Workshop Program

9:30 Welcome

10:00 Invited Talk: Creation and Management of Accurately Annotated Corpora
Yuji Matsumoto

11:00 Coffee break

11:30 Automatic Phonetic Transcription of Large Speech Corpora
Christophe Van Bael, Lou Boves, Henk van den Heuvel, Helmer Strik

12:15 Annotation of Grammatemes in the Prague Dependency Treebank 2.0
Magda Razímová, Zdenk Zabokrtsky

13:00 Lunch break

14:30 Invited Talk: Tagset Design for High Accuracy POS Tagging and Automatically Building Mapping between Arbitrary Tagsets
Dan Tufis

15:30 Constraint-Based Extract Alignment for Black-Box Evaluation of Extractive Summarization Methods
Jorge Marques Pelizzoni, Thiago Ianez Carbone, Lucia Helena Machado Rino

16:30 Coffee break

17:00 Semi-Automatic Phonological Annotations of Speech by Grammatical Inference
Robert Kelly and Julie Carson-Berndsen

17:45 Invited Talk: Toward a 'Science' of Annotation: Experiences from OntoNotes
Eduard Hovy

18:15 Panel Discussion

Workshop Organisers

Eric Atwell, University of Leeds, UK
Nancy Ide, Vassar College, USA

Program Committee

Eric Atwell, University of Leeds (UK)
Nigel Collier, National Institute of Informatics (Japan)
Atsushi Fujii, University of Tsukuba (Japan)
Rebecca Hwa, University of Pittsburgh (USA)
Nancy Ide, Vassar College (USA)
David Lewis, David D. Lewis Consulting USA)
Miles Osborne, University of Edinburgh (UK)
Anoop Sarkar, Simon Fraser University (CA)
Mark Steedman, University of Edinburgh (UK)
Takenobu Tokunaga, Tokyo Institute of Technology (Japan)
Kiyotaka Uchimoto, National Institute of Information and Communications Technology (Japan)

Table of Contents

Creation and Management of Accurately Annotated Corpora

Yuji Matsumoto

Graduate School of Information Science
Nara Institute of Science and Tachnology
8916-5 Takayama, Ikoma, Nara 630-0192 Japan
matsu@is.naist.jp

Recent progress in natural language processing has achieved implementation of highly accurate language processing tools such as Part-of-speech tagging, phrase and named entity chunking, and syntactic parsing, which are now used in various natural language processing application. Large scale annotated corpora are very important not only for linguistic research but also for improving accuracy of practical language processing tools, since most of the state-of-art practical NLP tools are machine learning-based systems which require large-scale and accurately annotated corpora. On the other hand, it is well-known that even a widely used annotated corpus such as Penn Treebank still includes a number of annotation errors. To realize highly accurate annotated corpora, it is indispensable to have a supporting environment for maintaining annotated corpora and for finding and correcting errors hauting in automatically or manually annotated corpora.

This talk presents our activities of machine learning-based development of natural language processing tools and our language resource management systems.

In the last decade, machine learning and corpus-based natural language processing methods have made remarkable progress, and we are now aware of good methodologies for developing pretty accurate NLP tools. Corpus-based NLP systems have an advantage over handcrafted systems in that they are easily adaptable to new domains as well as new languages, provided an annotated corpus is available. Compared with the ramarkable pregress in machine learning methodologies, deployment of annotated corpora in various domains and languages is very slow. Even for the most studied English language, we have very limited number of linguistically annotated corpora. While Penn Treebank is the most widely used annotated English corpus, its size and domain are quite limited and the improvement is very slow. Therefore, it is quite important not only to develop sophisticated corpus-based natural language processing tools but also to develop an environment or methodologies to create and manage accurately annotated language resources.

This talk will cover two main topics: The first is our activities on development of Natural Language Processing tools and resources, and the second is about the systems for maintaining natural language resources.

In the first part, I will introduce our activities on the development of machine learning-based NLP tools, such as segmentation and POS taggers, a general purpose chunker and its application to base phrase, Named Entity and unknown word chunking, and dependency parsers. We have been developing those systems for Japanese, Chinese and English. I will also introduce and compare various machine learning techniques used in the systems. I will also briefly talk about our recent application of machine learning technique to anaphora resolution. In parallel with those tools, we have been developing lexicons of three languages with the help of unknown word identification program. I will mention this activity as well.

In the second half of the talk, I will introduce two systems for natural language resource management: ChaKi and Cradle.

ChaKi is an annotated corpus management system. It currently deals with POS-tagged and dependency analyzed corpora. The main characteristics of the system are summarized as follows: (1) Coordination between annotated corpora and lexicon: The words in annotated corpora are represented as pointers to entries in the lexicon. This helps to keep consistency between the corpora and the lexicon. (2) Variation of annotation: POS tags, base phrase chunks and syntactic dependency structures (dependency between words or chunks) are handled. Multi-word expressions can be defined in the lexicon together with their constituents. (3) Search: Three modes of search are possible, string search, word search, and dependency structure search. (4) Browsing: KWIC presentation of retrieved sentences and a dependency structure tree browser are provided. (5) Error correction: Segmentation, word information, and dependency structure errors correction is supported on the system. (6) Statistic calculation: Basic statistic calculations such as word frequencies and collocation counts in fixed size windows are provided. (7) Multilinguality: The system is designed as language independent, and now handles Japanese, Chinese and English corpora.

Cradle is a lexicon management system. It currently keeps a lexicon for Part-of-speech tagging and maintain lexical information such as POS lables, information on inflection types, and constitutent structure information for a word if it is a compound or a multi-word expression.

Finally, I will talk about our future plan and use of those systems for development of large scale corpus and lexicon.

Tagset Design for High Accuracy POS Tagging and Automatically Building Mapping between Arbitrary Tagsets

Dan Tufis

Research Institute for Artificial Intelligence
Romanian Academy
13, Calea 13 Septembrie, 050711, Bucharest
tufis@racai.ro

The paper discusses various issues related to tagset design for maximizing the accuracy of subsequent tagging processes. It will review the tiered tagging design approach as well as some new developments. Tiered tagging is based on a methodology that tries to compromise the natural users' desire to have their texts tagged with as much informative tags as possible (therefore very large tagsets) with the requirement of having enough annotated data needed to build a reliable statistical language model. The methodology involves the use of a reduced hidden corpus tagset, automatically constructed from the large targeted lexical tagset, and a procedure to map back the reduced tagset into the large one in the final annotated text. The two tagsets (the lexical and corpus tagsets) are related by a subsumption relation.

When the reduction of the cardinality of the large tagset is information lossless (redundancy elimination) the mapping from the reduced tagset to the initial one is deterministic and is simply ensured by a lookup of a dictionary.

For tagset reduction with information loss, in spite of a much significant reduction of the tagsets, the recovering of the left out morpho-lexical information, although to a large extent deterministic, requires additional preprocessing to solve some non-deterministic cases. In the previous version of the tiered tagging approach we used several hand-crafted rules (regular expressions defined over the reduced tagset, with a span of ±4 tags around the ambiguously mapped tags). Recently we developed a new version of the tiered tagging where the need for the expert introspection in writing the rules was eliminated by a fully automatic procedure. The mapping is ensured by a supervised ME algorithm.

Another issue we will discuss refers to automatic construction of mappings for two completely unrelated tagsets so that one could turn the POS annotation of one text from one tagset into another tagset. While in the tiered tagging the mapping from the reduced tagset to the large tagset was inherently 1-n, here the mapping may be (and usually is) of the n-m type. There are several ways to solve this problem. We will analyze in some details our solution, which is called *cross-tagging*. Essentially, from two different texts tagged with two unrelated tagsets, be they GS-A(Tagset1) and GS-B(Tagset2) the cross-tagging procedure, builds a statistical mix language model MLM (Tagset1, Tagset2). This language model is used by a statistical tagger (in our case a second order HMM, but any tagger would do) to retag GS-A and GS-B both with their initial tagsets and with the other tagsets. The result of the cross-tagging is many-folded interesting to look at:

- The mapping between Tagset1 and Tagset2 is a very useful resource, the analysis of which may highlight the adequacy of morpho-syntactic distinctions made by one tagset or another; the mapping is a list of equivalence probabitilies: $\{P(T_i^{Tagset1}|T_j^{Tagset2}),\ P(T_i^{Tagset2}|T_j^{Tagset1})\}$.
- The cross-tagging annotations GS-A(Tagset2) and GS-B(Tagset1) are more accurate than the annotations obtained by direct tagging (the usual approach)
- By double-cross-tagging of GS-A and GS-B, GS-A*(Tagset1) and GS-B*(Tagset2), one can clearly identify (and correct) errors in the initial annotation of the reference texts.

We performed experiments on two English texts annotated with the Multext-East tagset and Penn-tagset respectively. The reference texts were "1984" and "Semcor". By cross-tagging, we produced new annotations: 1984(Penn) and Semcor(Multext-East). By double-cross-tagging we obtained improved versions of the initial tagging: 1984^{++}(Multext-East) and Semcor^{++}(Penn). For instance between the initial Semcor(Penn) and Semcor^{++}(Penn) there were more than 80,000 differences, out of which the vast majority represented tagging errors in Semcor(Penn). The cross-tagged and double-cross-tagged corpora are public.

Toward a 'Science' of Annotation: Experiences from OntoNotes

Eduard Hovy

Information Sciences Institute
Marina Del Rey, California
hovy@isi.edu

As machine learning algorithms and their application for NLP become better understood, attention turns toward the production of annotated corpora to which they can be applied. Numerous phenomena present themselves for annotation, including aspects in lexical semantics, discourse, pragmatics, and dialogue. But several questions immediately must be answered:

1. How does one obtain a balanced corpus to annotate? What *is* a balanced corpus?
2. How does one decide which aspects to annotate? How does one adequately express the theory behind the phenomena in simple annotation steps?
3. Which annotators does one hire? How does one ensure that they are adequately trained?
4. How does one establish a simple, fast, and trustworthy annotation procedure? What interfaces does one build? How does one ensure that the interfaces do not affect the annotation results?
5. How does evaluate the results? What are the appropriate agreement measures? At which cutoff points should one re-do the annotations? How does one ensure improvement?
6. How should one formulate and store the results? How does one ensure compatibility with other existing resources? How does one make results available for best impact?
7. How does one report the annotation effort and results? How does one actually get a paper on this work published at an important conference? What should the paper contain?

Despite their being so basic, there is almost no established procedure or standard set of answers to these questions today. In this talk I discuss some of these aspects, pointing to the lessons learned in the ongoing OntoNotes project (joint with BBN, the University of Colorado (PropBank), the University of Pennsylvania (Treebank), and ISI).

Automatic Phonetic Transcription of Large Speech Corpora

Christophe Van Bael, Lou Boves, Henk van den Heuvel, Helmer Strik

Centre for Language and Speech Technology (CLST)
Radboud University Nijmegen, the Netherlands
[c.v.bael,l.boves,h.v.d.heuvel,w.strik]@let.ru.nl

Abstract

This study is aimed at investigating whether automatic phonetic transcription procedures can approximate manual transcriptions typically delivered with contemporary large speech corpora. To this end, ten automatic procedures were used to generate a broad phonetic transcription of well-prepared speech (read-aloud texts) and spontaneous speech (telephone dialogues) from the Spoken Dutch Corpus. The resulting transcriptions were compared to manually verified phonetic transcriptions from the same corpus.

Most transcription procedures were based on lexical pronunciation variation modelling. The use of signal-based pronunciation variants prevented the approximation of the manually verified phonetic transcriptions. The use of knowledge-based pronunciation variants did not give optimal results either. A canonical transcription that, through the use of decision trees and a small sample of manually verified phonetic transcriptions, was modelled towards the target transcription, performed best. The number and the nature of the remaining disagreements with the reference transcriptions compared to inter-labeller disagreements reported in the literature.

1. Introduction

In the last decades we have witnessed the development of large multi-purpose speech corpora such as TIMIT (1990), Switchboard (Godfrey et al., 1992), Verbmobil (Hess et al., 1995), the Spoken Dutch Corpus (Oostdijk, 2002) and the Corpus of Spontaneous Japanese (Maekawa, 2003). In particular a good phonetic transcription increases the value of such corpora for scientific research and for the development of applications such as automatic speech recognition (ASR).

For some purposes (e.g. basic ASR development), a canonical phonetic representation of speech can be sufficient (Van Bael et al., 2006). However, for other purposes, such as linguistic research, a more accurate annotation of the signal is needed. For this reason, some corpora come with a manual transcription of the data (Hess et al., 1995; Greenberg et al., 1996; Oostdijk, 2002).

Despite efforts to improve the workflow of human experts, however, the human transcription process remains tedious and expensive (Cucchiarini, 1993). This explains why 'only' 4 hours of Switchboard speech were phonetically transcribed as an afterthought, and why the phonetic transcription of 'only' 1 million words of the 9-million-word Spoken Dutch Corpus was manually verified. Both for Switchboard and the Spoken Dutch Corpus, transcription costs were restricted by presenting trained students with an example transcription. The students were asked to verify this transcription rather than to transcribe from scratch (Greenberg et al. 1996; Goddijn & Binnenpoorte, 2003). Although such a check-and-correct procedure is very attractive in terms of cost reduction, it has been suggested that it may bias the resulting transcriptions towards the example transcription (Binnenpoorte, 2006). In addition, the costs involved in such a procedure are still quite substantial. Demuynck et al. (2002) reported that the manual verification process took 15 minutes for one minute of speech recorded in formal lectures and 40 minutes for one minute of spontaneous speech.

Several studies already reported the benefits of automatic phonetic transcriptions for ASR (e.g. Riley, 1999; Yang & Martens, 2000; Wester, 2003; Saraçlar & Khundanpur, 2004; Tjalve & Huckvale, 2005) and for speech synthesis (e.g. Bellegarda, 2005; Jande, 2005, Wang et al. 2005). In these studies, the phonetic transcriptions were used as tools to improve the performance of a specific system. Hence, they were not evaluated in terms of their similarity with manually verified broad phonetic transcriptions. Only a small number of studies evaluated automatic phonetic transcriptions in terms of their resemblance to manual transcriptions (e.g. Wesenick, & Kipp, 1996; Kipp, et al. 1997; Demuynck et al. 2004). These studies, however, reported the use and evaluation of only one or a limited number of similar procedures at a time. To our knowledge, no study has compared the performance of established automatic transcription procedures in terms of their ability to approximate manual transcriptions. We are also not aware of attempts to study the potential synergy of the combinatory use of existing transcription procedures.

The aim of this paper is to compare the performance of existing transcription procedures and to investigate whether combinations of these procedures lead to a better performance so that it will eventually be possible to minimise (or even eliminate) human labour in the phonetic transcription of large speech corpora, without reducing the quality of the transcriptions. Since transcriptions in large speech corpora are often designed to suit multiple purposes, our transcriptions are also intended to be multi-applicable rather than particularly suitable for one specific application such as ASR. Therefore, we will evaluate the transcriptions in terms of their similarity to a reference transcription, rather than in terms of a particular speech application. Because we want to approximate manually verified transcriptions, we will also discuss the characteristics of manual phonetic transcriptions obtained through verification of example transcriptions. Most of the procedures discussed in this article require a continuous speech recogniser to select the best fitting lexical pronunciation variant. The major difference between these procedures is the manner in which the lexical pronunciation variants were generated.

In order to ensure the applicability of the transcription procedure in situations where only limited resources are available, all procedures are designed to minimise human effort. Most procedures are based on the use of a standard continuous speech recogniser, an algorithm to align phonetic transcriptions, an orthographically transcribed

corpus, a lexicon with a canonical transcription of all words, and a manually verified transcription of a relatively small sample of the corpus. The manual transcriptions are required to tune the automatic transcription procedures and to evaluate their performance. Some procedures also require a list of phonological processes describing pronunciation variation in the language at hand. Human intervention and labour, if required at all, is limited to the compilation of such a list of phonological processes.

This paper is organised as follows. In Section 2, we introduce the corpus material used in our study. Section 3 sketches the various transcription procedures. Section 4 presents the validation of the corresponding transcriptions. In Section 5 the results are discussed, and in Section 6 general conclusions are formulated.

2. Material

2.1. Speech Material

The speech material was extracted from the Northern Dutch part of the Spoken Dutch Corpus (Oostdijk, 2002). In order not to restrict our study to one particular speech style, we selected read speech (RS) as well as spontaneous telephone dialogues (TD).

The RS was recorded at 16kHz with high-quality table-top microphones for the compilation of a library for the blind. The TD, comprising much more spontaneous speech, were recorded at 8kHz through a telephone platform. As part of the orthographic transcription process all speech material was manually segmented into chunks of approximately 3 seconds. The transcribers were instructed to put chunk boundaries in naturally occurring pauses; only if speech stretched for substantially longer than 3 seconds they had to put chunk boundaries between two words with minimal cross-word co-articulation. The experiments in this study have taken chunks as basic fragments. In order to be able to focus on phonetic transcription proper, we excluded speech chunks that, according to the orthographic transcription, contained salient non-speech sounds, broken words, unintelligible speech, overlapping and foreign speech.

