**).

Previous studies in the field of applying machine learning (ML) methods to disfluencies either employ classification and regression trees for identifying repair cues (Nakatani and Hirschberg, 1994) and for detecting disfluencies (Shriberg et al., 2001), or they use a combination of decision trees and language models to detect disfluency events (Stolcke et al., 1998) or to model repairs

[1]reparandum
[2]interruption point
[3]repair

(Heeman, 1999). Although Spilker et al. (2001) and Heeman (1999) observe that word fragments pose an unsolved problem in processing disfluencies, often the presence of a disfluent word is regarded as an integral property of a speech repair and is employed as a readily available feature in the ML tool (Nakatani and Hirschberg, 1994). However, automatic identification of a fragment is not straightforward, unlike the recognition of other disfluency types, such as filled pauses ("uhm").

Our study investigates the feasibility of automatically detecting fragments, for which we propose using learning algorithms, since they suit this problem formalised as a binary classification task of deciding whether a word is completely or incompletely uttered. The current paper first describes our large-scale experimental material, after which the learning process is explained, with particular emphasis on the features employed by the two different learning algorithms and the experimental setup. We also introduce the method of iterative deepening used for optimizing the parameters of both the memory-based and the rule induction classifier. In Section 4 the results of the fragment identification task are reported and the behaviour of the learners is analysed. The last section evaluates our approach and outlines the directions for further investigation.

2 The data

For our research the morphologically and syntactically annotated portion of the Spoken Dutch Corpus (Oostdijk, 2002) of Development Release 5 was used, which incorporates 203 orthographically transcribed discourses of various genres, sampled from diverse regions of The Netherlands and Flanders (Belgium). The transcribed sentences are tagged morpho-syntactically, and a complete and corrected syntactic dependency tree is built manually for each utterance.

The discourses are grouped into 10 levels of spontaneity, extending from television and radio broadcasts, interviews, lectures, meetings, to spontaneous telephone conversations. The number of speakers involved ranges from 1 (newsreading) to 7 (parliamentary session). As disfluencies are reported to occur both in dialogue and monologue (Shriberg et al., 2001), we did not weed out discourses from the corpus that feature only one speaker.

Altogether, our material counts 340,840 lexical tokens in 44,939 sentences. The tokens are marked for filled pauses, coordinating conjunctions ("and then"), grammatically or phonetically ill-formed but complete words ("hij blelde [belde] niet", i.e., "he did not clal [call]") and fragmented words ("hij be–* belde niet", i.e., "he did not c–* call"). There are 3,137 fragmented words in our material, constituting 0.9% of the lexical tokens. The average sentence length in the corpus in 7.6 words. Interestingly, the average length of sentences containing one or more fragments is much higher, namely 18.2 words. Oviatt (1995) finds indeed that longer utterances lead to more disfluencies than short ones.

The work of (Bear et al., 1992) reports that in 60% of self-repairs a fragment is involved, whereas this rate is 73% in the study of (Nakatani and Hirschberg, 1994) and 26% in (Heeman, 1999). In our material such a rate cannot be directly computed, since not all kinds of repairs are separately annotated in the CGN corpus. However, if those passages that are excluded from the syntactic trees are counted as self-repair events, we find that in 20% of those events a fragmented word is present.

3 Learning experiments

3.1 Selecting cues

Identification of cues for detecting incompletely uttered words was based on close inspection of our corpus and on the literature. In the current paper we focus on using word-based information only, in order to investigate the feasibility of fragment detection with readily available features. This is in line with Heeman and Allen (1994) who assume local context to be sufficient in detecting most speech repairs, without taking syntactic well-formedness or speech prosody into consideration.

Table 1 lists the 22 features that we extracted automatically from the corpus material, subdivided into four groups according to the aspect they describe. Five lexical string features represent the focus word itself and its neighboring two left and two right unigram contexts (if any). Four binary features mark if overlap in wording or in initial letter occurs between the focus item and/or its context. Matching words or word-initial letters are often to be found both at the reparandum onset and the repair onset, as in a correction of *Arnhem*:

> "de werkloosheid **in** Arnhe–* **in** Nijmegen" (the unemployment **in** Arnhe–* **in** Nijmegen).

