Information Extraction from Multi-Document Threads

David Masterson and Nicholas Kushmerick
Dept. of Computer Science, University College Dublin
{david.masterson,nick@ucd.ie}

Abstract

Information extraction (IE) is the task of extracting fragments of important information from natural language documents. Most IE research involves algorithms for learning to exploit regularities inherent in the textual information and language use, and such systems generally assume that each document can be processed in isolation. We are extending IE techniques to multi-document extraction tasks, in which the information to be extracted is distributed across several documents. For example, many kinds of work-flow transactions are realized as sequences of electronic mail messages comprising a conversation among several participants. We show that IE performance can be improved by harnessing the structural and temporal relationships between documents.

1 Introduction

Information Extraction (IE) is an important approach to automated information management. IE is the task of converting documents containing fragments of structured information embedded in other extraneous material, into a structured template or database-like representation. For example, to help a financial analyst identify trends in corporate mergers and acquisitions, an IE system could convert a stream of financial news articles into a series of templates, each of which captures the details of a single transaction as reported in one news article.

The key challenges to effective IE are the inherent complexity and ambiguity of natural language, and the need for IE systems to be rapidly ported to new domains. For example, one would hope that an IE system tuned for the mergers and acquisitions domain could be rapidly reconfigured to extract information about (for example) hostile takeovers. Machine learning has thus emerged has a powerful approach to developing adaptive information extraction systems that rapidly scale to new domains [3, 8, 10, 11, 6, 4]. The general approach is that the IE system learns extraction rules by generalizing from a set of training documents, each manually annotated with the correct information to extract.

As depicted in Figure 1, nearly all existing adaptive IE algorithms make a powerful assumption: each extracted template can be filled by analyzing a single document in isolation. In many cases this assumption is realistic, but we are interested in multi-document extraction tasks in which it is highly unrealistic.

Multi-document extraction tasks arise naturally in many work-flow scenarios, in which some operational process or transaction is realized as a "conversation" between several participants distributed over several natural language texts. We are particularly motivated by electronic mail work-flow streams [2]. For example, in Sec. 3, we evaluate our approach on a corpus of email message threads, each of which discusses a postgraduate student application, and the task is to convert each thread into the structured representation needed to actually register the student with the university authorities.

Multi-document workflow streams are important for two reasons. From an application perspective, our techniques are an enabling technology for a variety of automated logging and monitoring tools for work-flow processes that are realized as docu-

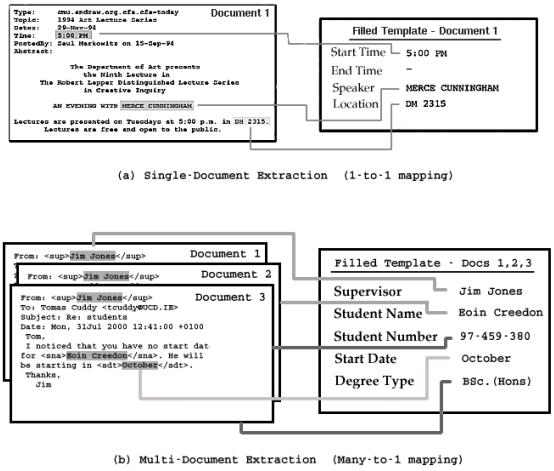

Figure 1: (a) Most IE research assumes a one-to-one correspondence between extracted templates and documents. (b) In multi-document tasks, each template draws content from multiple related documents.

ment sequences. From a research perspective, multi-document extraction is a challenging new direction for which existing techniques are inadequate.

Our approach to multi-document extraction is based on two phases. In a pre-processing phase, the training data is augmented with synthetic documents in an attempt to improve recall. We then invoke an off-the-shelf adaptive IE algorithm to learn a set of extraction rules. During extraction, these rules are invoked on a document thread, and the resulting templates are post-processed using the temporal structure of the message stream to disambiguate extraction decisions and hence improve precision. Our experiments in a a challenging real-world multi-document extraction task demonstrate a 15% improvement over the core learning algorithm.

The remainder of this paper is organized as follows. Section 2 describes our approach to multi-document extraction. Section 3 demonstrates that our approach is effective in a corpus of real email. Section 4 describes related work, and Section 5 summarizes our results and suggests future directions for our research.