The statistics of the data are presented in Table 1. The data from each speech style were divided into a training set, a development set, and an evaluation set. All data sets were mutually exclusive but they comprised similar material.

		Transcription sets		
Speech style		Training	Development	Evaluation
RS	# words	532,451	7,940	7,940
	hh:mm:ss	44:55:59	0:40:10	0:41:39
TD	# words	263,501	6,953	6,955
	hh:mm:ss	18:20:05	0:30:02	0:29:50

Table 1: Statistics of the phonetic transcriptions.

2.2. Canonical Lexicon

We used a comprehensive multi-purpose in-house lexicon that was compiled by merging various existing electronic lexical resources. The pronunciation forms in this lexicon reflected the pronunciation of words as carefully pronounced in isolation according to the obligatory word-internal phonological processes of Dutch (Booij, 1999). Each lexical entry was represented by just one standard broad phonetic transcription. Information about syllabification and syllabic stress was ignored in order to ensure the applicability of the transcription procedures to languages lacking a lexicon with such specific linguistic information.

2.3. Reference Transcription (RT)

Since we aimed at approximating the manually verified phonetic transcriptions of the Spoken Dutch Corpus, we used these transcriptions as Reference Transcriptions (RT) to tune (development set) and evaluate (evaluation set) our transcription procedures. The RTs were generated in three steps. First, a canonical transcription was generated through a lexicon-lookup procedure in a canonical lexicon. Subsequently, two phonological processes of Dutch, voice assimilation and degemination, were applied to the phones at word boundaries. This was justified by previous research indicating that these processes apply on more than 87% of the word boundaries where they can actually apply (Binnenpoorte & Cucchiarini, 2003). The enhanced transcriptions were verified and corrected by trained students. The transcribers acted according to a strict protocol instructing them to change the canonical example transcription only if they were certain that the example transcription did not correspond to the speech signal. The use of an example transcription resulted in reasonably consistent phonetic transcriptions, but the constraints imposed on the human transcribers also implied the risk of biasing the resulting transcriptions towards the canonical example transcription (Binnenpoorte, 2006).

2.4. Continuous Speech Recogniser (CSR)

Except for the canonical transcriptions, all automatic phonetic transcriptions (APTs) were generated by means of a continuous speech recogniser (CSR) based on Hidden Markov Models and implemented with the HTK Toolkit (Young et al., 2001). Our CSR used 39 gender- and context independent, but speech style-specific acoustic models with 128 Gaussian mixture components per state (37 phone models, 1 model for silences of 30 ms or more and 1 model for the optional silence between words).

The acoustic models were trained in three stages using the CAN-PTs (cf. 3.1.1.1) of the training data. First, flat start acoustic models with 32 Gaussian mixture components were trained through 41 iterative alignments. Subsequently, these models were used to obtain more realistic segmentations of the speech material. These segmentations were then used to bootstrap a new set of acoustic models, which were retrained (through 55 iterations) to acoustic models with 128 Gaussian mixture components per state.

2.5. Algorithm for Dynamic Alignment of Phonetic Transcriptions (ADAPT)

ADAPT (Elffers et al., 2005) is a dynamic programming algorithm designed to align strings of phonetic symbols according to the articulatory distance between the individual symbols. In this study, ADAPT was used to align phonetic transcriptions for the generation of lexical pronunciation variants, and to assess the quality of the automatic phonetic transcriptions through their alignment with a reference transcription.

3. Methodology

In Section 3.1, we introduce ten automatic transcription procedures to generate low-cost APTs. Section 3.2 describes the evaluation procedure with which the APTs and, consequently, the procedures were assessed.

3.1. Generation of phonetic transcriptions with different transcription procedures

Figure 1 shows ten APTs. The procedures from which they result can be divided into two categories: two procedures that did not rely on the use of a lexicon with multiple pronunciation variants per word, and eight procedures that did rely on the use of a multiple pronunciation lexicon in combination with a CSR. The latter procedures can be further categorised according to the way the pronunciation variants were generated. These variants were either based on knowledge from the literature, they were obtained by combining canonical, data-driven and knowledge-based transcriptions, or they were generated with decision trees trained on the alignment of the APTs and the RT of the development data. Most of the procedures required several parameters to be tuned to better approximate the RT of the development data. The optimal parameter settings were subsequently applied for the transcription of the data in the evaluation set.

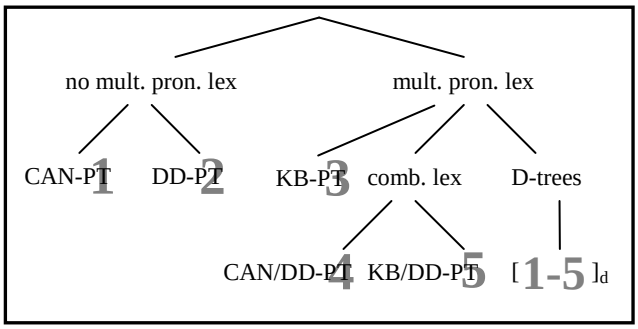

Figure 1: 10 different automatic phonetic transcriptions.

3.1.1. Transcription procedures without a multiple pronunciation lexicon

3.1.1.1. Canonical transcription (CAN-PT)

The canonical transcriptions (CAN-PTs) were generated through a lexicon look-up procedure. Cross-word assimilation and degemination were not modelled. Canonical transcriptions are easy to obtain, since many corpora feature an orthographic transcription and a canonical lexicon of the words in the corpus.

3.1.1.2. Data-driven transcription (DD-PT)

The data-driven transcriptions (DD-PTs) were based on the acoustic *data*. The DD-PTs were generated through constrained phone recognition; a CSR segmented and labelled the speech signal using its acoustic models and a 4-gram phonotactic model trained with the reference transcriptions of the development data in order to approximate human transcription behaviour. Transcription experiments with the data in the development set indicated that for both speech styles 4-gram models outperformed 2-gram, 3-gram, 5-gram and 6-gram models.

3.1.2. Transcription procedures with a multiple pronunciation lexicon

The transcription procedures described in this section differ in the way pronunciation variants were generated. The variants were always listed in speech style-specific multiple pronunciation lexicons. For every word, the best matching variant was selected through the use of a CSR that chose the best matching pronunciation variant from the lexicon given the orthography, the acoustic signal and a set of acoustic models. The development set was used to optimise various parameters in the individual procedures in order to optimise the selection of the lexical pronunciation variants of the words in the evaluation set.

3.1.2.1. Knowledge-based transcription (KB-PT)

In particular ASR research often draws on the literature for the extraction of linguistic knowledge with which lexical pronunciation variants can be generated (Kessens et al., 1999; Strik, 2001). We generated so-called knowledge-based transcriptions (KB-PTs) in three steps.

First, a list of 20 prominent phonological processes was compiled from the linguistic literature on the phonology of Dutch (Booij, 1999). These processes were implemented as context-dependent rewrite rules modelling both within-word and cross-word contexts in which phones from a CAN-PT can be deleted, inserted or substituted with another phone. Most of the processes identified by Booij (1999) can be described in terms of phonetic symbols or articulatory features. However, some of the processes can only be described with information about the prosodic or syllabic structure of words. Most of these processes were reformulated in terms of phonetic symbols and features, since we wanted to exclude non-segmental information (see Section 2.2). The rules were implemented conservatively to minimise the risk of over-generation. The resulting rule set comprised some rules specific for particular words in Dutch, and general phonological rules describing progressive and regressive voice assimilation, nasal assimilation, syllable-final devoicing of obstruents, t-deletion, n-deletion, r-deletion, schwa deletion, schwa epenthesis, palatalisation and degemination. The reduction and the deletion of full vowels, two prominent processes in Dutch, could not be easily formulated without the explicit use of syllabic and prosodic information.

In the second step, the phonological rewrite rules were ordered and used to generate optional pronunciation variants from the CAN-PTs of the speech chunks. The rules applied to the chunks rather than to the words in isolation to account for cross-word phenomena. The rules only applied once, and their order of application was manually optimised. Informal analysis of the resulting pronunciation variants suggested that few - if any - implausible variants were generated, and that no obvious variants were missing. It may well be, however, that two-level rules (Koskenniemi, 1983) or an iterative application of the rewrite rules is needed for the transcription of other languages.

In the third step of the procedure, chunk-level pronunciation variants were listed. Since the literature did not provide numeric information on the frequency of phonological processes, the pronunciation variants did not have prior probabilities. The optimal knowledge-based transcription (KB-PT) was identified through forced recognition.

3.1.2.2. Combined transcriptions (CAN/DD-PT, KB/DD-PT)

After having generated the CAN-PTs, DD-PTs and KB-PTs, these transcriptions were combined to obtain new transcriptions. This time lexical pronunciation variants were generated through the alignment of two APTs at a time. Since the KB-PTs were based on the CAN-PTs, we only combined the CAN-PT with the DD-PT (CAN/DD-PT) and the KB-PT with the DD-PT (KB/DD-PT). Figure 2 illustrates how different pronunciation variants were generated through the alignment of the phones in the CAN-PT and the DD-PT.

CAN-PT: d @	A p @ l t a r t
	+
DD-PT: d -	A b @ l t a - t

Multiple pronunciation variants in CAN/DD-PT :

d @	A p @ l t a r t
d	A p @ l t a r t
d @	A b @ l t a r t
d	A b @ l t a r t
d @	A p @ l t a t
d	A p @ l t a t
d @	A b @ l t a t
d	A b @ l t a t

Figure 2: Generation of pronunciation variants through the alignment of two phonetic transcriptions.

The combination of APTs emerging from different transcription procedures was aimed at providing our CSR with additional linguistically plausible pronunciation variants for the words in the orthography. After all, canonical transcriptions do not model pronunciation variation, and our KB transcriptions only modelled the pronunciation variation that was manually implemented in the form of phonological rewrite rules. The DD-PTs, however, were based directly on the speech signal. Therefore, they had the potential of better representing the actual speech signal, at the risk of being linguistically less plausible than CAN-PTs or KB-PTs. It was reasonable to expect that the combination of the different transcription procedures would alleviate the disadvantages and reinforce the advantages of the individual procedures.

3.1.2.3. Phonetic transcription with decision trees

The use of DD transcription procedures can result in too many, too few or very unlikely lexical pronunciation variants (Wester, 2003). In ASR research, the use of decision trees defining plausible alternatives for a phone given its context phones has often reduced the number of unlikely pronunciation variants and optimised the number of plausible pronunciation variants in recognition lexicons (Riley, 1999; Wester, 2003). We generated decision trees with the C4.5 algorithm (Quinlan, 1993), provided with the Weka package (Witten & Frank, 2005). The procedure pursued to successively improve the CAN-PTs, DD-PTs, KB-PTs, CAN/DD-PTs and KB/DD-PTs comprised four steps.

First, the APT (each of the aforementioned transcriptions consecutively) and the RT of the development data were aligned. Second, all the phones and their context phones in the APT were enumerated. The size of these "phonetic windows" was limited to three phones: the core phone, one preceding and one succeeding phone. The correspondences of the phones in the APT and the RT and the frequencies of these correspondences were used to estimate:

$$P(RT\_phone|APT\_phone,APT\_context\_phones) \quad (1)$$

i.e. the probability of a phone in the reference transcription given a particular phonetic window in the APT. In the third step of the procedure, the resulting decision trees were used to generate likely pronunciation variants for the APT of the unseen evaluation data. The decision trees were now used to predict:

$$P(pron\_variants|APT\_phone,APT\_context\_phones) \quad (2)$$

i.e. the probability of a phone with optional pronunciation variants given a particular phonetic window in the APT. All pronunciation variants with a probability lower than 0.1 were ignored in order to reduce the number of pronunciation variants and, more importantly, to prune unlikely pronunciation variants originating from idiosyncrasies in the original APT.

In the fourth and final step of the procedure, the pronunciation variants were listed in a multiple pronunciation lexicon. The probabilities of the variants were normalised so that the probabilities of all variants of a word added up to 1. Finally, our CSR selected the most likely pronunciation variant for every word in the orthography. The consecutive application of decision tree expansion to the CAN-PTs, DD-PTs, KB-PTs, CAN/DD-PTs and KB/DD-PTs resulted in five new transcriptions hereafter referred to as $[CAN\text{-}PT]_d$, $[DD\text{-}PT]_d$, $[KB\text{-}PT]_d$, $[CAN/DD\text{-}PT]_d$ and $[KB/DD\text{-}PT]_d$.

3.2. Evaluation of the phonetic transcriptions and the transcription procedures

The APTs of the data in the evaluation sets were evaluated in terms of their deviations from the human RT. The comparison was conducted with ADAPT (Elffers et al., 2005). The disagreement metric was formalised as:

$$\%disagreement = \left(\frac{Sub + Del + Ins}{N} \right) * 100 \quad (3)$$

i.e. the sum of all phone substitutions (S*ub*), deletions (*Del*) and insertions (*Ins*) divided by the total number of phones in the reference transcription (*N*). A smaller deviation from the reference transcription indicated a 'better' transcription. A detailed analysis of the number and the nature of the deviations allowed us to systematically investigate the magnitude and the nature of the improvements and deteriorations triggered by the use of the different transcription procedures.

4. Results

The figures in Table 2 describe the disagreements between the APTs and the RTs of the evaluation data. From top to bottom and from left to right we see the disagreement scores (%dis) between the different APTs and the RTs of the telephone dialogues and the read speech. In addition, the statistics of the substitutions (sub), deletions (del) and insertions (ins) are presented to provide basic insight in the nature of the disagreements.

comparison with RT	telephone dialogues				read speech			
	subs	del	ins	%dis	subs	dels	ins	%dis
CAN-PT	9.1	1.1	8.1	18.3	6.3	1.2	2.6	10.1
DD-PT	26.0	18.0	3.8	47.8	16.1	7.4	3.6	27.0
KB-PT	9.0	2.5	5.8	17.3	6.3	3.1	1.5	10.9
CAN/DD-PT	21.5	6.2	7.1	34.7	13.1	2.0	4.8	19.9
KB/ DD-PT	20.5	7.8	5.4	33.7	12.8	3.1	3.6	19.5
$[CAN\text{-}PT]_d$	7.1	3.3	4.2	14.6	4.8	1.6	1.7	8.1
$[DD\text{-}PT]_d$	26.0	18.6	3.8	48.3	15.7	7.4	3.5	26.7
$[KB\text{-}PT]_d$	7.1	3.5	4.2	14.8	5.0	3.2	1.2	9.4
$[CAN/DD\text{-}PT]_d$	20.1	7.2	5.5	32.8	12.0	2.3	4.3	18.5
$[KB/\ DD\text{-}PT]_d$	19.3	9.4	4.5	33.1	11.6	3.1	3.1	17.8

Table 2: Comparison of APTs and human RTs. Fewer disagreements indicate better APTs.

The proportions of disagreements observed in the CAN-PTs and the KB-PTs were significantly different from each other ($p < .01$). The CAN-PT of the read speech was more similar to the RT than the KB-PT (Δ = 6.3% rel.) while the opposite held for the telephone dialogues (Δ = 5.9% rel.). The proportion of substitutions was about equal for the CAN-PTs and the KB-PTs. Most mismatches in the CAN-PTs were due to substitutions and insertions. There were more deletions than insertions in the KB-PT of the read speech, but there were fewer deletions than insertions in the KB-PT of the telephone dialogues. Detailed analysis of the aligned transcriptions showed that most frequent mismatches in the CAN-PTs and the KB-PTs of the two speech styles were due to voiced/unvoiced classifications of obstruents, and insertions of schwa and various consonants (in particular /r/, /t/ and /n/). Most substitutions and deletions (about 62-75% for the various transcriptions) occurred at word boundaries, but the absolute numbers in the KB-PTs were lower due to cross-word pronunciation modelling.

The disagreement scores obtained for the DD-PTs were much higher than the scores for the CAN-PTs and the KB-PTs. This holds for both speech styles. Most discrepancies between the DD-PTs and the RTs were substitutions and deletions. When compared to the CAN-PTs and the KB-PTs, in particular the high proportion of deletions and the wide variety of substitutions were striking. Not only did we observe consonant substitutions due to voicing, we also observed various consonant substitutions due to place of articulation, and vowel substitutions with schwa (and vice versa).

The proportion of disagreements in the CAN/DD-PTs and the KB/DD-PTs was lower than in the DD-PTs, but the individual CAN-PTs and KB-PTs resembled the RT better than the CAN/DD-PTs and the KB/DD-PTs. The CAN/DD-PTs and the KB/DD-PTs comprised twice as many substitutions and even more deletions than the CAN-PTs and the KB-PTs. Whereas the increased number of deletions in the CAN/DD-PT of the telephone dialogues coincided with a - be it moderate - decrease of insertion errors, the CAN/DD-PT of the read speech showed even more insertions than the CAN-PT.