The last member of this group is a ternary feature, showing the extent to which left and right context words overlap (0-1-2 letters).

Four attributes in the feature vector describe general properties of the given utterance, indicating sentence length and the focus item's relative position in the sentence, as well as the total amount of filled pauses and of identical lexical sequences in the sentence. By employing these features we allow the learners to make use of possible correlations between certain values of these attributes and the potential presence of an incomplete word. Finally, eight binary features convey information about those phenomena in the two left and two right context items of the focus that, according to empirical studies, might be repair-signalling: filled pauses, coordinating conjunctions, as well as the presence of items that either elicit or often co-occur with disfluencies: named entities[4], unintelligible mumbling, and laughter. Except for named entities, these features were identified using the corpus markups.

Some seemingly redundant features of the *Overlap* and *Context-type* groups deliberately reintroduce properties that are implicitly present in the lexical features. By making these explicit we ensure that the learners, unable to capture sub-wordform similarities between the features, will not ignore possibly important information.

3.2 Data preparation

In order to conduct 10-fold cross-validation experiments, the discourses were randomized and subsequently partitioned into 90% training sets and 10% test sets. The sizes of the ten resulting training sets are roughly similar and so are the sizes of the ten resulting test sets. Partitioning was discourse-based to ensure that no material from one and the same dialogue could be present both in the training and the test set of a partition.

[4]i.e., capitalized words. Sentence-initial words are not capitalized in the corpus.

Aspect	Features
Lexical	(1) Left2 context item (2) Left1 (3) Focus item (4) Right1 (5) Right2
Overlap	(1) Left1/Right1 items identical (2) Left1/Right2 items identical (3) First letter of Left1/First letter of Focus overlap (4) First letter of Focus/First letter of Right1 overlap (5) First and/or second letter of Left1/Right2 overlap
General	(1) Number of tokens in utterance (2) Proportional position of focus item (3) Amount of filled pauses in sentence (4) Amount of lexical repetitions
Context-type	(1) Left2 is filled pause (2) Left1 is FP (3) Right1 is FP (4) Right2 is FP (5) Named entity in context of focus item (6) Laughter (7) Unintelligible material (8) Coordinating conjunction

Table 1: Overview of the employed features, grouped according to their aspect.

We automatically generated learning instances from each word form token, extracting the values corresponding to the features described above. The class symbol of the learning instance (*F*ragment or *N*on-Fragment) indicates whether the focus item is an incompletely uttered word or not. Subsequently, the extracted feature values and the class symbol were arranged into a flat, fixed-length format of 23 elements, illustrated in Table 2. For example, the binary representation of a letter-overlap phenomenon can be observed in line 7 : the first letter of the fragmented focus item "ru" overlaps with the first letter of its immediate right context (*R1*) "rugbyteam", so the *O4* feature, representing the fourth feature of the *Overlap* group, is set to 1.

3.3 The learners

We used two learning algorithms to carry out fragment detection. The TiMBL 4.3 software package (Daelemans et al., 2002) incorporates a variety of memory-based pattern classification algorithms, each with fine-tunable metrics. We chose for working with the IB1 algorithm only (the default in TiMBL), taking the classical k-nearest neighbor approach to classification: looking for those instances among the training data

L2	L1	Focus	R1	R2	O1	O2	O3	O4	O5	G1	G2	G3	G4	C1	C2	C3	C4	C5	C6	C7	C8	Class
_	_	ggg	ja	hij	0	0	0	0	0	9	0.00	3	0	0	0	1	0	0	0	0	0	N
_	ggg	ja	hij	is	0	0	0	0	0	9	0.11	3	0	0	0	0	0	0	1	0	0	N
ggg	ja	hij	is	uh	0	0	0	0	0	9	0.22	3	0	0	1	0	1	0	1	0	0	N
ja	hij	is	uh	met	0	0	0	0	0	9	0.33	3	0	1	0	1	0	0	0	0	0	N
hij	is	uh	met	ru	0	0	0	0	0	9	0.44	3	0	0	0	0	0	0	0	0	0	N
is	uh	met	ru	rugbyteam	0	0	0	0	0	9	0.56	3	0	0	1	0	0	0	0	0	0	N
uh	met	ru	rugbyteam	uh	0	0	0	1	0	9	0.67	3	0	1	0	0	1	0	0	0	0	Fr
met	ru	rugbyteam	uh	...	0	0	1	0	0	9	0.78	3	0	0	0	1	0	0	0	0	0	N
ru	rugbyteam	uh	...	_	0	0	0	0	0	9	0.89	3	0	0	0	0	0	0	0	0	0	N
rugbyteam	uh	...	_	_	0	0	0	0	0	9	1.00	3	0	0	1	0	0	0	0	0	0	N