2 Multi-document extraction

We begin by describing the multi-document extraction problem in more detail, and then describe our proposed approach to multi-document extraction.

2.1 Problem formulation

As in standard information extraction tasks, we assume as input a template that is to be instantiated from a set of documents. The template comprises several fields, and the goal is to extract particular fragments from the documents to assign to each field. Accuracy is measured in terms of whether the extracted values are correct, compared to a reference template provided by a human annotator.

As shown in Figure 1, traditional single-document extraction techniques assume that the template values should be drawn from a single document. In contrast, in a multi-document setting, the document collection is partitioned into sets of related documents, and a single template is constructed from each such set. In our work-flow scenario, each set corresponds to a single operational process or transaction, and the goal is to summarize the entire transaction in one template. Note that we assume that this partitioning is provided as input. We defer to future work the use of text classification and clustering algorithms to automatically partition a sequence of interleaved messages.

2.2 Approach

Our approach to multi-document extraction involves invoking an existing adaptive IE algorithm as a subroutine. More precisely, an adaptive IE algorithm is a function LEARN that takes as input a set of training documents TRAIN, and outputs a set of extraction rules RULES. Informally, we describe this process using a functional notation: LEARN(TRAIN) $\rightarrow$ RULES. Note that the training documents TRAIN have been manually annotated to indicate the correct fragments.

In our experiments, we used the state-of-the-art $(LP)^2$ algorithm [4] for LEARN, but our technique is applicable to any adaptive IE algorithm. In particular, our approach does not depend on any specific rule language (indicated generically as RULES above). Indeed, our approach is suitable for IE systems that do not learn explicit rules but rather stochastic structures such as hidden Markov models (eg, [6]).

Following the standard methodology, an adaptive IE system is evaluated by invoking a set of learned rules on a disjoint set of test documents, and comparing the extracted templates to the reference templates TEMPS associated with the test data. The following notation indicates this methodology:

$$\text{INV}(\text{LEARN}(\text{TRAIN}),\text{TEST}) \rightarrow \text{TEMPS}$$

As demonstrated in Section 3, a direct application of conventional IE techniques to our multi-document extraction task yields poor performance. Our approach to multi-document extraction is to two-fold. First, we pre-process the training data before it is submitted to the LEARN algorithm. Second, we post-process the fields extracted by INV before comparing them to the reference templates. Informally, we can characterize our multi-document extraction approach as follows:

$$\text{POST}(\text{INV}(\text{LEARN}(\text{PRE}(\text{TRAIN})),\text{TEST})) \rightarrow \text{TEMPS}$$

As described in the Sections 2.3 and 2.4, we have investigated several ways to PREprocess the training data we well as to POSTprocess the extracted content. An IE system can make two kinds of errors: false negatives (resulting in low recall) and false positives (resulting in low precision). Our pre-processing strategies are designed to improve recall, while our post-processing strategies attempt to improve precision.

2.3 Pre-processing the training set

Manually annotating training documents is expensive and error-prone, so a central challenge for adaptive IE is to be able to learn from small amounts of training data. Generally, insufficient training data results in an IE system that suffers from poor recall. This issue is particularly acute in the sort of specialized, sparse multi-document extraction task we studied, in which the small community of users may not generate much training data even after months of email conversations, yet they still expect a robust IE system.

Several researchers have investigated so-called active learning strategies that ask humans to annotate only those documents that will actually lead to improved performance [5, 12, 13]. Our document preprocessing step is based on an alternative approach. Rather than assuming any control over the annotated documents, we have explored ways to augment the existing training data in an attempt to "trick" the learning algorithm to generalize more effectively.

In particular, the goal of our PREprocess strategy is to improve recall by automatically creating additional synthetic training documents from those provided by the human annotator. The expectation is that these additional documents will improve the recall of the learned extraction rules. (Of course, overall performance depends on both precision and recall. In Section 2.4 we discuss how the POSTprocess step will ensure adequate precision.)

As shown in Figure 2, we have explored three distinct PREprocessing strategies. The general idea is to make a "copy" of one of the original training documents, and then modify the copy in one of three ways:

1. **Replace.** Replace field values with alternative values mined from various field-specific Web sources. For example, we harvested a list of people's names from the U.S. Census web site, which were used to replace fields that contain a person's name.