Decision trees were applied to the ten aforementioned APTs (5 procedures x 2 speech styles). In nine out of ten cases, the application of decision trees improved the original transcriptions; only the $[DD\text{-}PT]_d$ of the telephone dialogues comprised more disagreements than the original DD-PT. The magnitude of the improvements differed substantially, though. The differences were negligible for the DD-PTs, somewhat larger for the APTs emerging from the combined procedures, and most outspoken for the CAN-PTs and KB-PTs. For both speech styles, the $[CAN\text{-}PT]_d$ proved most similar to the RT. The $[KB\text{-}PTs]_d$ were slightly worse. The $[CAN\text{-}PTs]_d$ comprised on average 20.5% fewer mismatches with the RTs than the original CAN-PTs, which is a significant improvement at a 99% confidence level. Likewise, we observed on average 14.1% fewer mismatches in the $[KB\text{-}PTs]_d$ than in the original KB-PTs ($p < .01$).

5. Discussion

5.1. Reflections on the evaluation procedure

In this study, the reference transcriptions were based on example transcriptions. Previous studies have shown that the use of an example transcription for verification speeds up the transcription process (relative to manual transcription from scratch), but that it also tempts human experts into adhering to the example transcription, despite contradicting acoustic cues in the speech signal. Demuynck et al. (2004), for example, reported cases where human experts preferred not to change the example transcription in the presence of contradicting acoustic cues, and cases where human experts approved phones in the example transcription that had no trace in the signal.

This observation is important for our study, since our RTs may have been biased towards the canonical example transcription they were based on. Considering that both the RTs and the KB-PTs were based on the CAN-PTs, the quality assessment of the CAN-PTs and the KB-PTs may have been positively biased. Consequently, the assessment of the DD-PTs may have been negatively biased, since the DD-PTs were based on the signal. Their assessment may have suffered from the human tendency to accept the canonical example transcription irrespective of the information in the acoustic signal (most probably because the human transcribers were instructed to change the example transcription only in case of obvious discrepancies).

In corpus creation projects, however, manually verified phonetic transcriptions are often preferred over automatic phonetic transcriptions. Therefore, in the light of the phonetic transcription of large speech corpora, our automatic procedures were tuned towards and evaluated in terms of this type of transcription.

5.2. On the suitability of low-cost automatic transcription procedures for the phonetic transcription of large speech corpora

5.2.1. Canonical transcription

The quality of the CAN-PT of the telephone dialogues (18% disagreement) already compared favourably to human inter-labeller disagreement scores reported in the literature. Greenberg et al. (1996), for example, reported 25 to 20% disagreements between manual transcriptions of American English telephone conversations, and Kipp et al. (1997) reported 21.2 to 17.4% inter-labeller disagreements between manual transcriptions of German spontaneous speech. Binnenpoorte (2006), however, reported better results: from 14 to 11.4% disagreements between manual transcriptions of Dutch spontaneous speech. The proportion disagreement between the CAN-PT and the human RT (10.1% disagreement) of the read speech was not yet at the same level as human inter-labeller disagreement scores reported in the literature. Kipp et al. (1996) reported 6.9 to 5.6% disagreements between human transcriptions of German read speech, and Binnenpoorte (2006) reported 6.2 to 3.7% disagreements between human transcriptions of Dutch read speech.

The apparent contradiction that the quality of the CAN-PT of the telephone dialogues already compared well to published human inter-labeller disagreement scores, whereas the CAN-PT of the read speech did not, may be explained by the different degrees of spontaneity in the speech samples. There is a higher chance for human inter-labeller disagreement in transcriptions of spontaneous than of well-prepared speech, since human transcribers have to transcribe or verify more phonological processes as speech becomes more spontaneous (Binnenpoorte et al. 2003). Nevertheless, considering the trade-off between overall transcription quality and the time and expenses involved in the human transcription and verification process, and considering the similarities with previously published human inter-labeller disagreement scores, we can conclude that the CAN-PTs were of a satisfactory quality. However, the high proportion of substitutions and insertions at word boundaries still implied the necessity of pronunciation variation modelling to better resemble the RT.

5.2.2. Data-driven transcription

Constrained phone recognition proved suboptimal for the generation of the targeted type of transcriptions. The high number and the wide variety of substitutions suggest that the use of a phonotactic model did not sufficiently tune our CSR towards the RT. The high number of deletions implies that, in spite of extensive tuning of the phone insertion penalty, our CSR had too large a preference for transcriptions containing fewer symbols. An informal inspection of the DD-PTs revealed that many deletions were unlikely, thus ruling out the possibility that the CSR analysed the signal more accurately than the human experts did. Kessens & Strik (2004) observed that the use of shorter acoustic models (e.g. using 20 ms models instead of 30 ms models) may reduce this tendency for deletions, but the diverse nature of the deletions in our study makes a substantial reduction of deletions through the mere use of different acoustic models rather unlikely.

5.2.3. Knowledge-based transcription

The use of linguistic knowledge to model pronunciation variation at the lexical level improved the quality of the transcription of the telephone dialogues, but it deteriorated the transcription of the read speech. This was probably due to the different degree of spontaneity in the two speech styles; the availability of pronunciation variants is probably more beneficial for the transcription of spontaneous speech, since more spontaneous speech comprises more pronunciation variation than well-prepared speech (Goddijn & Binnenpoorte, 2003). Most probably, the CSR preferred non-canonical variants in the read speech where the human transcribers adhered to the canonical example.

The knowledge-based recognition lexicon of the telephone dialogues comprised on average 1.39 pronunciation variants per lexeme, the lexicon of the read speech 1.47 variants per lexeme. The higher average number of pronunciation variants in the read speech lexicon is not contradictory, since the pronunciation variants of both speech styles were based on the canonical transcription, and not on the actual speech signal (which would, most probably, have highlighted more pronunciation variation in the telephone dialogues than in the read speech). Moreover, since the words in the telephone dialogues were shorter than the words in the read speech (an average of 3.3 vs. 4.1 canonical phones per word in the telephone dialogues and the read speech, resp.), the canonical transcription of the telephone dialogues was less susceptible to the application of rewrite rules than the CAN-PT of the read speech.

In order to estimate the possible impact of the application of KB rewrite rules on the CAN-PTs, we computed the maximum and minimum accuracy that could be obtained with the two KB recognition lexicons. For every chunk, every combination of the pronunciations of the words was consecutively aligned with the RT, and the highest and the lowest disagreement measures were retained. We found that the KB recognition lexicon of the telephone dialogues was able to provide KB-PTs of which 22.6 to 13.2% phones differed from the RT. The KB lexicon of the read speech was able to provide KB-PTs of which 16.3 to 7.4% phones differed from the RT. The eventual quality of the KB-PTs (17.3% and 10.9% disagreement for the telephone dialogues and the read speech, respectively) shows that there was still room for improvement, but that the acoustic models of our CSR often opted for suboptimal transcriptions. In this respect, the use of acoustic models trained on a KB-PT instead of a CAN-PT might have improved the selection of pronunciation variants.

5.2.4. Combined transcriptions

The blend of DD pronunciation variants with canonical or KB variants into CAN/DD and KB/DD lexicons allowed our CSR to better approximate human transcription behaviour than through constrained phone recognition alone, but the combination of the procedures did not outperform the canonical lexicon-lookup and the KB transcription procedure. The DD-PT benefited from the blend with the canonical and the KB pronunciation variants, while the influence of DD pronunciation variants increased the number of discrepancies between the resulting transcriptions and the RTs (as compared to the original CAN-PTs and KB-PTs).

5.2.5. Phonetic transcription with decision trees

Contrary to our expectations, the $[DD\text{-}PT]_d$ of the telephone dialogues comprised more (though not significantly more, $p > .1$) mismatches than the original DD-PT. The $[DD\text{-}PT]_d$ of the read speech was only slightly (again, not significantly, $p > .1$) better than the original DD-PT. This was probably due to the increased confusability in the recognition lexicons. The size of the lexicons had grown to an average of 9.5 variants per word in the recognition lexicon for the telephone dialogues, and an average number of 3.5 variants per word in the lexicon for the read speech. Note that, contrary to the pronunciation variants in the KB recognition lexicons, the pronunciation variants in the $[DD\text{-}PT]_d$ lexicons were based on the speech signal rather than on the application of phonological rewrite rules on the CAN-PT. This resulted, in particular for the $[DD\text{-}PTs]_d$ of the more spontaneous telephone dialogues, in more discrepancies with the RTs, all of which were modelled in the decision trees. Even after pruning unlikely pronunciation variants from the decision trees, the decision trees apparently still comprised enough pronunciation variants to pollute the recognition lexicon.

The small improvements obtained through the use of decision trees for the enhancement of the CAN/DD-PTs and the KB/DD-PTs, as well as the large improvements obtained through the use of decision trees for the enhancement of the CAN-PTs and the KB-PTs can be explained through the same line of reasoning. The numerous discrepancies between the CAN/DD-PTs and the KB/DD-PTs and the RTs yielded numerous pronunciation variants in the resulting recognition lexicons (though less than in the DD-PT lexicons). The higher similarity between the original $[CAN\text{-}PT]_d$, the $[KB\text{-}PTs]_d$ and the RTs, led to fewer branches in the decision trees and fewer pronunciation variants in the resulting recognition lexicons. Moreover, the corresponding lexical probabilities were intrinsically more robust than the probabilities in the DD lexicons comprising more pronunciation variants per lexeme. Since the $[CAN\text{-}PTs]_d$ were better than the $[KB\text{-}PTs]_d$ of both speech styles, and since informal inspection of the rules seems to suggest that the KB-PTs and the $[KB\text{-}PTs]_d$ could not be drastically improved through the modelling of vowel reduction and vowel deletion, we conclude that prior knowledge about the phonological processes of a language, and the subsequent implementation of knowledge-based phonological rules are not necessary to approximate the quality of manually verified phonetic transcriptions of large speech corpora. Instead, the use of decision trees and a small sample of manually verified phonetic transcriptions suffice to make canonical transcriptions approximate human transcription behaviour.

5.3. What about the remaining discrepancies?

The number of remaining discrepancies in the $[CAN\text{-}PTs]_d$ of the telephone dialogues (14.6% disagreement) and the read speech (8.1% disagreement) was only slightly higher than human inter-labeller disagreement scores reported in the literature. Recall that Binnenpoorte (2006) reported human inter-labeller disagreements between 14 and 11.4% on transcriptions of Dutch spontaneous speech, and between 6.2 and 3.7% disagreements on transcriptions of Dutch read speech. A closer look at the 20 most frequent dissimilarities distinguishing the $[CAN\text{-}PTs]_d$ from the human RTs, shows a comparable number of insertions and deletions, and a set of substitutions in which the mismatches between voiced and voiceless phones were dominant. Similar differences were observed between manual transcriptions that were based on the same example transcription (Binnenpoorte et al., 2003). The remaining mismatches can be largely attributed to the very nature of human transcription behaviour. Varying disagreement scores like the ones reported in Binnenpoorte et al. (2003) seem to suggest that it is intrinsically very hard, if not impossible, to model the often whimsical human transcription behaviour with one automatic transcription procedure. Therefore, we are inclined to believe that we should not try to further model the inconsistencies in manual transcriptions of speech, and we conclude that we found a very quick, simple and cheap transcription procedure approximating human transcription behaviour for the transcription of large speech samples. Our procedure uniformly applies to well-prepared and spontaneous speech.

6. Conclusions

The aim of our study was to find an automatic transcription procedure to substitute human efforts in the phonetic transcription of large speech corpora whilst ensuring high transcription quality. To this end, ten automatic transcription procedures were used to generate a phonetic transcription of spontaneous speech (telephone dialogues) and well-prepared speech (read-aloud texts). The resulting transcriptions were compared to a manually verified phonetic transcription, since this kind of transcription is often preferred in corpus design projects.

An analysis of the discrepancies between the different transcriptions and the reference transcription showed that purely data-driven transcription procedures or procedures partially relying on data-driven input could not approximate the human reference transcription. Much better results were obtained by implementing phonological knowledge from the linguistic literature. The best results, however, were obtained by expanding canonical transcriptions with decision trees trained on the alignment of canonical transcriptions and manually verified phonetic transcriptions. In fact, our results show that an orthographic transcription, a canonical lexicon, a small sample of manually verified phonetic transcriptions, software for the implementation of decision trees and a standard continuous speech recogniser are sufficient to approximate human transcription quality in projects aimed at generating broad phonetic transcriptions of large speech corpora.

Our procedures uniformly applied to well-prepared and spontaneous speech. Hence, we believe that the performance of our procedures will generalise to other speech corpora, provided that the emerging automatic phonetic transcriptions are evaluated in terms of a similar reference transcription, viz. a manually verified automatic phonetic transcription of speech.

Acknowledgement

The work of Christophe Van Bael was funded by the Speech Technology Foundation (Stichting Spraaktechnologie, Utrecht, The Netherlands).

References

Bellegarda, J.R. (2005). Unsupervised, language-independent grapheme-to-phoneme conversion by latent analogy. In: *Speech Communication*, vol. 46/2, pp. 140-152.

Binnenpoorte, C., Goddijn, S.M.A., Cucchiarini, C. (2003). How to Improve Human and Machine Transcriptions of Spontaneous Speech. In: *Proceedings of ISCA/IEEE Workshop on Spontaneous Speech Processing and Recognition (SSPR)*, Tokyo, Japan, pp. 147-150.

Binnenpoorte, D., Cucchiarini, C. (2003). Phonetic Transcription of Large Speech Corpora: How to boost efficiency without affecting quality. In: *Proceedings of ICPhS*, Barcelona, Spain, pp. 2981-2984.

Binnenpoorte, D., (2006). *Phonetic transcription of large speech corpora.* Ph.D. thesis, Radboud University Nijmegen, the Netherlands.

Booij, G. (1999). *The phonology of Dutch*. Oxford University Press, New York.

Cucchiarini C. (1993). *Phonetic transcription: a methodological and empirical study.* Ph.D. thesis, University of Nijmegen.

Demuynck, K., Laureys, T., Gillis, S. (2002). Automatic generation of phonetic transcriptions for large speech corpora. In: *Proceedings of International Conference on Spoken Language Processing (ICSLP)*, Denver, USA, pp. 333-336.

Demuynck. K., Laureys, T., Wambacq, P., Van Compernolle, D. (2004). Automatic phonemic labeling and segmentation of spoken Dutch. In: *Proceedings of LREC*, Lisbon, Portugal, pp. 61-64.

Elffers, B, Van Bael, C., Strik, H. (2005). *ADAPT: Algorithm for Dynamic Alignment of Phonetic Transcriptions*. Internal report, CLST, Radboud University Nijmegen. http://lands.let.ru.nl/literature/elffers.2005.1.pdf.

Godfrey, J., Holliman, E. and McDaniel, J. (1992) SWITCHBOARD: Telephone speech corpus for research and development. *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, San Francisco, USA, pp. 517-520.

Goddijn, S.M.A. & Binnenpoorte, D. (2003). Assessing Manually Corrected Broad Phonetic Transcriptions in the Spoken Dutch Corpus. In: *Proceedings of ICPhS*, Barcelona, Spain, pp. 1361-1364.

Greenberg, S., Hollenback, J. and Ellis, D. (1996). Insights into spoken language gleaned from phonetic transcription of the Switchboard corpus. In: *Proceedings of the International Conference on Spoken Language Processing (ICSLP)*, Philadelphia, USA.

Hess, W., Kohler, K.J., Tillman, H.-G. (1995) The Phondat-Verbmobil speech corpus. In: *Proceedings of Eurospeech*, Madrid, Spain, pp. 863-866.

Jande, P.A. (2005). Inducing Decision Tree Pronunciation Variation Models from Annotated Speech Data. In: *Proceedings of Interspeech*, Lisbon, Portugal, pp. 1945-1948.

Kessens, J.M., Wester, M., Strik, H. (1999). Improving the performance of a Dutch CSR by modelling within-word and cross-word pronunciation variation. In: *Speech Communication*, vol. 29, pp. 193-207.

Kessens, J.M., Strik, H. (2004). On automatic phonetic transcription quality: lower word error rates do not guarantee better transcriptions. In: *Computer, Speech and Language*, vol. 18(2), pp. 123-141.

Kipp, A., Wesenick, M.-B., Schiel F. (1996) Automatic detection and segmentation of pronunciation variants in German speech corpora. In: *Proceedings of ICSLP*, Philadelphia, USA, pp. 106-109.

Kipp, A., Wesenick, M.-B., Schiel F. (1997). Pronunciation modelling applied to automatic segmentation of spontaneous speech. In: *Proceedings of Eurospeech*, Rhodes, Greece, pp. 1023-1026.

Koskenniemi, K. (1983) *Two-level morphology: A general computational model of word-form recognition and production.* Tech. Rep. Publication No. 11, Dept. of General Linguistics, University of Helsinki.

Maekawa, K. (2003). Corpus of Spontaneous Japanese: Its design and evaluation. In: *Proceedings of ISCA/IEEE Workshop on Spontaneous Speech Processing and Recognition (SSPR)*, Tokyo, Japan.