Table 2: Ten instances built from the ten elements of the utterance "<laughter> yes he is with ru–* rugby team uh ..." : the focus item in windowed context, the numeric features and the class symbol.

that are most similar to the test instance, and extrapolating their majority outcome to the test instance's class. Memory-based learning is often called "lazy" learning, because the classifier simply stores all training examples in memory, without abstracting away from individual instances in the learning process.

In contrast, our other classifier is a "greedy" learning algorithm, RIPPER (Cohen, 1995), version 1, release 2.4. This learner induces rule sets for each of the classes in the data, with built-in heuristics to maximize accuracy and coverage for each rule induced. This approach aims at discovering the regularities in the data, and represent it by the simplest possible rule set. Rules are by default induced first for low-frequency classes, leaving the most frequent class the default rule. This suits our purpose well as we are interested in making rules for the minority Fragment class.

3.4 Optimization with iterative deepening

For both classifiers the learning process consisted of two parts per data partition. First, an iterative deepening search algorithm (Kohavi and John, 1997; Provost et al., 1999) was used to automatically construct a large number of different learners by varying the parameters of IB1 and of RIPPER. These learners were systematically trained on portions of the 90% training set, starting with a small sample and doubling it over the iterative optimization rounds. This test data was variedly represented by all possible combinations of our four feature groups in the case of IB1 experiments, in order to exploit the benefits of interleaved parameter optimization and feature selection (Daelemans and Hoste, 2002). In experiments with RIPPER the data was represented by all the features, assuming that this algorithm's architecture will abandon useless features anyway . At the same time, RIPPER was allowed to arbitrarily add redundant features to the learning instances.

The test set for the iterative deepening experiments consisted of about 11,000 instances taken from elsewhere in the 90% training set. Due to the sparse distribution of the Fragment class in the data (recall that less than 1% of the words are fragments), it was important to allow the learners access to enough test material on the Fragment class during the optimization. Therefore we boosted this test set with Fragment-class instances from the remaining (i.e., selected neither for the training nor for the test set) portion of the original 90% training set.

Throughout the learning experiments we worked with the evaluation metrics of predictive accuracy, as well as the Fragment class's precision, recall, and F-score[5]. In the embedded rounds of the iterative deepening process the classifiers recursively searched for the optimal combination of parameter setting and feature selection by maximizing the F-score performance on the Fragment class. In each round the learners were ranked according to their performance. The lower half of these were discarded, whereas the well-performing combinations were re-trained.

Both the size of our material and the search space of the task were large, thus conducting

[5]The harmonic mean of precision and recall. We employ the unweighted variant of F, defined as $2PR/(P+R)$ (P = precision, R = recall) (van Rijsbergen, 1979).

an exhaustive search for our study was computationally not feasible. The iterative deepening algorithm conducted 4,301 learning experiments with IB1 and 3,187 with RIPPER during the optimization rounds for each partition even with this heuristic search that constrained the amount of learners that got optimized by the iterative rounds, the size of data the learners were trained and tested on, the choice of classifier parameters to be optimized, as well as the values of these parameters.

In IB1 the following settings were tested (for details, cf. (Daelemans et al., 2002)):

- the number of nearest neighbors used for extrapolation were odd numbers varied between 1 and 25
- the distance weighting metric of the k nearest neighbors was either majority class voting or inverse distance weighting
- for computing the similarity between features either the overlap function or the modified value difference metric (MVDM) function was used
- the frequency threshold that allows calculation of MVDM instead of overlap was varied between 1-10
- for estimating the importance of the attributes in the classification task either no weighting, or Gain Ratio, or Chi-squared weighting was used.