2. **Scramble.** First convert each training document into the set of fragments that are *not* extracted. For example, a seminar announcement email containing a start time, speaker name and location, is converted into four segments: the text before the speaker, the text between the speaker and the start time, the text between the start time and the location, and the text after the location. Create the synthetic document by replacing the original document's inter-field fragments with suitable fragments randomly selected from other training documents.

3. **Replace & Scramble.** Make both of the above modifications.

The synthetic documents created by PREprocess are generally neither semantically meaningful nor grammatically correct. But from the perspective of the learning algorithm, we heuristically assume that they are annotated in the same way that a human would have annotated them if requested, and thus can serve as additional weak training data.

In particular, we expect that this process will increase recall. An important reason for poor recall is that the learned rules over-fit the training data. For example, a rule for identifying a seminar speaker might stipulate that the speaker's name is "John", since this is a common first name. By randomly inserting alternative names ("Henry", "Archibald", ...) we are encouraging the learning algorithm to generalize beyond the superficial cue "John".

2.4 Post-processing extracted fields

Document PREprocessing is designed to improve recall. The later POSTprocessing step is designed to increase precision. The basic issue—like any situation in which recall and precision compete against each other—is that learned rules that extract lots of true positives are also likely to extract false positives.

In single-document extraction tasks, there are no constraints beyond what explicitly available in the document being processed. In contrast, multi-document extraction tasks offer the possibility of disambiguating one document on the basis on the other documents in its thread. As an extreme example, our email corpus (see Section 3) contains multiple copies of the same message, one in each thread to which it is relevant, and different fragments are supposed to be extracted depending on the context (ie, the the thread in which it occurs). Clearly, it is impossible for any single-document IE system to achieve high performance in this case!

Our post-processing strategies revolve is to use knowledge learned about the temporal and/or structural inter-relationships between documents in a thread, in order to prune a large candidate set of extracted fragments to a smaller, more accurate set. By doing so, recall remains relatively high while precision is increased.

In more detail, the POSTprocess step works as follows. The learned rules are first invoked on each of the N documents in a thread, to create our large candidate set, complete with extraction confidance values. We then start to fill the template using the best values from document 1. As we move through the documents in temporal order (2, 3, 4, ..., N), we replace values in the filled template only if the extraction confidence of a new slot fragment exceeds that of a fragment already extracted. Once we reach the the N'th document, we have filled the template in the best possible way using temporal ordering.

This algorithm can be confused by, for example, a single high-confidence (but incorrect) fragment causing POSTprocess to discard several correct occurrences of the correct fragment that were extracted with lower confidence. In Section 5 we discuss our ongoing extensions and generalization of this simple approach to POSTprocessing.

3 Experiments

We have performed several experiments that demonstrate that our techniques are useful in the extraction of information from multi-document threads.

3.1 Postgraduate e-mail corpus

Our experiments are conducted on a collection of email conversations between prospective postgraduate students, their research supervisors, and the postgraduate director, of the Computer Science Department at University College Dublin. These conversations typically involve a student wishing to apply for a postgraduate position, and the director and supervisor collating the information necessary to process the application.

As mentioned in subsection 2.3, we expect datasets from these specialised domains to be especially sparse. Our dataset was obtained by sifting through 4 years of archived email messages, and yet in the end, it consisted of only 107 emails, which make up 48 distinct conversational threads.