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

Quinlan, J. R. (1993). *C4.5: Programs for Machine Learning*. San Mateo: Morgan Kaufmann.

Riley, M., Byrne, W., Finke, M., Khudanpur, S., Ljolje A, McDonough, J., Nock, H., Saraçlar, M., Wooters, C., Zavaliagkos, G. (1999). Stochastic pronunciation modelling from hand-labelled phonetic corpora. In: *Speech Communication*, vol. 29, pp. 209-224.

Saraçlar, M., Khundanpur, S (2004). Pronunciation change in conversational speech and its implications for automatic speech recognition. In: *Computer, Speech and Language*, vol. 18, pp. 375-395.

Strik, H. (2001). Pronunciation adaptation at the lexical level. In: *Proceedings of the ISCA Tutorial & Research Workshop (ITRW) 'Adaptation Methods for Speech Recognition'*, Sophia-Antipolis, France, pp. 123-131.

TIMIT Acoustic-Phonetic Continuous Speech Corpus (1990). National Institute of Standards and Technology Speech Disc 1-1.1, NTIS Order No. PB91-505065, 1990.

Tjalve, M., Huckvale, M., (2005). Pronunciation variation modelling using accent features. In: *Proceedings of Interspeech*, Lisbon, Portugal, pp.1341-1344.

Van Bael, C., Van den Heuvel, H., Strik, H. (2006). Validation of phonetic transcriptions in the context of automatic speech recognition. Submitted to: *Language Resources and Evaluation.*

Wang, L., Zhao, Y., Chu, M., Soong, F., Cao, Z. (2005). Phonetic transcription verification with generalised posterior probability. In: *Proceedings of Interspeech*, Lisbon, pp. 1949-1953.

Wesenick, M.-B., Kipp, A. (1996) Estimating the quality of phonetic transcriptions and segmentations of speech signals. In: *Proceedings of ICSLP*, Philadelphia, USA, pp. 129-132.

Wester, M. (2003). Pronunciation modeling for ASR - knowledge-based and data-derived methods. In: *Computer Speech & Language*, vol. 17/1, pp. 69-85.

Witten, I.H., Frank, E. (2005). *Data Mining: Practical machine learning tools and techniques*, 2nd Edition, Morgan Kaufmann, San Francisco, USA.

Yang, Q., Martens, J.-P., (2000). Data-driven lexical modelling of pronunciation variations for ASR. In: *Proceedings of ICSLP*, Beijing, China, pp. 417-420.

Young, S., Evermann, G., Kershaw, D., Moore, G., Odell, J., Ollason, D., Valtchev, V., Woodland, P. (2001). *The HTK book (for HTK version 3.1)*, Cambridge University Engineering Department.

Annotation of Grammatemes in the Prague Dependency Treebank 2.0

Magda Razímová, Zdeněk Žabokrtský

Institute of Formal and Applied Linguistics, Charles University, Prague
Malostranské náměstí 25, Prague 1, 118 00, Czech Republic
{razimova,zabokrtsky}@ufal.mff.cuni.cz

Abstract

In this paper we report our work on the system of grammatemes (mostly semantically-oriented counterparts of morphological categories such as number, degree of comparison, or tense), the concept of which was introduced in Functional Generative Description, and has been recently further elaborated in the layered annotation scenario of the Prague Dependency Treebank 2.0. We present also a hierarchical typology of tectogrammatical nodes, which is used as a formal means for ensuring presence or absence of respective grammatemes.

1. Introduction

Human language, as an extremely complex system, has to be described in a modular way. Many linguistic theories attempt to reach the modularity by decomposing language description into a set of layers, usually linearly ordered along an abstraction axis (from text/sound to semantics/pragmatics). One of the common features of such approaches is that word forms occurring in the original surface expression are substituted (for the sake of higher abstraction) with their lemmas at the higher layer(s). Obviously, the inflectional information contained in the word forms is not present in the lemmas. Some information is 'lost' deliberately and without any harm, since it is only imposed by government (such as case for nouns) or agreement (congruent categories such as person for verbs or gender for adjectives). However, the other part of the inflectional information (such as number for nouns, degree for adjectives or tense for verbs) is semantically indispensable and must be represented by some means, otherwise the sentence representation becomes deficient (naturally, the representations of sentence pairs such as '*Peter met his youngest brother*' and '*Peter meets his young brothers*' must not be identical at any level of abstraction). At the tectogrammatical layer of Functional Generative Description (FGD, (Sgall, 1967), (Sgall et al., 1986)), which we use as the theoretical basis of our work, these means are called grammatemes.[1]

The theoretical framework of FGD has been implemented in the Prague Dependency Treebank 2.0 project (PDT 2.0, (Hajičová et al., 2001)), which aims at a complex annotation of large amount of Czech newspaper texts. Although grammatemes are present in the FGD for decades, in the context of PDT they were paid for a long time a considerably less attention, compared e.g. to valency, topic-focus articulation, or coreference. However, in our opinion grammatemes will play a crucial role in NLP applications of FGD and PDT (e.g., machine translation is impossible without realizing the differences in the above pair of example sentences). That is why we decided to further elaborate the system of grammatemes and to implement it in the PDT 2.0 data. This paper outlines some of the results of more than two years of the work on this topic.

The paper is structured as follows: after introducing the basic properties of the PDT 2.0 with focus on the tectogrammatical layer in Section 2., we will describe the classification of t-layer nodes in Section 3., enumerate and exemplify the individual grammatemes and their values in Section 4. After outlining the basic facts about the (mostly automatic) annotation procedure in Section 5. we will add some final remarks in Section 6.

2. Sentence Representation in the Prague Dependency Treebank 2.0

In the Prague Dependency Treebank annotation scenario, three layers of annotation are added to Czech sentences (see Figure 1 (a)):[2]

- morphological layer (m-layer), on which each token is lemmatized and POS-tagged,
- analytical layer (a-layer), on which a sentence is represented as a rooted ordered tree with labeled nodes and edges, corresponding to the surface-syntactic relations; one a-layer node corresponds to exactly one m-layer token,
- tectogrammatical layer (t-layer), which will be briefly described later in this section.

The full version of the PDT 2.0 data consists of 7,129 manually annotated textual documents, containing altogether 116,065 sentences with 1,960,657 tokens (word forms and punctuation marks). All these documents are annotated at the m-layer. 75 % of the m-layer data are annotated at the a-layer (5,338 documents, 87,980 sentences, 1,504,847 tokens). 59 % of the a-layer data are annotated also at the t-layer (i.e. 44 % of the m-layer data; 3,168 documents,

[1] Just for curiosity: almost the same term 'grammemes' is used for the same notion in the Meaning-Text Theory (Mel'čuk, 1988), although to a large extent the two approaches were created independently.

[2] Technically, there is also one more layer below these three layers which is called w-layer (word layer); on this layer the original raw-text is only segmented into documents, paragraphs and tokens and all these units are enriched with identifiers.

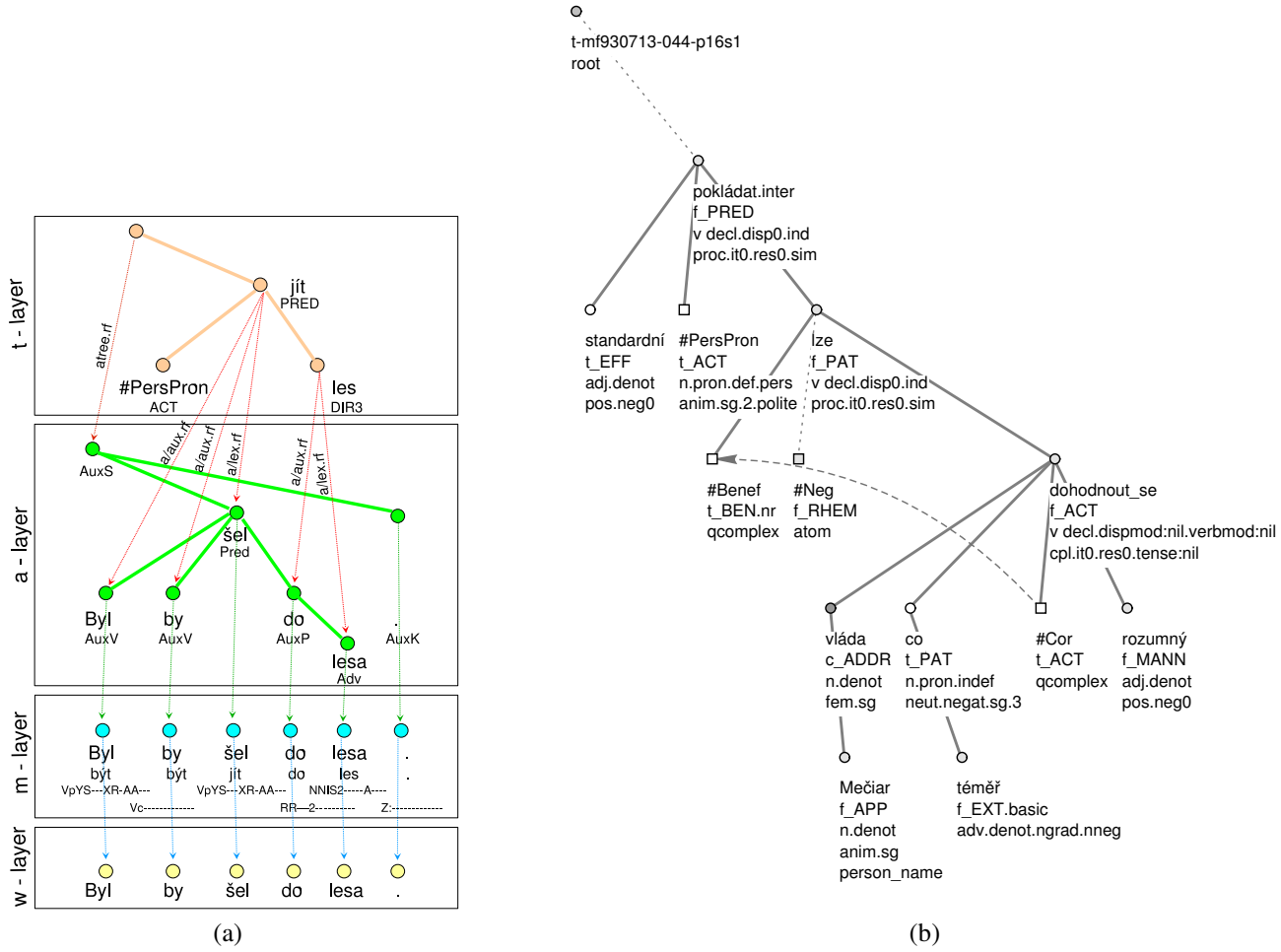

Figure 1: (a) PDT 2.0 annotation layers (and the layer interlinking) illustrated (in a simplified fashion) on the sentence *Byl by šel do lesa.* ([He] would have gone into forest.), (b) tectogrammatical representation of the sentence: *Pokládáte za standardní, když se s Mečiarovou vládou nelze téměř na ničem rozumně dohodnout?* (Do you find it standard if almost nothing can be reasonably agreed on with Mečiar's government?)

49,442 sentences, 833,357 tokens).[3] The annotation at the t-layer started in 2000 and was divided into four areas:

a. building the dependency tree structure of the sentence including labeling of dependency relations and valency annotation,

b. topic / focus annotation,

c. annotation of coreference (i.e. relations between nodes referring to the same entity),

d. annotation of grammatemes and related attributes, the description of which is the main objective of this paper.

After the annotation of data had finished in 2004, an extensive cross-layer checking took over a year. The CD-ROM including the final annotation of PDT 2.0-data, a detailed documentation as well as software tools is to be publicly released by Linguistic Data Consortium in 2006.[4]

[3]The previous version of the treebank, PDT 1.0, was smaller and contained only m-layer and a-layer annotation (Hajič et al., 2001).

[4]See `http://ufal.mff.cuni.cz/pdt2.0/`

At the t-layer, the sentence is represented as a dependency tree structure built of nodes and edges (see Figure 1 (b)). Tectogrammatical nodes (t-nodes) represent auto-semantic words (including pronouns and numerals) while functional words such as prepositions have no node in the tree (with some exception of technical nature: e.g. coordinating conjunctions used for representation of coordination constructions are present in the tree structure). Each t-node is a complex data structure – it can be viewed as a set of attribute-value pairs, or even as a typed feature structure as used in unification grammars such as HPSG (Pollard and Sag, 1994).

For the purpose of our contribution, the most important attributes are the attribute t-lemma (tectogrammatical lemma), attribute functor, grammatemes and the classifying attributes nodetype and sempos. The annotation of attributes t-lemma and functor belongs to the area marked above as (a); these attributes will be introduced in the next paragraphs. Grammatemes and the attributes nodetype and sempos – all of them coming under the area (d) – will be characterized from the standpoint of annotation in Section 3. (The annotation of attributes belonging to the areas

(b) and (c) goes beyond the scope of this paper.)
The attribute t-lemma contains the lexical value of the t-node, or an 'artificial' lemma. The lexical value of the t-node is mostly a sequence of graphemes corresponding to the 'normalized' form of the represented word (i.e. infinitive for verbs or nominative form for nouns). In some cases, the t-lemma corresponds to the basic word from which the represented word was derived, e.g. in Figure 1 (b), the possessive adjective *Mečiarova* (*Mečiar's*) is represented by the t-lemma *Mečiar*, or the adverb *rozumně* (*reasonably*) is represented by the adjectival t-lemma *rozumný* (*reasonable*). The artificial t-lemma appears at t-nodes that have no counterpart in the surface sentence structure (e.g. the t-lemma #Gen at a verbal complementation not occurring in the surface structure because of its semantic generality), or it corresponds to personal pronouns, no matter whether expressed on the surface or not (e.g. the t-lemma #PersPron at the t-node in Figure 1 (b)). The dependency relation between the t-node in question and its parent t-node is stored in the attribute functor, e.g. functor EFF at the t-node with t-lemma *standardní* (*standard*), which plays the role of an effect of the predicate in the sentence displayed in Figure 1 (b).

3. Two-level Typing of Tectogrammatical Nodes

While the attributes t-lemma and functor are attached to each t-node of the tectogrammatical tree, grammatemes are relevant only for some of them. The reason for this difference consists in the fact that only some words represented by t-nodes bear morphological meanings.

3.1. Types of Tectogrammatical Nodes

To differentiate t-nodes that bear morphological meanings from those without such meanings, a classification of t-nodes was necessary. Based on the information captured by the above mentioned attributes t-lemma and functor, eight types of t-nodes were distinguished. The appurtenance of the t-node to one of the types is stored in the attribute nodetype.[5]

- **Complex nodes** (nodetype='complex') as the most important node type should be named in the first place: since they represent nouns, adjectives, verbs, adverbs and also pronouns and numerals (i.e. words expressing morphological meanings), they are the only ones with which grammatemes are to be assigned.

The other seven types of t-nodes and the corresponding values of the attribute nodetype are as follows:

- **The root of the tectogrammatical tree** (nodetype='root') is a technical t-node the child t-node of which is the governing t-node of the sentence structure.
- **Atomic nodes** (nodetype='atom') are t-nodes with functors RHEM, MOD etc. – they represent rhematizers, modal modifications etc.
- **Roots of coordination and apposition constructions** (nodetype='coap') contain the t-lemma of the coordinating conjunction or an artificial t-lemma of a punctuation symbol (e.g. #Comma).
- **Parts of foreign phrases** (nodetype='fphr') are components of phrases that do not follow rules of Czech grammar (labeled by a special functor FPHR in the tree).
- **Dependent parts of phrasemes** (nodetype='dphr') represent words that constitute a single lexical unit with their parent t-node (labeled by a special functor DPHR in the tree); the meaning of this unit does not follow from the meanings of its component parts.
- **Roots of foreign and identification phrases** (nodetype='list') are nodes with special artificial t-lemmas (#Forn and #Idph), which play the role of a parent of a foreign phrase (i.e. of nodes with nodetype='fphr' – see above) or the role of a parent of a phrase having a function of a proper name.
- So called **quasi-complex nodes** (nodetype= 'qcomplex') stand mostly for obligatory verbal complementations that are not present in the surface sentence structure (i.e. they have the same functors as complex nodes but, unlike them, quasi-complex t-nodes have artificial t-lemmas, e.g. #Gen).