For the RIPPER algorithm the learners to be optimized were created by systematically varying the following parameters and their values:

- negative tests on the feature attributes were either allowed or disallowed
- the number of optimization rounds on the induced ruleset was within a range of 0-3
- the amount of learning instances to be minimally covered by each rule was set to values in the range of 1-5
- the coding cost of theory was allowed to be multiplied by various values, leading to simplification or complication of hypotheses
- the loss ratio of costs was varied between 0.5-100.

In the second part of the fragment detection experiments the highest-scoring learner of the given partition was trained on the total 90% training set and tested on the held-out 10% test set, finalizing the 10-fold cross-validation experiment. The performance of these ten classifiers were finally combined in a single figure to represent the average performance of the learning algorithm in the fragment classification task.

3.5 Baselines

In order to evaluate our classifiers, a baseline of the fragment identification task needs to be established. Predicting if a certain word is a completely or an incompletely uttered one can hardly be modelled along simple lines. By constructing a lexicon of all the words in the training portion of the corpus a simple check could determine if a given test item is a suspectedly incomplete word (not being present in the lexicon), or is a complete word (if present in the lexicon).

However, an "in-lexicon" property of an item does not automatically guarantee that the word is a completely uttered element in the given context: there are numerous words in Dutch that are present in even a small lexicon, for example "in" (in), "zo" (so), "nee" (no), "na" (after), 'moe" (tired), and which occur very frequently as fragmented beginnings of some other, longer words. Furthermore, applying this baseline approach to our data, we find that the accuracy (91.4%) and recall (53.6%) figures are reasonable, but precision is very low (2.4%) as all new words in the test set are regarded as fragments. This baseline has a 4.6% F-score.

A second baseline model, that obtains higher precision, is to consider all 1-letter items a fragment. This baseline has an accuracy of 97.4% in detecting fragmented items, with 54.3% precision, 43.9% recall, thus 48.5% F-score. It ignores that there are frequent, legal 1-letter words in Dutch.

4 Results

4.1 Learner Performance

The average performance of IB1 in the 10-fold cross-validation experiments is shown in Table 3. The diversity among the learners per partition is characterized by the mean and standard deviation figures for the four evaluative measures. The optimized IB1 algorithm classifies fragmented words

Learner	Accuracy	Precision	Recall	F-score
In-lexicon baseline	91.4	2.4	53.6	4.6
1-letter baseline	97.4	54.3	43.9	48.5
Default IB1	99.6±0.1	81.3±4.5	65.3±4.4	72.4±4.2
Optimized IB1	99.6±0.1	83.9±3.5	67.7±4.6	74.9±3.9
Default RIPPER	99.3±0.1	98.6±3.5	17.4±2.3	29.5±3.3
Optimized RIPPER	99.4±0.1	81.8±4.6	32.7±4.4	46.5±4.7

Table 3: Results of default and optimized IB1 and RIPPER in 10-fold cross-validation.

with 83.9% precision and 67.7% recall, obtaining a 74.9% F-score, which is a significant improvement over both baseline models. Furthermore, the optimized IB1 classifier (shown in the same table) outperforms the non-optimized IB1's F-score by 2.5 points (significant in a paired t-test, $p < 0.01$).

In order to point out problematic cases for the learner, we examined the classified material and found that it often produced false negatives in cases when a fragmented item resembled a true word (this corresponds to the problems with the In-lexicon baseline), or when fragmented acronyms, named entities or foreign words (e.g. the English word "I") had to be classified. Annotation errors in the corpus lead to similar problems. On the other hand, the same word types caused many false positives as well when it came to classifying non-fragmented but short lexical items, foreign words and named entities.

The outcome of the 10-fold cross-validation experiment with the optimized RIPPER is shown in the bottom line of Table 3. It scores below IB1 and the 1-letter baseline model, producing 99.4% accuracy but only 46.5% F-score in classifying fragmented words. However, the optimized RIPPER produces much better classification results than the default algorithm.

When trained on the total training set with the optimized settings, the number of induced rules is well above one hundred. Our largest ruleset consists of 193 rules. The hypotheses incorporate between one and seven conditions each, mainly conditioning on the immediate right context, particularly when it has the value of "...", indicating an abandoned sentence. The letter overlap between focus word and immediate right context (*O4*) has indeed proven to be a very frequently employed, useful feature, as well as the identity of the focus word. Other attributes often used in the rules are the lexical context items, and features from the *General* group: relative sentence position, sentence length, and the amount of lexical repetitions in the utterance.