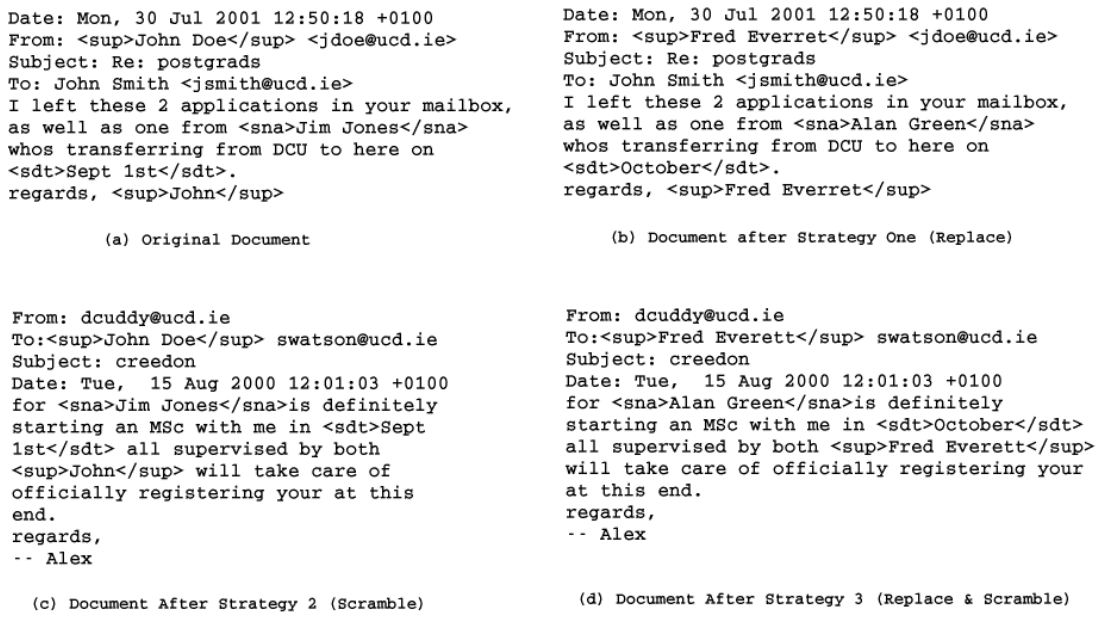

Figure 2: An example document before and after the three PREprocess strategies.

The template is described in detail in Figure 3. Not all values for these slots are guaranteed to be present within a thread of emails. This document corpus is available to the research community; contact the authors for details.

3.2 Baseline performance

Our experiments are based on the $(LP)^2$ adaptive IE algorithm [4]. This algorithm learns rules that incorporate syntactic and contextual information to help locate the probable position of pieces of desired information in a new unseen document. These rules are used to insert tags into new unseen documents. When all the rules have been applied, pieces of text from the new document that are surrounded by correctly paired tags are deemed to be extracted. The algorithm also learns a second set of rules that can alter the position of tags already placed in order to make the encapsulated text better fit the slot it is designated for.

As a baseline, Figure 4 shows the average cross-validated F1 performance of $(LP)^2$ on the email corpus. These data demonstrate that this task is inherently very challenging. Even when trained on most of the available data, F1 is well below 50% for most fields. F1 only exceeds 50% on one slot (`sup`). Unlike the others, this field takes values from a small finite set, and it appears that the learned rules simply memorize these values. Note that $(LP)^2$ is a state-of-the-art adaptive IE algorithm across a wide variety of domains, so we attribute these poor results to the task rather than the learning algorithm.

3.3 Pre-processing experiments

As described in Section 2.3, we have explored several ways to construct additional synthetic training examples from the manually-annotated training documents.

Slot	Meaning	Example
sna	Student name	John, **Amy O'Neill** should also
sno	Student ID number	his ID is **99-459-381**. He got a 1st
sup	Prospective supervisor	From: **John Smith** <john.smith@ucd.ie>
sdt	Expected start date	she should start in **October**. thanks
deg	Primary degree type	he got a **B.sc.** in Physics from
sub	Subject of primary degree	he got a 1st class B.sc. in **Physics** from
grd	Grade achieved in primary degree	BSc, 2002, **2.1** (Computer Science)
ins	Institution awarding primary degree	she finished at **trinity** in 99
dyr	Year of primary degree	BSc, **2002**, 2.1 (Computer Science)

Figure 3: The postgraduate email template comprises nine fields.

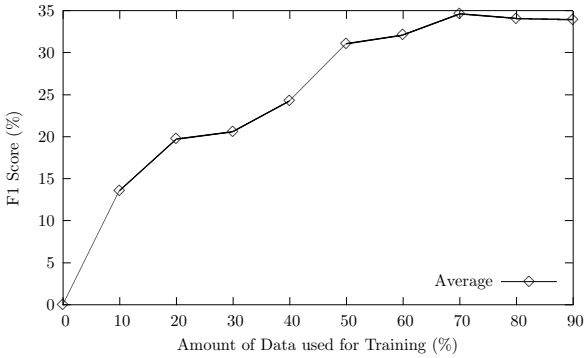

Figure 4: Baseline F1 performance of $(LP)^2$ on the email corpus.

Strategy	Synthetic threads per original				
	0	1	3	5	8
Replace	20.13	23.62	24.98	26.47	24.94
Scramble	20.13	21.45	23.02	24.63	22.51
Both	20.13	22	23.69	-	-

Figure 5: F1 performance for three PREprocessing strategies, as a function of the number of synthetic threads created.