3.2. Semantic Parts of Speech

Not all morphological meanings (chosen as tectogrammatically pertinent) are relevant for all complex t-nodes (cf., for example, the category of tense at nouns or the degree of comparison at verbs). As we did not want to introduce any 'negative' value to identify the non-presence of the given morphological meaning at a t-node (i.e., if all grammatemes would be annotated at each complex t-node, the negative value would be filled in at the irrelevant ones), the attribute sempos for sorting the t-nodes according to morphological meanings they bear had to be introduced into the attribute system.
The groups into which the complex t-nodes were further divided are called semantic parts of speech. According to basic onomasiological categories of substance, quality, event and circumstance (Dokulil, 1962), four semantic parts of speech were distinguished: semantic nouns, semantic adjectives, semantic verbs and semantic adverbs. These groups are not identical with the 'traditional' parts of speech: while ten traditional parts of speech are discerned in Czech and the appurtenance of the word to one of them is captured by a morphological tag (i.e. by an attribute of m-layer in the PDT 2.0), the 'only' four semantic parts of speech are categories of the t-layer and are captured by the attribute sempos (values n, adj, v and adv). The relations between semantic and traditional parts of speech are demonstrated in Figure 2. We would like to illustrate them on the example of semantic adjectives in more detail.
The following groups traditionally belonging to different parts of speech count among the semantic adjectives: (i) traditional adjectives, (ii) deadjectival adverbs, (iii) adjectival pronouns, and (iv) adjectival numerals.

[5] Some of the nodetype values are present in Figure 1 (b). If none of the nodetype values is indicated with the t-node, the nodetype is 'complex'.

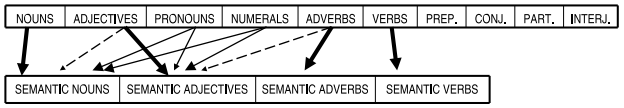

Figure 2: Relations of traditional parts of speech to their semantic counterparts. Arrows in bold denote a prototypical relation, thin arrows indicate the distribution of pronouns and numerals into semantic parts of speech and dotted arrows stand for the classification according to derivational relations.

(i) Traditional adjectives, e.g. *standardní* (*standard*) in Figure 1 (b), are mostly regarded as semantic adjectives (with the already mentioned exception of possessive adjectives converted to nouns).
(ii) At the t-layer, deadjectival adverbs, e.g. *rozumně* (*reasonably*) in Figure 1 (b), are represented by the t-lemma of the corresponding adjective, here by the t-lemma *rozumný* (*reasonable*). In this way, a derivational relation is followed: the word is represented by its basic word. Other types of derivational relations analyzed in PDT 2.0 will be introduced in the next sections.
(iii) and (iv) Since there are no groups such as 'semantic pronouns' or 'semantic numerals' at the t-layer, these words were distributed into semantic nouns and adjectives according to their function they fill in the sentence. While pronouns and numerals filling typical positions of nouns (such as agent or patient) belong to semantic nouns, pronouns and numerals playing an adjectival role are classified as semantic adjectives. For examples of nominal usage of the pronoun *který* (*which*) and of the numeral *sto* (*hundred*) see sentences (1), and (2) respectively:

(1) *Kurz, který*.n *jsem si vybral, je špatný.*
The course that I have chosen is bad.

(2) *Už vedl sto*.n *kurzů.*
He has already taught one hundred courses.

For examples of adjectival usage of the pronoun *který* (*which*) and of the numeral *tři* (*three*) see sentences (3), and (4) respectively:

(3) *Který*.adj *kurz si mám vybrat?*
Which course should I choose?

(4) *Vyučuje tři*.adj *kurzy.*
He teaches three courses.

The subgroups of semantic adjectives presented above are viewed as constituting the inner structure of this class. Also the classes of semantic nouns and semantic adverbs were sub-classified in a similar way. (Semantic verbs cannot be subdivided by the same principles as the other semantic parts of speech.)[6] The appurtenance of a t-node to a concrete subgroup of semantic parts of speech is captured as a detailed value of the attribute sempos (e.g. adj.denot or adj.quant.def in Figure 3).

[6]The sub-classification of semantic verbs is one of our future aims; properties of verbal systems in other languages (as studied e.g. in (Bybee, 1985)) will be considered.

The t-node hierarchy including the detailed subclassification of semantic adjectives is displayed in Figure 3.

4. Grammatemes and Their Values

There are 15 grammatemes at the t-layer of PDT 2.0. Grammatemes number, gender, person and politeness were assigned to t-nodes belonging to the subclasses of semantic nouns. The grammatemes degcmp, negation, numertype and indeftype were annotated with semantic nouns as well as with semantic adjectives, the latter two of them also with semantic adverbs. The other seven grammatemes belong to semantic verbs: tense, aspect, verbmod, deontmod, dispmod, resultative, and iterativeness.
All the grammatemes will be explained and exemplified in the following subsections one by one. A separate subsection is devoted to a more detailed discussion about pronominal words.

4.1. Number

The grammateme **number** is the tectogrammatical counterpart of the morphological category of number – the grammateme values, sg (for singular) and pl (for plural), mostly correspond to the values of this morphological category, e.g. the noun *vláda*.sg (*government*) in Figure 1 (b) is in singular while *vlády*.pl (*governments*) would be plural. However, as the grammateme captures the 'semantic' number, its value differs from that of the morphological category in some cases: e.g. while the morphological number of pluralia tantum is always 'plural' (e.g. the Czech word *dveře*, *door*), the tectogrammatical singular in a sentence like (5) is discerned from the tectogrammatical plural in the sentence (6) – at these nouns, the decision by an annotator was necessary; if such a decision were not possible on the basis of context (e.g. in the sentence (7)), a special value nr ('not recognized') was assigned.

(5) *Neotevírej tyto dveře*.sg
Do not open this door.

(6) *Šel dlouhou chodbou*
He walked through a long corridor
a minul několikery dveře.pl
and passed several doors.

(7) *Otevřel dveře*.nr
He opened the door/doors.

4.2. Gender

In PDT 2.0, values of the grammateme **gender** correspond to the morphological gender: anim (for masculine animate), inan (for masculine inanimate), fem (for feminine), and neut (for neuter).

4.3. Person and Politeness

The grammatemes **person** and **politeness** have been assigned to one subclass of semantic nouns that contains personal pronouns. These words are represented by the artificial t-lemma #PersPron at the t-layer (e.g. in the Figure 1 (b), where the t-node with the t-lemma #PersPron represents the actor that is not present in the surface sentence structure). The values of the former grammateme (1, 2, 3) distinguish among the 1st, 2nd and 3rd person pronouns;

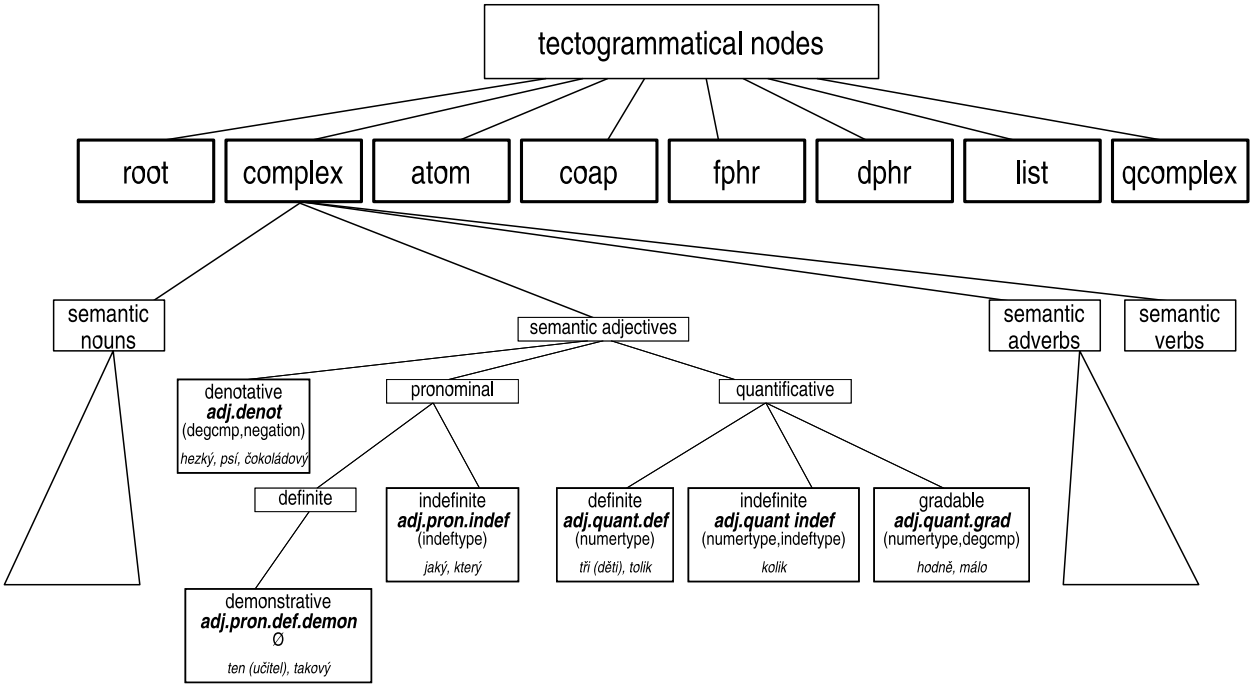

Figure 3: Hierarchy of t-nodes. The first branching renders the nodetype distinctions. Then, only complex t-nodes are further subdivided into four semantic parts of speech. Semantic nouns, semantic adjectives and semantic adverbs are further subclassified. Due to space limitations, only the subclassification of semantic adjectives is displayed in detail. In the leaf t-nodes of this subclassification, the values of attribute sempos is given on the second line and the list of grammatemes associated with the given class follows on the third line in the boxes.

the values of the latter one (basic, polite) discern the common from the polite usage of 2nd person pronouns. The surface pronoun is derived from the combination of t-lemma and values of grammatemes number, gender, person and politeness. E.g., the pronoun *vy* (*you*) in the sentence (8) is derived from the tectogrammatical representation #PersPron+**pl**+anim+2+**basic** in contrast to the same pronoun in the sentence (9) that is derived from the representation #PersPron+**sg**+anim+2+**polite**.

(8) *Vy jste vybrali dobrý kurz.*
'You have chosen a good course'
(- said to a group of persons)

(9) *Vy jste vybral dobrý kurz.*
'You have chosen a good course'
(- said politely to a single person)

4.4. Degree of Comparison

The grammateme **degcmp** corresponds to the morphological category of degree of comparison. Besides the values pos (for positive), comp (comparative) and sup (superlative), a special value acomp for comparative forms of adjectives/adverbs without a comparative meaning (so called 'absolute comparative', also 'elative') was established. The common usage of comparative forms such as *Jan je starší*.comp *než ona* (*Jan is elder than her*) was distinguished from the absolute usage e.g. in *starší*.acomp *muž* (*an elder man*) by the manual annotation.

4.5. Types of Numeral and Pronominal Expressions

Neither the grammateme **numertype** nor **indeftype** have a counterpart in the traditional set of morphological categories. They capture information on derivational relations among numerals, and pronominal words respectively, analyzed at the t-layer: derived words are represented by the t-lemma of its basic word and the feature that would be lost by such a representation is captured by values of these grammatemes. As all types of numerals are seen as derivations from the corresponding basic numeral and thus represented by its t-lemma, the grammateme numertype captures the type of the numeral in question. The surface numeral is then derived from the t-lemma and the value of this grammateme, e.g. the ordinal numeral *třetí* (*the third*) is derived form the following tectogrammatical representation: t-lemma *tři* (*three*) + numertype='ord' (for ordinal). Besides the value ord, the value set of this grammateme involves four other values: basic for basic numerals (*tři kurzy–three courses*), frac for fractional numerals (*třetina kurzu–the third of the course*), kind for numerals concerning the number of kinds/sorts (*trojí víno–three sorts of wine*), and set for numerals with meaning of the number of sets (*troje klíče–three sets of keys*).

In a similar vein, indefinite, negative, interrogative, and relative pronouns are represented by the t-lemma corresponding to the relative pronoun – the specific semantic feature is stored in the grammateme indeftype. Surface pronouns are derived from the lemma and the value of this grammateme: e.g. the indefinite pronoun *někdo* (*somebody*) and the negative pronoun *nikdo* (*nobody*) are derived from the

following tectogrammatical representations: t-lemma *kdo* + indeftype='indef', and t-lemma *kdo* + indeftype='negat' respectively.[7] Such representation of derivational relations makes it possible to represent all these words by a very small set of t-lemmas. The question of applying similar principles to pronominal words in other languages will be mentioned in Subsection 4.11.

4.6. Negation

Also the grammateme **negation** captures a lexical information needed for derivation of surface forms: it enables to represent both, the positive and the negative forms of adjectives, adverbs and (temporarily, only a group of) nouns by a single t-node with the same t-lemma – e.g. the adjective *standardní* (*standard*) in Figure 1 (b) as well as its negative form *nestandardní* (*non-standard*) are represented by the t-node with t-lemma *standardní* and the absence/presence of negation is captured by the value of the grammateme: the value neg0 was assigned to the t-node representing the positive form, the value neg1 to the t-node corresponding to the negative form.[8]

4.7. Tense

The grammateme **tense** corresponds to the morphological category of tense. The values sim (simultaneous with the moment of speech/with other event), ant (anterior to the moment of speech/to other event), and post (posterior to the moment of speech/to other event)[9] have been assigned automatically.

4.8. Aspect

The grammateme **aspect** is the tectogrammatical counterpart of the category of aspect. As there are verbs in Czech that can express both, imperfective and perfective aspects by the same forms (so called bi-aspectual verbs), manual annotation was necessary to make a decision with these verbs.

4.9. Verbal Modalities

There are three grammatemes concerning modality. The grammateme **verbmod** captures if the represented verbal form expresses the indicative (value ind), the imperative (imp), or the conditional mood (cdn). Since modal verbs do not have a t-node of their own at the t-layer (for explanation see (Panevová et al., 1971)), the deontic modality expressed by these verbs is stored in the grammateme **deontmod**, e.g. the predicate of the sentence *Už může odejít* (*He can already leave*) is represented by a t-node with t-lemma *odejít* (*to leave*) and the modality is stored as the value poss (for possibilitive) in the grammateme deontmod. The last of the modality grammatemes, the grammateme **dispmod**, concerns the so-called dispositional modality. This type of modality is represented by a special syntactic construction involving a 'reflexive-passive' verb construction, a dative form of a noun/personal pronoun playing the role of agent, and a modal adverb, e.g. the sentence (10):

(10) *Studentům se ta kniha čte dobře.*
Lit. *To students the book reads well.*
It is easy for the students to read the book.

4.10. Resultative and Iterativeness

While the grammateme **resultative** (values res1, res0) reflects the fact whether the event is/is not presented as a resultant state, the last verbal grammateme **iterativeness** indicates whether the event is/is not viewed as a repeated (multiplied) action (values it1, it0).

4.11. Pronominal Words at the T-layer

In this chapter, we would like to provide a deeper view into the principles of representation of pronominal words at the t-layer of PDT 2.0, and then to outline how this representation can be applied to such words in English or German.
As already mentioned above, pronouns are represented by a minimal set of t-lemmas at the t-layer. Personal pronouns by a single (artificial) t-lemma #PersPron; grammatemes assigned to the t-nodes of personal pronouns were presented in the previous chapter. Indefinite, negative, in-

T-lemma:	*kdo*	*co*	*který*	*jaký*
indefype:				
relat	kdo	co	který, jenž	jaký
indef1	někdo	něco	některý	nějaký
indef2	kdosi kdos	cosi cos	kterýsi	jakýsi
indef3	kdokoli kdokoliv	cokoli cokoliv	kterýkoli kterýkoliv	jakýkoli jakýkoliv
indef4	ledakdo leckdo	ledaco lecco	leckterý ledakterý	lecjaký ledajaký
indef5	kdekdo	kdeco	kdekterý	kdejaký
indef6	kdovíkdo málokdo	kdovíco máloco	kdovíkterý málokterý	kdovíjaký všelijaký
inter	kdo kdopak	co copak	který kterýpak	jaký jakýpak
negat	nikdo	nic	žádný	nijaký
total1	všechen	všechno vše	-	-
total2	-	-	každý	-

Table 1: The indeftype grammateme has actually eleven values (1st column in the table). It makes it possible to represent all semantic variants of pronouns *kdo* (*somebody*), *co* (*something*), *který* (*that*) and *jaký* (*what*) (in the 2nd, 3rd, 4th and 5th column) by only four t-lemmas at the t-layer.

[7] A similar treatment of indefinite and negative pronouns as of two subtypes of the same entity can be found in (Helbig, 2001).

[8] Unlike this representation, negative verbal forms (verbal negation is expressed also by the prefix *ne-* in Czech) are represented by a sub-tree consisting of a t-node with a verbal t-lemma the child of which is a t-node with the artificial t-lemma #Neg; cf. the representation of the negated verb *nelze* ((it) *can not be*) by two t-nodes, with the t-lemmas *lze* ((it) *can be*) and #Neg, in Figure 1 (b). The explanation can be found in (Hajičová, 1975).

[9] As the class of semantic verbs has not been sub-classified yet and all verbal grammatemes were annotated with each verbal t-node, a special value nil was inserted into the value system for cases when the represented word does not express a feature captured by the grammateme (cf. the value of grammateme tense at a t-node representing an infinitive form).

terrogative and relative pronouns are all represented by a t-lemma corresponding to the relative pronoun. In this way, only four lemmas – i.e. *kdo* (*somebody*), *co* (*something*), *který* (*which*) and *jaký* (*what*) – are sufficient to represent all Czech pronouns of named types at the t-layer. The pronouns with corresponding values of the grammateme indeftype are displayed in Table 1.