We see that, when negation is allowed in the learner, this is mostly applied to the focus word. Namely, when making rules for the Fragment class, the hypotheses forbid the focus item to have certain values such as filled pauses, unintelligible material, and coordinating conjunctions, supposedly because such items are mostly short and occur in similar contexts as fragmented words, but are not fragments themselves.

4.2 Optimized parameters

The interleaved parameter optimization and feature selection process for IB1 resulted in ten learners with identical parameter settings. Namely, the overlap similarity metric worked uniformly best for all data partitions, with k=1, employing the Chi-squared feature weighting metric.

When k is set to 1, IB1's strategy is to return the class of the immediate nearest neighbor, which is, according to the resulting overlap similarity metric, the one having the least difference in a feature-per-feature match between the test instance and a training instance stored in memory. When calculating the differences, the features are ranked according to their importance in the classification task. According to the results of iterative deepening, this importance is defined by the Chi-squared statistic measure, computed by using observed and expected feature value and class co-occurrences.

There is a marked difference between the weights the Chi-squared metric assigns to features, as opposed to those of the default gain ratio metric: Chi-squred statistics considers the focus item's identity most important, followed by the right context (*R1* and *R2*), and the left context (*L1* and *L2*). On the other hand, the gain ratio metric assigns the highest weight to the overlap between the first letter of the immediate left and right context (*O1*), followed by *O4*, and only the third most important feature with a much lower weight is the focus word itself. It is noteworthy that despite the similarity between our optimized settings and the default settings in IB1 (the only difference obtained via iterative deepening being the above metric choice), the optimized learner is able to perform significantly better.

Although the best optimized learners per folds are identical, there are alterations in the way they combine with the feature groups. For the majority of the partitions the best results were obtained when all features were available to the learners. In three partitions a learner that did not exploit all feature groups could outperform those that employed all available features: twice the *Overlap* as well as the *Context-type* attributes were considered unneccessary by the learner, and in one case the *General* features were not beneficial for classification. We see indeed that the Chi-squared metric assigns much lower weights to these feature groups than to the members of the Lexical feature group. Most importantly, the *Lexical* features were always incorporated in the well-performing classifiers during the optimization process, which proves that the identity of the focus word and its immediate context provides the most valuable source in learning the fragment detection task.

For the RIPPER algorithm we observe the same uniformity among the resulting best optimized learners per partition. The best-performing options are always those that allow optimization three times in the rule induction process while forcing each rule to cover at least one example, with the loss ratio value set to 0.5. The optimized value by which the coding cost of the theory is to be multiplied is 0.1 for the top-scoring learners of eight partitions, and is 0.25 in two partitions. This value allows for constructing much more complicated hypotheses than by done by RIPPER's default. There is also divergence among the top learners with respect to allowing negative tests on feature values: in five partitions negation is used by the top learner, whereas in the other half of the cases negation is not employed. Finally, the option of using random features in RIPPER has not proven to be useful.

We assume that by allowing RIPPER to induce more complicated hypotheses than by default, the learning becomes more case-specific, shifting RIPPER in the direction of IB1's strategy, namely not to abstract away from the examples. We indeed see that the induced rules' coverage is mostly well below ten examples. As the option of inducing detailed hypotheses has proven to work optimally for RIPPER, we conclude that the reason why memory-based classification performs better is its approach of taking the specificities of all training instances into consideration instead of generalizing from those.

5 Discussion and Future Work

We tested a memory-based and a rule induction ML algorithm in the task of automatically classifying words in transcripts of spoken Dutch discourses as completely uttered or fragmented ones. We employed readily available, lexically-oriented features in the learning process. The method used for optimizing the two classifiers was iterative deepening search for parameter settings combined with feature selection. We optimized the algorithms by maximizing the F-score performance of a large number of learners constructed by varying the parameter values of IB1 and RIPPER. It is preferable to base evaluation on the harmonic mean of precision and recall measures, as in our data the Fragment class is sparse, thus simply always predicting that a word is not a fragment yields high accuracy scores.