Figure 5 shows the F1 performance as function of the numbers of new threads added to the training set, for each of the three strategies. For example, in "3" column, PREprocess generated three synthetic threads from each original thread.

These data demonstrate that using a modest number of synthetic training documents yields a dramatic improvement in F1. As the ratio between synthetic and original documents increases, performance declines, and eventually deteriorates below the baseline.

Based on these results, we fixed the number of synthetic threads at 5 per original thread, and adopt the **Replace** strategy. Figure 7 shows a learning curve for this configuration as well as the baseline with neither PRE nor POSTprocessing. (We discuss the two POSTprocessing curves in Section 3.4.) We observe in this graph that PREprocess improves F1 by about 15% compared to the baseline.

Recall that the motivation of PREprocess is to increase recall. Figure 7 showed that F1 increases, and Figure 6 confirms that this was partly due to a large increase in recall. Fortunately, this did not come at the cost of decreased precision; indeed precision rose substantially as well.

Finally, we also tried our PREprocess techniques on Freitag's seminar announcements dataset [7]. The results were quite different: performance deteriorated slightly (though less than 5%) rather than improved. We defer to future work an exploration of the conditions under which our approach is suitable.

3.4 Post-processing experiments

Returning to Figure 7, we also show the results of the POSTprocess algorithm describe in Section 2.4. As

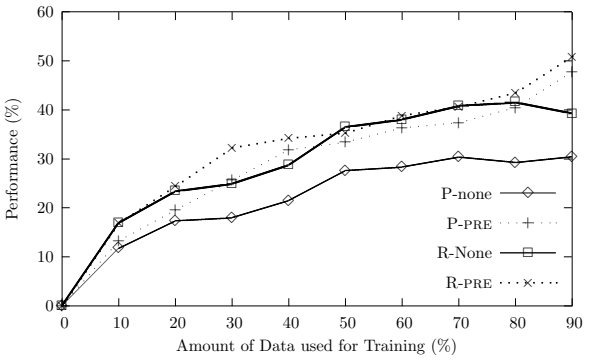

Figure 6: Effect of using PREprocessing on precision and recall.

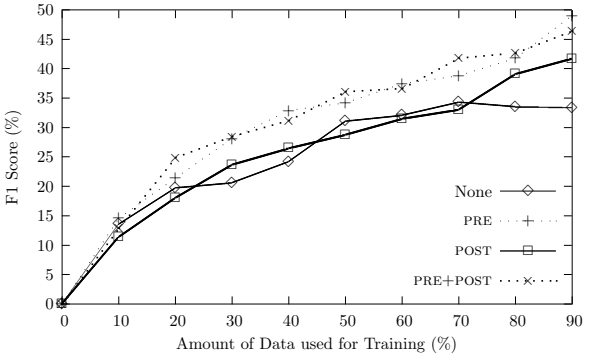

Figure 7: Performance with just PREprocessing, just POSTprocessing, both PRE and POSTprocessing, and neither.

with PREprocess, we see a substantial improvement over the baseline of approximately 8%. If we combine PRE and POSTprocessing, we see an improvement of about 13%.

It would appear that both PREprocess and POSTprocess contribute to improved performance, but we note that the performance gains are not cumulative. In Section 5, we describe improved POSTprocessing strategies that we hope yield further improvements.

4 Related Work

While there is much work in the field of information extraction, there is virtually no work on multi-document extraction. There has been considerable effort on the task of discourse processing, which involve recognizing discourse acts in speech segments. Discourse processing has been applied to IE [9], but was found to give disappointing results on the MUC-6 Business News task. However, Kehler's experiments involved single-document rather than multi-document extraction, so no document interrelationships were assumed or used.

Another related sub-field of discourse processing is the the problem of dialogue act modeling [15], the analysis of conversational structure by identifying and labeling dialogue acts (such as "question" or "statement"). Dialogue act modeling is similar in spirit to our problem, in that we could view a document thread as a series of dialogue acts, and then try to identify the various pieces of information within them using the labeling of the dialogue acts as temporal and structural clues.

5 Discussion and Future Work

Information extraction is a powerful approach knowledge management, but existing techniques emphasize single-document extraction tasks in which templates do not span across documents. However, many workflow scenarios (such as intelligent email management) involve information distributed across several documents, so novel approaches to multi-document extraction are required.