Since the semantic features stored in the grammateme indeftype are expressed also by other words of pronominal character in Czech, e.g. by pronominal adverbs *nikde* (*nowhere*) or *nějak* (*somehow*), or by an indefinite numeral *několik* (*a few*), we can use this grammateme also for the tectogrammatical representation of these words.[10]

As the groups of pronominal words are unproductive classes with (at least to a certain extent) transparent derivational relations not only in Czech, but also in other languages, we believe that similar regularities to those captured in Czech by the indeftype grammateme can be found also elsewhere. However, as it is obvious from the preliminary sketch of several English and German pronouns classified in Table 2,[11] the application of our scheme to other languages will not be straightforward and various subtle differences have to be taken into account. For instance, there is only one negative form *nikdo* corresponding to the t-lemma *kdo* in Czech, therefore the present system provides no means for distinguishing German negative pronouns *niemand* and *niergendjemand*. A new question arises also in the case of English *anybody* when used in negative clauses, which has no counterpart in Czech or German.

	English	English	German	German
T-lemma	*who*	*what*	*wer*	*was*
indefype:				
relat	who	what	wer	was
indef1	somebody	something	jemand	etwas
indef2	-	-	irgendjemand	irgendetwas
indef3	whoever	whatever	-	-
inter	who	what	wer	was
negat	nobody	nothing	niemand	nichts
total1	all	everything	alle	alles
total2	each	each	jeder	jedes

Table 2: Selected English and German pronouns preliminarily classified according to the indeftype grammateme.

5. Implementation

The procedure for assigning grammatemes (and nodetype and sempos) to nodes of tectogrammatical trees was implemented in ntred[12] environment for processing the PDT data. Besides almost 2000 lines of Perl code, we formulated a number of rules for grammateme assignment written in a text file using a special economic notation (roughly 2000 lines again), and numerous lexical resources (e.g. special-purpose list of verbs or adverbs). As we intensively used all information available also at the two 'lower' levels of the PDT (morphological and analytical), most of the annotation could have been done automatically with a highly satisfactory precision.

It should be emphasized that the inter-layer links played a key role in the procedure. As it is clear from Figure 1 (a), it would not be possible to set e.g. the value of the number grammateme of the (already lemmatized) t-node *les* (forest) without having the access to the morphological tag of the corresponding m-layer unit in the given sentence, or to find out that the verb *jít* (to go) is in conditional mood (verbmod=cdn) without knowing that the corresponding a-layer complex verb form subgraph contains the node *by*.

Due to the fact that a lot of effort had been spent on checking and correcting of the inter-layer pointers in PDT 2.0, finally we needed only around 5 man-months of human annotation for solving just the very specific issues (as mentioned at single grammatemes in the previous section).

Now we would like to show a fragment of the above mentioned rules. For a given t-node: if the lemma of the corresponding m-node is *který* (which), the t-node itself is not in the attributive syntactic position and participates in grammatical coreference (i.e., it forms a relative construction), then sempos=n.pron.indef, indeftype=relat, and the values of the grammatemes gender and number are inherited from the coreference antecedent. This rule would be applied on the sentence (1).

To further demonstrate that grammatemes are not just dummy copies of what was already present in the morphological tag of the node, we give two examples:

- Deleted pronouns in subject positions (which must be restored at the t-layer) might inherit their gender and/or number from the agreement with the governing verb (possibly complex verbal form), or from an adjective (if the governor was copula), or from its antecedent (in the sense of textual coreference).

- Future verbal tense in Czech can be realized using simple inflection (perfectives), or auxiliary verb (imperfectives), or prefixing (lexically limited).

The procedure was repeatedly tested on the PDT data, which was extremely important for debugging and further improvements of the procedure. Final version of the procedure was applied to all the available tectogrammatical data (as for its size, recall the second paragraph in Section 2.). This data, enriched with node classification and grammateme annotation, will be included in PDT 2.0 distribution.

Due to the highly structured nature of the task, it is difficult to present the results of the annotation procedure from the quantitative viewpoint. However, at least the distribution of the values of nodetype and sempos are shown in Tables 3 and 4.

[10] The indeftype grammateme is applied to indefinite numerals together with the above-mentioned grammateme numertype – thus only a single t-lemma *kolik* (*how many*) represent words of different nature: e.g. *několik´at´y* (*not the first*), *kolikr´at* (*how many times*) etc.

[11] We chose English and German, because, first, the two languages are the most familiar to the present authors, and second, certain experiments concerning their t-layer have already been performed, see e.g. (Cinková, 2004) or (Kučerová and Žabokrtský, 2002).

[12] `http://ufal.mff.cuni.cz/~pajas`

complex	550947
root	49442
qcomplex	46015
coap	35747
atom	34035
fphr	4549
list	2512
dphr	1282

Table 3: Values of nodetype sorted according to the number of occurences in the PDT 2.0 t-layer data.

n.denot	236926
adj.denot	100877
v	88037
n.pron.def.pers	32903
adj.quant.def	19441
n.denot.neg	18831
n.pron.indef	11343
adv.denot.ngrad.nneg	8947
n.quant.def	7994
adj.pron.def.demon	5746
n.pron.def.demon	4759
adj.pron.indef	3383
adv.pron.indef	3107
adv.pron.def	2928
adj.quant.grad	1865
adv.denot.grad.neg	1315
adv.denot.grad.nneg	1139
adv.denot.ngrad.neg	751
adj.quant.indef	655

Table 4: Detailed values of sempos sorted according to the number of occurences in the PDT 2.0 t-layer data.

6. Conclusion

We believe that two important novel goals have been achieved in the present enterprise:

- We proposed a formal classification of tectogrammatical nodes and described its consequences on the system of grammatemes, and thus the tectogrammatical tree structures become formalizable e.g. by typed feature structures.
- We implemented an automatic and highly-complex procedure for capturing the node classification, the system of grammatemes and derivations, and verified it on large-scale data, namely on the whole tectogrammatical data of PDT 2.0. Thus the results of our work will be soon publicly available.

In the paper we do not compare our achievements with related work, since we are simply not aware of a comparably structured annotation on comparably large data in any other publicly available treebank. For instance, to our knowledge no other treebank attemps at reducing the (semantically redundant) morphological attributes imposed only by agreement, or at specifying verbal tense for a complex verb form as for a whole, or at representing a noun (or a personal pronoun) and the corresponding possessive adjective (or possessive pronoun, respectively) in a unified fashion. However, from the theoretical viewpoint the presented model bears some resemblances with the system of grammemes in the deep-syntactic level of the already mentioned Meaning-Text Theory (Mel'čuk, 1988).

In the near future, we plan to separate the grammatemes that bear the derivational information (such as numertype) from the grammatemes having their direct counterpart in traditional morphological categories. The long-term aim is to describe further types of derivation: we should concentrate on productive types of derivation (diminutive formation, formation of feminine counterparts of agentive nouns etc.). The set of 'derivational' grammatemes will be extended in this way. The next issue is the problem of subclassification of semantic verbs. The challenging topic is also the study of grammatemes in other languages.

Acknowledgements

The research reported in this paper was supported by the projects 1ET101120503, GA-UK 352/2005 and GD201/05/H014. We would also like to thank professors Jarmila Panevová and Eva Hajičová for numerous comments on the draft of the paper.

7. References

Joan L. Bybee. 1985. *Morphology: A study of the relation between meaning and form*. Benjamins, Philadelphia.

Silvie Cinková. 2004. Manuál pro tektogramatickou anotaci angličtiny. Technical report, ÚFAL/CKL MFF UK.

Miloš Dokulil. 1962. *Tvoření slov v češtině I.* Academia, Prague.

Jan Hajič, Eva Hajičová, Petr Pajas, Jarmila Panevová, Petr Sgall, and Barbora Vidová Hladká. 2001. Prague Dependency Treebank 1.0.

Eva Hajičová, Jan Hajič, Barbora Vidová-Hladká, Martin Holub, Petr Pajas, Veronika Kolářová-Řezníčková, and Petr Sgall. 2001. The Current Status of the Prague Dependency Treebank. In *Proceedings of the 5th International Conference on Text, Speech and Dialogue*, pages 11–20, Berlin, Heidelberg, New York. Springer-Verlag.

Eva Hajičová. 1975. *Negace a presupozice ve významové stavbě věty*. Academia, Prague.

Hermann Helbig. 2001. *Die semantische Struktur natürlicher Sprache*. Springer-Verlag, Berlin, Heidelberg, New York.

Ivona Kučerová and Zdeněk Žabokrtský. 2002. Transforming Penn Treebank Phrase Trees into (Praguian) Tectogrammatical Dependency Trees. *Prague Bulletin of Mathematical Linguistics*, (78):77–94.

Igor A. Mel'čuk. 1988. *Dependency Syntax: Theory and Practice*. State University of New York Press.

Jarmila Panevová, Eva Benešová, and Petr Sgall. 1971. *Čas a modalita v češtině*. Univerzita Karlova, Prague.

Carl Pollard and Ivan A. Sag. 1994. *Head-Driven Phrase Structure Grammar*. The University of Chicago Press, Chicago.

Petr Sgall, Eva Hajičová, and Jarmila Panevová. 1986. *The Meaning of the Sentence in Its Semantic and Pragmatic Aspects*. D. Reidel Publishing Company, Dordrecht.

Petr Sgall. 1967. *Generativní popis jazyka a česká deklinace*. Academia, Prague.

Constraint-Based Extract Alignment for Black-Box Evaluation of Extractive Summarization Methods

Jorge Marques Pelizzoni*, Thiago Ianez Carbonel†, Lucia Helena Machado Rino‡

*Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo
Cx. Postal 668 – CEP 13560-970 – São Carlos – SP – Brasil
†Departamento de Letras, ‡Departamento de Computação, Universidade Federal de São Carlos
Cx. Postal 676 – CEP 13565-905 – São Carlos – SP – Brasil
Jorge.Pelizzoni@loria.fr, thiagocarbonel@gmail.com, lucia@dc.ufscar.br

Abstract

The purely extractive approach to Automatic Summarization aims at producing condensed versions of source texts by selecting more salient text spans and juxtaposing them. The resulting summaries, or rather, extracts may be seriously impaired with regard to textuality and even grammaticality, which thus figure as possible criteria for the evaluation of extractive alignment systems. This work was carried out in such an evaluation scenario, aiming at automating at least part of the task. By establishing an alignment between positions in abstracts and those in their respective sources, we allow extracts to "inherit" existent annotation of the sources for subsequent analysis by evaluation tools. The main goal of this paper is to describe a constraint-based model for producing such alignments and its efficient implementation, as well as report on a preliminary evaluation experiment yielding very promising results.

1. Introduction

Automatic Summarization (AS) aims at automatically producing condensed, though still useful versions of (source) texts. One of the most robust, domain-independent and low-cost approaches to AS is the extraction-based one. Given a source text, purely extractive AS systems/methods — the class of interest here — perform a segmentation of the source, select the most salient segments and simply juxtapose them as output. The result is called an *extract*. As extracts are obtained by verbatim reuse of segments already present in the source texts, no wonder extractive summarization usually risks producing meaningless output, even when it actually succeeds in identifying the most relevant fragments of a text. Textuality is the feature at stake here and thus figures as a criterion for the evaluation of extractive summarization artifacts. It is exactly in such an evaluation scenario that we developed the work presented here.

This paper tackles the annotation of linguistic resources of a rather unusual type, namely corpora of extracts, by establishing an alignment between each extract and its respective source text, both viewed as sequences of tokens. In fact, alignments themselves are our annotation objects of interest here, each consisting of a correspondence between (token) positions in an extract and positions in a source text. Not any correspondence, however: whenever an extract token t_1 is aligned with a source token t_2, the real information being conveyed is that t_1 is — or should be — actually t_2 having "survived" an extractive summarization process.

Though not so complex as compared to most language processing tasks, extract alignment poses its difficulties, especially if one takes into account that:

a) the units of extraction might not be exactly sentences, but clauses or other syntactic constituents. See e.g. (Mani & Maybury, 1999) for a survey on work involving various degrees of granularity in extractive AS;

b) in face of (a), the less trivial the segmentation grows, the more prone extractive systems are to introduce errors. Extracts might, therefore, contain incomplete and even non-grammatical structures. For example, one of the systems we experiment with extracted only the underlined part from the sentence[1] "A Russian Soyuz spacecraft, carrying replacement personnel, reached the International Space Station (ISS) two days after leaving Earth." This reused segment appears in the produced extract in the guise of a self-contained sentence, keeping its final comma and being simply juxtaposed with the following reused segment: "Two of the three astronauts that traveled in the Soyuz will spend 175 days in the ISS.";

c) the exact segmentation of the source text by the summarizer is not known to the aligner. This condition is especially interesting as it allows extractive systems to be treated as black boxes.

The interest in extract alignment might not be obvious at first. In fact, it directly depends on whether valuable annotation is already available for the *source corpus*[2] whose projection onto the extract corpus makes some task easier, i.e. at least partially automatable. An alignment provides exactly the means to project annotation automatically, whenever it makes sense, and that is all.

The main purpose of this paper is to describe *Sumalign*, an automatic extract alignment system, as well as provide evidence of its performance obtained in a preliminary experiment. As already mentioned, the scenario in which *Sumalign* was born is the black-box evaluation of extractive summarization methods. In specific, our current focus is on coherence impairment due to anaphoric expressions rendered dangling after extraction. This is a quite troublesome phenomenon in extractive AS and occurs whenever an anaphoric text span is included in the extract and its antecedent is not. Our source corpus is being annotated, among others, with respect to coreference chains and dependencies, and we intend to use alignments to project coreference chain annotation onto extracts and identify malformations in the obtained projections automatically. By means of *Sumalign* and other more trivial tools, we must be able to automate most of the task. Figure 1 exemplifies the application of Sumalign in our

[1] All examples are in Portuguese in the original.

[2] As opposed to *extract corpus*. A corpus of extracts presupposes a corpus of source texts.

scenario. In this figure, source annotation states that "A spaceship Soyuz" is the antecedent that allows the reference of "It" to be resolved. In turn, the alignment allows detecting that the extract includes the latter while omitting the former. By projecting the source annotation onto the extract, it is possible to tell that the original dependency link of "It" is now dangling.

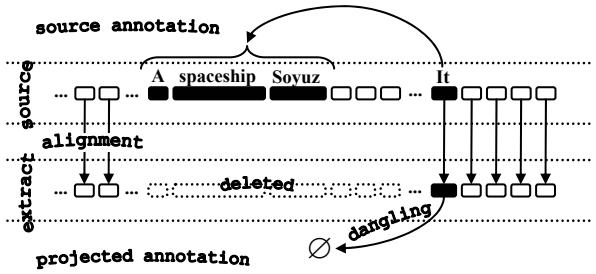

Figure 1: Applying extract alignment

The system tackles extract alignment as an optimization problem. As we shall discuss in Section 3, it is not enough to constrain alignments to ensure token-equality[3] between aligned positions and to preserve order. Given an extract and its source, there may well be a set of alternative alignments satisfying these constraints while just a few of them can be considered as correct output from an aligner though. Fortunately, it was not difficult to conceive a syntax-based penalty function that is likely to be minimized only by the "correct" alignments in a set of alternatives. *Sumalign* thus requires syntactically annotated sources, but keeps its underlying model rather independent of the grammatical framework of choice.

Sumalign employs *Concurrent Constraint Programming* (CCP) as a means to tame search complexity, since the problem could be expressed as a *Constraint Satisfaction Problem* (CSP). Our preliminary experiment suggests that *Sumalign* is a successful application of CCP, achieving outstanding performance and solving all instance problems in (short) polynomial time, with no or very little real resort to search, and with 100% precision.

The paper proceeds as follows. In Section 2, we briefly review some CCP-related concepts in order to enable understanding how *Sumalign* works. Next we develop a formalization of the problem of extract alignment in Section 3 and translate it into a constraint-based model in Section 4, in which we also highlight some implementation details. Our preliminary evaluation experiment is described and discussed in Section 5. Finally, we conclude in Section 6.

2. Concurrent Constraint Programming

In this section we briefly review some concepts of Concurrent Constraint Programming (CCP) in the sense e.g. of Van Roy & Haridi (2004) and Schulte (2002), specifically as implemented by the Mozart system[4] and the Gecode[5] C++ library. This programming paradigm is typically employed in the solution of optimization problems, such as scheduling, timetabling and configuration. Common characteristics to these problems recommending a constraint-based solution are the following:

a) lack of a polynomial-time algorithm that satisfactorily solves them;

b) expressibility as a Constraint Satisfaction Problem (CSP), i.e. as a finite set C of constraints (applications of simple mathematical/logical predicates from a rather restricted library) over a finite set of variables V such that:

- **i.** C must contain a **basic constraint** for each $v \in V$, i.e. a statement of a finite domain containing all the values admissible for v. Usually, the domains are (finite) subsets of $\mathbb{N}$ and $2^{\mathbb{N}}$ — i.e. the admissible values are usually non-negative integers or sets thereof;
- **ii.** any assignment to the variables in V satisfying all constrains in C is a valid solution to the problem.