We observe that memory-based learning results in more success than rule induction, as IB1's F-score on the task is 74.9%, and that for RIPPER is 46.5%. Even when RIPPER is allowed to induce very specific rules, it still abstracts away from the data, whereas for IB1 it pays off to consider specific instances. We assume that the iterative deepening method is beneficial for both classifiers, as the optimized parameters turned out to be different from and better performing than the default ones.

For feature selection we did not observe a definitive impact of the iterative deepening search.

Most studies in the field of applying ML to disfluency resolution employ features that are extracted from hand-annotated resources. In the current study we made use of lexical information only, considering that exploiting the gold-standard syntactic annotation of the corpus would give too much advantage to our model, as opposed to a fragment detection task in a real application where no perfect information would be available. It seems intuitive that self-repairs in spoken language are signalled not only verbally but prosodically as well. In the future we plan not only to incorporate prosodic features into our learners, but to use the lexical output of an automatic speech recognizer as well as to generate syntactic information from it automatically. Moreover, we plan to extend our study to identifying other types of disfluency in order to construct a pre-processing module of spoken language utterances.

References

J. Bear, J. Dowding, and E. Shriberg. 1992. Integrating multiple knowledge sources for detection and correction of repairs in human-computer dialog. In *Meeting of the Association for Computational Linguistics*, pages 56–63.

W. W. Cohen. 1995. Fast effective rule induction. In *Proceedings of the Twelfth International Conference on Machine Learning*, Lake Tahoe, California.

W. Daelemans and V. Hoste. 2002. Evaluation of machine learning methods for natural language processing tasks. In *Third International Conference on Language Resources and Evaluation (LREC 2002)*, pages 755–760.

W. Daelemans, J. Zavrel, K. van der Sloot, and A. van den Bosch. 2002. TiMBL: Tilburg memory based learner, version 4.3, reference guide. ILK technical report, Tilburg University. Available from http://ilk.uvt.nl.

R. Eklund and E. Shriberg. 1998. Crosslinguistic disfluency modeling: A comparative analysis of Swedish and American English human-human and human-machine dialogs. In *Proc. Int. Conf. on Spoken language processing*.

P. Heeman and J. Allen. 1994. Detecting and correcting speech repairs. In *Proc. 32nd Annual Meeting of the Association for Computational Linguistics (ACL-94)*, pages 295–302.

P. Heeman. 1999. Modeling speech repairs and intonational phrasing to improve speech recognition. In *IEEE Workshop on Automatic Speech Recognition and Understanding*.

R. Kohavi and G. John. 1997. Wrappers for feature subset selection. *Artificial Intelligence*, 97(1-2):273–324.

C. Nakatani and J. Hirschberg. 1994. A corpus-based study of repair cues in spontaneous speech. In *JASA*.

N. Oostdijk, 2002. *The Design of the Spoken Dutch Corpus.* In: New Frontiers of Corpus Research. P. Peters, P. Collins and A. Smith (eds.), pages 105-112. Amsterdam: Rodopi.

S. Oviatt. 1995. Predicting spoken disfluencies during human-computer interaction. *Computer Speech Language*, 9:19–36.

F. Provost, D. Jensen, and T. Oates. 1999. Efficient progressive sampling. In *Knowledge Discovery and Data Mining*, pages 23–32.

E. Shriberg, A. Stolcke, and D. Baron. 2001. Can prosody aid the automatic processing of multi-party meetings? Evidence from predicting punctuation, disfluencies, and overlapping speech. In *Proc. ISCA Tutorial and Research Workshop on Prosody in Speech Recognition and Understanding*, pages 139–146.

E. Shriberg. 1994. *Preliminaries to a theory of speech disfluencies.* Ph.D. thesis, University of California at Berkeley.

J. Spilker, A. Batliner, and E. Nöth. 2001. How to Repair Speech Repairs in an End-to-End System. In *Proc. ISCA Workshop on Disfluency in Spontaneous Speech*, pages 73–76.

A. Stolcke, E. Shriberg, R. Bates, M. Ostendorf, D. Hakkani, M. Plauche, G. Tur, and Y. Lu. 1998. Automatic detection of sentence boundaries and disfluencies based on recognized words. In *Proc. Int. Conf. on Spoken Language Processing*, volume 5, pages 2247–2250.

C. van Rijsbergen. 1979. *Information Retrieval.* Buttersworth, London.