We described a two-phase approach to multi-document extraction. In a pre-processing phase, the training data is augmented with synthetic documents in an attempt to improve recall. In a post-processing phase, the temporal relationships between documents are exploited to disambiguate extraction decisions. Our approach resulted in a 15% improvement in a challenging real-world multi-document extraction task.

Our current work is focused on enhancing our POSTprocessing strategies. In the extraction results,

we often encounter multiple results being extracted for a single field (i.e. "John Smith" and "Dr John Smith" being extracted as candidates for a supervisors name). We are experimenting with the use of string similarity metrics, such as edit distances, to merge fragments that are deemed to be sufficiently similar before pruning of the candidate set begins, and in doing so raising their confidance values. This should help to remove ambiguous results as well as boosting the score of fragments that are extracted multiple times. However, this technique has so far yielded mixed results.

A more sophisticated strategy that we intend to investigate would be to learn temporal regularities from document threads, in order to disambiguate between several possible extraction values. For example, if document number n in a thread contains a request for a student number ("Can you please tell me your student number?"), then document number $n + 1$ is likely to contain a student number. This could be accomplished by mining the thread tag sequences for frequent temporal patterns [1], and then adjusting the confidence of extracted values to reflect this additional inter-document evidence.

Acknowledgments. This research was supported by grants SFI/01/F.1/C015 from Science Foundation Ireland, and N00014-03-1-0274 from the US Office of Naval Research. We thank Fabio Ciravegna for access to $(LP)^2$.

References

[1] R. Agrawal and R. Srikant: Mining Sequential Patterns. *Proc. 11th International Conf. on Data Engineering* (1995) 3–14

[2] J. Cadiz and L. Dabbish and A. Gupta and G. Venolia: Supporting email workflow. *Microsoft Research Technical Report*, (2001) MSR-TR-2001-88.

[3] M. E. Califf and R. J. Mooney: Relational Learning of Pattern-Match Rules for Information Extraction. *Working Notes of AAAI Spring Symposium on Applying Machine Learning to Discourse Processing* (1998) 6–11

[4] F. Ciravegna: $(LP)^2$, an Adaptive Algorithm for Information Extraction from Web-related Texts. *Proc. IJCAI-01 Workshop on Adaptive Text Extraction and Mining* (2001)

[5] F. Ciravegna et al. User-System Cooperation in Document Annotation based on Information Extraction *Proc. 13th International Conference on Knowledge Engineering and Knowledge Management (EKAW02)* (2002)

[6] D. Freitag and A. McCallum: Information Extraction with HMMs and Shrinkage. *Proc. AAAI-99 Workshop on Machine Learning for Information Extraction* (1999)

[7] D. Freitag: Machine learning for information extraction in informal domains. *Ph.D. Dissertation*, Carnegie Mellon University (1998)

[8] D. Freitag and N. Kushmerick: Boosted Wrapper Induction. *Proc. 17th National Conference on Artificial Intelligence* (2000) 577–583

[9] A. Kehler: Learning Embedded Discourse Mechanisms for Information Extraction. *Proc. AAAI Spring Symposium on Applying Machine Learning to Discourse Processing.* (1998)

[10] N. Kushmerick: Wrapper induction: Efficiency and Expressiveness *Artificial Intelligence* (2000) **118**(1–2):15–68.

[11] I. Muslea and S. Minton and C. Knoblock: Hierarchical Wrapper Induction for Semistructured Information Sources. *Autonomous Agents and Multi-Agent Systems* **4 1/2** (2001) 93–114

[12] I. Muslea and S. Minton and C. Knoblock: Active learning with strong and weak views: A case study on wrapper induction *Proc. 18th International Joint Conference on Artificial Intelligence* (2003)

[13] T. Scheffer and S. Wrobel Active Learning of Partially Hidden Markov Models *Proc. ECML/PKDD Workshop on Instance Selection* (2001)

[14] S. Soderland: Learning Text Analysis Rules for Domain-specific Natural Language Processing. *PhD thesis, University of Massachusetts* (1996)

[15] A. Stolcke et al. : Dialogue Act Modelling for Automatic Tagging and Recognition of Conversational Speech. *Computational Linguistics* **26(3)** 339–373 (2000)