Non-basic constraints, i.e. those involving two or more variables, are usually simple arithmetic and set-theoretic relations such as $x = 2y+1$, $w = x+y+z$ and $x \in A \cap B$, where all x, y, w, z, A and B are variables.

Any such CSP formulation is said to be a **model** for the problem at hand, and the task of building models is known as **modeling**;

c) the enumeration of all possible candidate solutions/assignments has exponential complexity; whereas

d) the size of the CSP and the cost of building it from input must have polynomial complexity.

Characteristics (a) and (c) are typical of *search problems*, i.e. those to which search-based solving (Russel & Norvig, 1995; Rich & Knight, 1991) is the only currently available approach. When characteristics (a) and (d) are also present, CCP is eligible as a device to help tame search complexity, i.e. avoid combinatorial explosion. The really tricky thing about (a) is that it implies that, before search may start, the CSP must be completely set up. In specific, new variables/constraints cannot be created/posed dynamically during search. This restriction may render the paradigm simply inapplicable or bring about rather complex modeling. Fortunately, neither has been the case for extract alignment.

2.1. Constraint Propagation

CCP is a specialized search-based problem solving technique that exploits the particular properties of a certain way of describing the problem — characteristics (a) and (d) — so that (i) the description of the search space is automatically derived from the CSP and (ii) the search space is *(re)active*, automatically inferring corollaries from what is known or assumed during search and using them to eliminate part of the necessarily failed alternatives *without ever actually trying them out*. This (re)active character — termed **constraint propagation** — is a central concept to CCP and is implemented by "giving life" to the non-basic constraints of a CSP. That is, for every non-basic constraint $p[v_i]$ (where p is a logic predicate; and $[v_i]$, $1 \le i \le n$, $n > 2$, the vector of variables interrelated by this application of p), a concurrent agent — termed **propagator** — is created not only to signal inconsistence of $p[v_i]$ against a complete determination of $[v_i]$, but also *to run a local inference procedure to maintain the consistence of* $p[v_i]$. In other words, the propagator (i) reacts to

[3] A position in an extract is token-equal to a position in its source iff they are occupied by the same token.

[4] http://www.mozart-oz.org

[5] http://www.gecode.org

new basic constraints for any v_i, (ii) calculates corollaries for the remaining v_k, $k \neq i$, in the form of new basic constraints and (iii) broadcasts them to any other interested propagators, probably triggering them into action for even further propagation. The resulting chain reaction incrementally narrows variable domains — and thus the range of solution candidates — and might at times yield a solution with no resort to non-deterministic choice.

Take, for example, the CSP whose constraint set is shown in Figure 2, in which lines 1 to 5 pose the starting basic constraints for each variable and lines 6 to 9 contain four non-basic constraints establishing the order:

$$a < b < c < d < e .$$

Small though it may be, this CSP lets nonetheless show the exponential complexity of its search space, counting on a number of solution candidates equal to:

$$\prod_{v \in V} |dom(v)| = 2 \times 4 \times 4 \times 3 \times 2 = 192 \cong 2.86^{|V|},$$

where $V = \{a, b, c, d, e\}$, and dom(v) denotes the domain in the starting basic constraint for variable v. It is worth noticing that only one of those candidates is satisfactory, namely $\{a = 4, b = 5, c = 6, d = 7, e = 10\}$.

```
1  a ∈ {4, 10}
2  b ∈ {2, 5, 8, 11}
3  c ∈ {3, 6, 9, 12}
4  d ∈ {1, 7, 13}
5  e ∈ {4, 10}
6  a < b
7  b < c
8  c < d
9  d < e
```

Figure 2: Basic (lines 1-5) and non-basic (6-9) constraints in a sample CSP. Line numbers left for convenience

If solved with CCP, the above CSP will normally yield four propagators, one for each non-basic constraint. The local inference procedure of each $x < y$ propagator is very simple and, roughly speaking, proceeds as follows: given the current basic constraints for x and y — $x \in Xs$ and $y \in Ys$ — it infers and poses new basic constraints as follows:

$$x \in \{k : k \in Xs \wedge k < \max(Ys)\},$$

$$y \in \{k : k \in Ys \wedge k > \min(Xs)\}.$$

Naturally, such a propagator fails whenever it infers a basic constraint of the form $v \in \varnothing$.

Even this very primitive, local inference mechanism is capable of solving the CSP in Figure 2 deterministically as a result of the collaboration between propagators. Figure 3 provides a simulation of how this might happen in the form of a "decorated" table, each of whose cells contains (the domain of) the basic constraint valid for a variable (column) at a given moment in time (row). Moreover, each vertical dotted line not only serves as a column separator, but also refers to the propagator it crosses at the top. Arrows and stars refer to propagator activity thus: an arrow from value k_1 of variable x to value k_2 of variable y means that, at the given time, the propagator corresponding to the dotted line crossed by the arrow uses current minimum/maximum k_1 to rule out value k_2 from the domain of y. Stars just draw attention to the fact that dotted lines are being crossed and thus mark which propagators contribute at each time. For example, the leftmost arrow in the first row reads thus: at $t = 0$, the $a < b$ propagator uses 4, the current minimum value for a, to rule out 2 from the domain of b.

Figure 3 only shows one of the possible ways propagation could be carried out for this CSP, as the scheduling of propagators is usually non-deterministic. However, the result of propagation is always deterministic, i.e. the same, granted the *strictly monotonic character* of the corollaries propagators contribute.

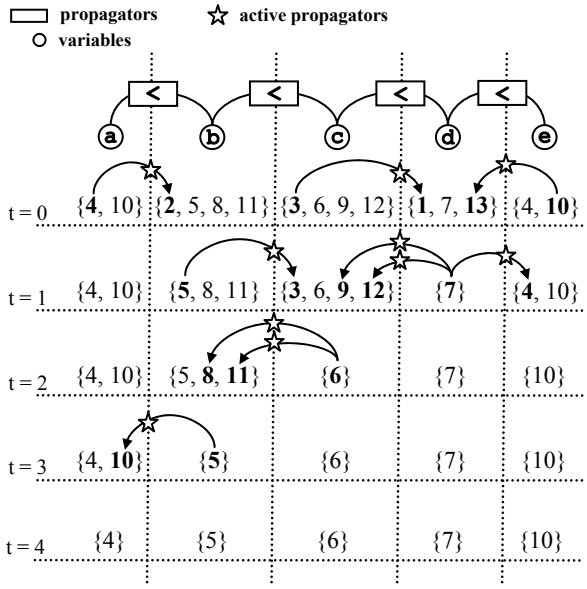

Figure 3: Simulating propagation for the CSP in Figure 2. An arrow from x to y reads "boundary value x rules y out"

2.2. Propagation + Distribution = Completeness

Constraint propagation is not a complete solving method. Its result is not necessarily the determination of all the variables of the CSP. It may as well achieve failure or a state of consistent stability in which, even though the domains of the variables may have been narrowed, there are still undetermined variables and no propagator can contribute new information. The main reason for this is that propagators perform local inference only and derive simple corollaries. In specific, there is no superior entity trying to derive theorems from their declarative semantics, which is exactly what allows propagation to have polynomial complexity and sanction the whole CCP paradigm.

The counterpart of propagation that renders CCP complete is termed **distribution**, which represents the element of search and non-determinism in the paradigm. Whenever consistent stability is achieved and not every variable is determined, it is necessary to induce some instability so as to restart propagation. That is the object of any distribution step, which consists of elaborating an artificial constraint C — said to be a **distribution constraint** — and forking search, assuming now C, now $\neg C$. Recursively interleaving propagation and distribution constitutes a complete and potentially efficient problem solving method. Naturally, some distribution constraints assumed during search may lead to failure, and thus backtracking follows. For completeness therefore, all CCP solutions must specify a **distribution strategy**, i.e. when and how exactly to elaborate distribution constraints given a situation of consistent stability

3. Formalizing Extract Alignment

In this section we formalize the problem at hand. The constraint-based model we describe in the following section closely implements this formalization, which can therefore be regarded at once as a rationale for our model, an opportunity to verify its adequacy and an aid to understand it.

However painstaking an extraction method might be, it is always possible to regard its task as simply deciding what to omit from sources. In other words, every extract is a source that has undergone a series of deletion operations. In specific, the original order between spared tokens should be intact, as well as no new tokens are to be introduced. This point of view strongly bears on the formalization we develop here.

3.1. Text

Extracts and sources are objects of one same formal type: text. For our purposes, it suffices to define a **text over an alphabet Σ** as any function $f: P \to \Sigma$ where P is a finite subset of $\mathbb{N}$. This definition allows us to capture the intuition that a text has a finite toset[6] of positions (the elements of P under the default order for $\mathbb{N}$), each of which is occupied by one single symbol/token (from Σ). As usual, the type of the symbols in an alphabet (whether character strings, numbers, etc.) is immaterial provided they can be tested for equality. For example, the following function:

$$\{2 \to a,\ 3 \to dollar,\ 7 \to a,\ 10 \to week\}$$

is just one of the possible text objects that could be used to represent the real text "a dollar a week".

3.2. Extract and Alignment

Given two texts t_1 and t_2, t_1 is said to be an **extract** of t_2 iff there exists $f: \mathrm{dom}(t_1) \to \mathrm{dom}(t_2)$ — i.e. a function mapping/aligning every position in the extract to/with some position in the source — such that: :

a) for every position $p \in \mathrm{dom}(t_1)$ in the extract, $t_1(p) = t_2(f(p))$. This ensures that the symbol occupying any given position in the abstract also occupies the corresponding aligned position in the source. In other words, token-equality is enforced between every extract position and its aligned source position.
It follows from this condition that $\mathrm{ran}(t_1) \subseteq \mathrm{ran}(t_2)$, where **ran(*f*)** denotes the **range** of function f, i.e. the set $\{f(x): x \in \mathrm{dom}(f)\}$. In our particular case, the image of a text corresponds to the set of all symbols that actually appear in the text. This result thus ensures that all symbols appearing in an extract also appear in the source;

b) for every $p, q \in \mathrm{dom}(t_1)$, $p < q \to f(p) < f(q)$. This ensures that the alignment preserves order and is an injection, i.e. never two different positions in the abstract are aligned with one same position in the source.

Any such f is said to be an **alignment** of t_1 in relation to t_2.

3.3. Deletion

Given a text t and a subset of its positions $P \subseteq \mathrm{dom}(t)$, the **deletion** of P from t, denoted ***t » P***, is itself a text defined thus:

$$t \mathbin{»} P : \big(\mathrm{dom}(t) - P\big) \to \mathrm{ran}(t),$$
$$\big(t \mathbin{»} P\big)(x) = t(x).$$

3.4. Relabeling

Given two texts t_1 and t_2, t_1 is said to be a **relabeling** of t_2 iff there is a bijective alignment of t_1 in relation to t_2.

This definition allows one to abstract away a mere difference in "labeling" of positions between two texts, i.e. the exact numbers that are used to identify positions. Informally speaking, two texts are equivalent in every respect possibly except their exact "labeling" iff one is a relabeling of the other.

3.5. Completeness Theorem

A text t_1 is an extract of t_2 iff there is some subset of positions $P \subseteq \mathrm{dom}(t_2)$ such that t_1 is a relabeling of $t_2 \mathbin{»} P$.

Informally speaking, t_1 is an extract of t_2 iff there is a sequence of deletion operations that turn t_2 into t_1. This theorem is important for completeness as our starting definition of extract had no reference to the way extracts are obtained. In specific, this theorem proves that, whenever a text can be obtained from another solely by deletion, then there is necessarily a way of aligning them. The following proof shows how exactly to do this.

Proof:

- **left-to-right implication:** if t_1 is an extract of t_2, then there is an alignment f of t_1 in relation to t_2. Letting $P = \mathrm{dom}(t_2) - \mathrm{ran}(f)$, we have $\mathrm{dom}(t_2 \mathbin{»} P) = \mathrm{ran}(f)$. It follows that f is also an alignment of t_1 in relation to $t_2 \mathbin{»} P$, and bijective at that, as every alignment is an injection. Hence t_1 is a relabeling of $t_2 \mathbin{»} P$;
- **right-to-left implication:** if there is some subset of positions $P \subseteq \mathrm{dom}(t_2)$ such that t_1 is a relabeling of $t_2 \mathbin{»} P$, then there is a bijective alignment f of t_1 in relation to $t_2 \mathbin{»} P$. This f is also an alignment of t_1 in relation to t_2, hence t_1 is an extract of t_2.

As this theorem and its proof already suggests and would be trivial to prove, each alignment is uniquely associated with a set of spared source positions — $\mathrm{ran}(f)$ — and a set of deleted ones — $P = \mathrm{dom}(t_2) - \mathrm{ran}(f)$. So to speak, alignments tell the story of how the source was shortened into the extract.

3.6. Rationale for Considering Source Structure and Stipulating Penalties

As we shall see in Section 4, our definition of alignment is useful, having a direct effect on the design of our constraint-based model. However, it is too relaxed to characterize the output expected of an aligner fully. In fact, assuming otherwise would imply accepting that the different possible ways to shorten a source into one same extract have the same meaning. Consider the following example:

Source: "… Two of the three astronauts that traveled in the Soyuz will spend 175 days in $\underline{\text{the}}_{\mathbf{A}}$ $\underline{\text{ISS}}_{\mathbf{B}}$. The third one is returning to Earth with the previous occupants of $\underline{\text{the}}_{\mathbf{C}}$ $\underline{\text{ISS}}_{\mathbf{D}}$. The Soyuz makes all trips to $\underline{\text{the}}_{\mathbf{E}}$ $\underline{\text{ISS}}_{\mathbf{F}}$ since the disaster with the space shuttle Columbia in 2003. [end]"
Extract: "… Two of the three astronauts that traveled in the Soyuz will spend 175 days in $\underline{\text{the}}_{\mathbf{X}}$ $\underline{\text{ISS}}_{\mathbf{Y}}$. [end]"

[6] totally ordered set

In the above excerpts, relevant positions are identified with subscript capital letters (A to F, in the source, and X and Y, in the extract) immediately following their respective tokens. As far as unrestrained deletion and alignment are concerned, the extract above could be obtained in a series of ways, since the pair (X,Y) may well be aligned with six different pairs as given by:

$$\big(\{\mathrm{A}\}\times\{\mathrm{B,D,F}\}\big)\cup\big(\{\mathrm{C}\}\times\{\mathrm{D,F}\}\big)\cup\{\mathrm{E,F}\}\,.$$

However, it is clear that only the alignment of (X,Y) with (A,B) — denoted **(X,Y) ⇒ (A,B)** — is acceptable. A formal characterization is in order of what is wrong with the other alignments. Let us start by noticing that all of them would imply very strange behavior on the part of an aligner. Take for instance the hypothesis (X,Y) ⇒ (C,F). It would imply that, for some reason, the aligner decided to delete the tokens at A and B, outputting a sentence that finishes in an uncomplemented "in", and spare the tokens at C (an article, apropos) and F all by themselves, omitting the whole rest of the sentences they belong to. If we put ourselves in the place of the aligner — to which the extractive system is a black box and the exact segmentation it performs on the source is unknown — we will perceive that any argument against such hypotheses requires knowledge of some structure in the source.

In fact, an alignment implies the survival of much more than mere tokens, namely the syntactic phrases they compose. For example, we know that, if position C is present in an extract, then at least part of that specific phrase "the ISS" is also present. Moreover, if position D is not present in turn, then there has been a violation of some sort to the syntactic structure of the source. Therefore, alignments also imply structural violations and, if we provide a formal counterpart to the syntactic structure of a source, it will be possible to detect them.

Notwithstanding, we cannot simply constrain alignments not to commit any such violations, since extractive systems might actually produce some malformed output as exemplified in Section 1. The solution we adopted is to penalize alignments for each violation they imply and accept as output only the least penalized alignment(s) in a set of alternative ones. We conceived a very simple penalty scheme that seems to be sufficient, accumulating *one violation for each phrase missing its governor*, i.e. each implied phrase whose governor is not also implied. For example, alignments splitting the phrase "the ISS", such as (X,Y) ⇒ (C,F), will count one violation for this, no matter whether the grammatical framework of choice considers determiner phrases as governing noun phrases or vice-versa, because the governee inside "the ISS" misses a governor. The same alignment separates "in" from "the" and "ISS". Any such alignment also incurs one more violation irrespective of which token is considered as the head of "in the ISS". If it is "in" (most standard hypothesis), then, although "in" is not directly punished, the governor in "the ISS" is, since it lacks a governor itself. If "in" is a governee in "in the ISS", then it lacks a governor and is punished. In summary, alignments splitting "in the ISS" in any way are penalized, and the least penalized alignment is in effect (X,Y) ⇒ (A,B).

As exemplified above, governors with missing governees — i.e. implied governors with non-implied governees — need not be directly penalized, as any really misaligned governor will leave governees behind as scapegoats to pay for the mistake. All these notions are formalized in the following subsections.

3.7. Attribute-Value Matrix Types

As a notational aid for the following definition, we define a type constructor for a functional flavor of Attribute-Value Matrices (AVMs). Given a *set* of n distinct elements/attributes $\{a_i\}$, $1 \le i \le n$, and a sequence of n (not necessarily distinct) sets/types $\langle T_1, T_2, \ldots, T_n\rangle$, the expression $\{a_1{:}T_1, a_2{:}T_2, \ldots, a_n{:}T_n\}$ denotes the set defined thus:

$$\{a_1:T_1,\ldots,a_n:T_n\}=\left\{f:\left(f:\{a_i,\ldots,a_n\}\to\bigcup_{i=1}^{n}T_i\right)\wedge\left(\forall i\big(1\le i\le n\to f(a_i)\in T_i\big)\right)\right\}.$$

In other words, $\{a_1{:}T_1, \ldots, a_n{:}T_n\}$ is the set/type comprising all and only functions mapping each "attribute" a_i to a value of its respective type T_i. In effect, each of these functions can be regarded as an AVM.

3.8. Phrase and Structure

A **phrase** in a text t is an element of (the type) $phrase(t)$ defined as follows:

$$phrase(t)=\{\mathrm{myself}\}\cup\left\{\begin{array}{l}\mathrm{span}:2^{\mathrm{dom}(t)}\\ \mathrm{governor}:phrase(t),\\ \mathrm{head}:phrase(t)\end{array}\right\},$$

where 'myself', 'span', 'governor' and 'head' denote unique atomic constants. This means that a phrase in a text t is either 'myself', a sort of null value, or an AVM standing for a real phrase in t and containing the following attributes:

a) **span**, the set of positions in t comprised by the phrase;
b) **governor**, whose value is either:
  - **i.** 'myself', when the phrase has no governor and thus can never be penalized, or
  - **ii.** another AVM standing for the governor of the phrase; and
c) **head**, whose value is either:
  - **i.** 'myself', when the span of the phrase coincides with the **head span** of the phrase, i.e. the span of its syntactic head. These phrases would correspond to leaves in syntactic trees and will usually comprise one single word; or
  - **ii.** another AVM standing for a subphrase whose head span coincides with that of the current phrase. By recursively accessing this feature, it should be possible to determine the head span of any phrase.

Any set of phrases $S \subset phrase(t)$ is said to be a **structure** on text t.

It is worth noticing that our definitions of phrase and structure are rather relaxed and allow such abuses as infinite-depth AVMs and head features that never actually lead to a head span. We assume that further restraining the definitions would be to no avail and rely on users to provide the correct instantiations according to the application.

3.9. (Dangling) Phrase Implication

An alignment f implies a phrase p — denoted $\boldsymbol{f \to p}$ — iff any of the following conditions is valid:

a) p = myself, i.e. the null phrase is always implied;
b) $p(\mathrm{head}) = \mathrm{myself} \wedge p(\mathrm{span}) \cap \mathrm{ran}(f) \neq \varnothing$, i.e., when the phrase comprises exactly its head, it is implied iff some vestige of the head can be found among the

spared positions — ran(f) — implied by the alignment; or

c) p(head) ≠ myself ∧ $f \rightarrow p$(head), i.e., otherwise, the phrase is implied iff its head attribute is.

In summary, a phrase is implied iff its head is implied, no matter how many of its other subphrases may be missing.

Finally, in order to identify constituents missing their governors, we define the relation "**implies dangling**" — denoted $\boldsymbol{f \not\rightarrow p}$ — that holds between an alignment f and a phrase p iff f implies p, p has a governor, but the latter is not also implied. Formally, we have the following definition:

$$f \not\rightarrow p \leftrightarrow (f \rightarrow p) \wedge \exists g \begin{pmatrix} g = p(\text{governor}) \wedge \\ \neg(f \rightarrow g) \end{pmatrix}.$$

3.10. Penalty and Restatement

Given a structure S on some text, it is now straightforward to build a function calculating the number of violations to S implied by an alignment thus:

$$penalty(S) = \lambda f.\left|\{p : p \in S \wedge f \not\rightarrow p\}\right|.$$

Finally, the problem of extract alignment can be properly stated in the following fashion: given two texts — *extr* and *src* — and a structure S on *src*, the solution is any alignment of *extr* in relation to *src* that minimizes the function *penalty*(S).

4. Constraint-Based Extract Alignment

In the following subsections, we (i) build a constraint-based model to solve the problem of extract alignment as formalized previously, (ii) describe a search/optimization procedure composing a complete solving method, and (iii) highlight some details of its implementation by *Sumalign*.

4.1. Model Creation

As formalized earlier, our input is composed by two texts — an extract *extr* and a source *src* — and a structure S on *src*. Let us assume without loss of generality that dom(*extr*) is a simple set of consecutive numbers in $\mathbb{N}$ starting at 1. Under this condition, the alignment can be modeled as a vector A of |dom(*extr*)| integer variables. After determination of the whole vector, it will effectively implement a mapping from extract positions (vector indices) to source ones (vector elements).

The starting basic constraints for the elements of A are given thus:

$$\forall i \in arity(A)\left(A_i \in src^{-1}\left[\{extr(i)\}\right]\right), \quad (1)$$

where ***arity*(*V*)** denotes the set of indices in the vector V, and $\boldsymbol{f^{-1}[Y]}$ is the **inverse image** of set Y under f, defined thus:

$$f^{-1}[Y] = \{x \in \text{dom}(f) : f(x) \in Y\}.$$

Given a token t, the expression $src^{-1}[\{t\}]$ yields a set with all source positions occupied by t. Therefore, for a given extract position i, $src^{-1}[\{extr(i)\}]$ yields all source positions occupied by the same token found at i. Constraint (1) is thus enough to ensure token-equality between aligned positions (Section 3.2, item a). It is worth remarking that $src^{-1}[Y]$ returns determined sets at model creation, and hence (1) is actually a source of basic constraints.

In order to ensure that A is indeed an alignment, it suffices to constrain it to preserve order (3.2.b) thus:

$$\forall i \in arity(A) - \{1\}\left(\langle A_{i-1} < A_i \rangle\right). \quad (2)$$

In constraint (2), as well as in all the others, we attempt to make clear which (sub)constraints are really non-basic, i.e. involving two or more undetermined variables at model creation and corresponding to the creation of a propagator. Consequently, we write $\langle C \rangle$, such as $\langle A_{i-1} < A_i \rangle$ in (2), only to identify C as a non-basic constraint. This is not current CCP practice, but might be helpful to readers not as yet acquainted with CCP.

Constraints (1) and (2) usually yield very strong propagation for extract alignment, comparable to that simulated in Figure 3. In fact, the CSPs resulting from the expansion of these constraints are very similar to the one in Figure 2. Section 5 details the impressive results these two constraints alone were observed to produce in our experiment.

It is in order now to tackle the application of the penalty function. To this end, it is first necessary to model phrase implication, whose first requirement is the declaration of a set variable *Ran* modeling the range of the alignment (3.9.b), in the following fashion:

$$Ran = \left\langle \bigcup_{i=1}^{ub(A)} X : toSingleton(A_i, X) \right\rangle, \quad (3)$$

where *ub*(V) denotes the maximum index for vector V and *toSingleton*(*x, X*) is a predicate constraining a set variable X to be a singleton containing a integer variable x defined thus:

$$toSingleton(x, X) \leftrightarrow x \in \mathbb{N} \wedge X \in 2^{\mathbb{N}} \wedge |X| = 1 \wedge \langle x \in X \rangle.$$

Variable *Ran* is therefore constrained to be the union of all A_i taken as singletons, which is exactly the range of the alignment that A is supposed to model.

As our aim is not directly to impose (dangling) phrase implication, but rather detect (count) it, we apply the **reification** technique, rather routine in CCP. A reified version of a constraint C — denoted $[\![C]\!]$ — yields an integer-encoded Boolean variable constrained to coincide with the (possibly yet to be determined) truth-value of C. For simplicity, we assume that this notation obviates writing ⟨C⟩. The encoding is standard: 0 for false and 1 for true. A reified version of phrase implication is defined thus:

$$impl(p) = \begin{cases} 1, & p = \text{myself} \\ impl(p(\text{head})), & p(\text{head}) \neq \text{myself} \\ [\![\, |\langle p(span) \cap Ran \rangle| > 0 \,]\!], & \text{otherwise.} \end{cases} \quad (4)$$

Dangling phrase implication is defined below in (5), which uses the logic operators for conjunction and negation — $[\![\wedge]\!]$ and $[\![\neg]\!]$, respectively — in reified versions, i.e. defined for integer-encoded Boolean variables and yielding the truth-value of the expression itself also as a variable. Application of these operators thus creates propagators.

$$dang(p) = \begin{cases} 0, & p = \text{myself} \\ \langle impl(p)[\![\wedge]\!]\langle[\![\neg]\!] impl(p(\text{governor}))\rangle\rangle, & \text{else.} \end{cases} \quad (5)$$

Finally, the application of the penalty function can be modeled by a variable *Penalty* constrained to be the summation of *dang*(p) for every p in the input structure S, as shown in constraint (6). As *dang*(p) yields 1 for phrases implied dangling or 0 otherwise, the summation correctly equals the number of phrases implied dangling by the modeled alignment.

$$Penalty = \left\langle \sum_{p \in S} dang(p) \right\rangle \quad (6)$$

4.2. Searching for a Minimum

Given such a model as described in the previous section and according to the general CCP paradigm, solving is performed by interleaving propagation cycles with distribution steps. In this section we focus on the distribution strategy we adopted.

Evidence (cf. Section 5) suggests (i) that our model yields excellent propagation for extract alignment and (ii) the simplest optimization technique in CCP is enough to solve the problem adequately. It consists of distributing first over the variable to be minimized/maximized naïvely, i.e. optimistically assuming the best possible value according to the current basic constraints. If failure ensues in the following propagation and distribution (over other interest variables) steps, the second best value is assumed, search is resumed and so on until the first solution is found, which must thus be an optimum. Such a process can be very inefficient in some scenarios, where less *eager* optimization strategies are recommended (e.g. branch-and-bound). In fact, we actually experimented with alternative strategies, but their discretion not to assume best values right away proved wasteful.

Therefore, the first *distribution* constraint we apply is simply:

$$Penalty = \min\big(\mathrm{dom}(Penalty)\big) \quad (7)$$

where **dom(*V*)** denotes the domain of variable V according to the current basic constraints. Next, we distribute over the vector A using a first-fail strategy. While there are undetermined variables in A, this strategy does the following:

a) select one undetermined A_i whose domain size $|\mathrm{dom}(A_i)|$ is minimum, i.e. one of those offering fewest alternative values. This represents the "first-fail" character of the strategy, which prioritizes distribution over more constrained variables as an attempt to induce failure as early as possible, avoiding depth in search;

b) assume the distribution constraint $A_i = \min(\mathrm{dom}(A_i))$, while creating a backtrack point for $A_i > \min(\mathrm{dom}(A_i))$;

c) wait for the resulting propagation to cease and iterate.

As a result of distributing over A, either a total determination of A — i.e. an alignment — is produced or failure ensues, which means that the current value assumed for *Penalty* is too strict and the next best value should be tried. However, in our experiment (cf. Section 5), due to the remarkable side-effects of (7), distribution over A never actually happened. At all times, propagation just after model creation — i.e. before (7) is first imposed — was so strong as to ensure that the first min(dom(*Penalty*)) coincided with the actual best value for *Penalty*; and, once this value had been assumed, propagation alone was able to determine all variables of interest.

In order to show how this might happen, let us resume the "in the ISS" case in Section 3.6. For the referred extract (which 3.6 shows only partially, being actually one of the instances in our experiment), the starting propagation promoted by constraints (1) and (2) is able to determine the alignment for several extract positions, which allows (4) to determine implication (or not) of a great number of phrases. Three among the (already known to be) implied phrases share one same governor that, in turn, is known *not* to be implied. Hence (5) is able to determine that these three phrases are dangling, which enables (6) to infer that *Penalty* cannot be less than 3. This value is assumed in (7), which makes (6) impose $dang(p) = 0$ for all phrases p except those three ones. As the governors of phrases $\overline{\mathrm{CD}}$ (i.e. "the$_C$ ISS$_D$") and $\overline{\mathrm{EF}}$ are known not to be implied from the very start, the following happens to both $p \in \{\overline{\mathrm{CD}}, \overline{\mathrm{EF}}\}$:

a) constraint (5) infers that the only way to maintain $dang(p) = 0$ is to impose $impl(p) = 0$, triggering (4) to forbid the head span of p to have any intersection with the range of the alignment;

b) constraint (3) is then able to forbid all A_i to assume any value in the head span of p, which means that no alignment is possible anymore with the head of p;

c) analogously, the governee inside p is ruled out for alignment, since its governor (the head of p) is now known not to be implied.

Finally, extract positions X and Y have their alignment candidates short-listed to exactly A and B and thus no real choice is required.

4.3. Implementation

We directly implemented the CCP solution to extract alignment described here in *Sumalign*, an application entirely developed by means of the Mozart programming system. *Sumalign* reads sources and their (syntactic) structure in an XML-based format described in (Vieira at al., 2003) and (Gasperin et al., 2003). Extracts are read as raw text files, though necessarily tokenized.

It is worth highlighting a few implementation details conferring *Sumalign* a performance boost and at times representing a slight deviation from the original model, namely:

a) punctuation includes very frequent tokens such as commas and full stops and, even under constraints (1) and (2), often allows a great number of alternative alignments. Take, for example, a source text comprising several sentences and an extract composed by the first sentence and the last one only. The full stop at the end of the first sentence in the extract might potentially be aligned with that of any of the deleted sentences. At the same time, source structure and other annotation sometimes not even make reference to punctuation tokens. Above all, in our specific application, the alignment of punctuation is irrelevant. Therefore, *Sumalign* allows specifying a set of **free punctuation tokens**. Whenever an A_i in the model corresponds to an extract position occupied by one of these tokens, this variable is never selected for distribution. In the end, if propagation alone was not able to fully determine such variables, their alignments are given as a sets of alternative values;

b) before structure files are read, *Sumalign* first poses constraints (1) and (2) and waits for stability. At this point, the upper-bound value of *Ran* (the range of the alignment) is usually considerably narrowed and is used to avoid storing phrases which are already known not to be implied. This requires adding a value 'missing' to $phrase(t)$ and defining $impl(\mathrm{missing}) = 0$;

c) function *impl* in (4) is implemented with memoization to avoid creating new propagators at every invocation. For the same reason, a memoized negated version $\neg impl$ is also supplied.

5. Preliminary Evaluation

As a first evaluation experiment, a small test corpus was submitted to *Sumalign* with the following features: a **source corpus** comprising 45 short newspaper articles amounting to 5507 tokens and an **extract corpus** amounting to 2449 tokens with an average compression rate of 55%. The structure files comprise 4021 phrases, 507 of which are not actually stored due to the mechanism in (4.3.b), a gain margin that might be more expressive were the compression rate greater. The extracts were automatically generated by pipelining the rhetoric analyzer DiZer (Pardo, Nunes & Rino, 2004) and the (extractive) rhetoric summarizer Rhesuma (Seno & Rino, 2005). After alignment, the extracts were found to commit 99 structural violations. It is worth reminding that all such violations are due exclusively to irregular segmentation by the summarizer, which the aligning process only exposes. In fact, all generated alignments were found to be 100% precise. Further details on the test corpus are available in Table 1.

In our experiment, three dimensions were measured for each problem instance, namely:

a) search space size, taken as the product $|\mathrm{dom}(A_1)| \times \ldots \times |\mathrm{dom}(A_n)|$, i.e. the number of possible assignments to A according to the basic constraints valid at a given moment. For each instance, size is probed twice: on stability after constraint (1) is posed and on stability after (2). This is done to sense how well these two constraints alone propagate with no resort whatsoever to search. In Table 1, size data is given in order of magnitudes, as the numbers involved are very large and vary wildly;

b) search depth, taken as the number of choices made and failures observed during search;

c) turnaround time to find a solution, including reading and processing of input files, on a computer with a 3Ghz Pentium4 CPU and 1.25Gb RAM running Windows XP.

		total	mean	max	min
corpus	source tokens	5507	122	215	62
	extract tokens	2449	54.4	113	22
	phrases	4021	89.4	172	35
	stored phrases	3514	78.1	159	29
	violations	99	2.2	11	0
performance	size after (1)*	438	9.73	29	1
	size after (2)*	0			
	depth (total/failures)	11/0	0.24/0	1/0	0/0
	turnaround (s)	15.2	0.34	1.43	0.05
	precision	100%			

Table 1: Summary of evaluation results. Items marked with an asterisk (∗) are measured in orders of magnitude

Table 1 summarizes the results of the experiment, which sets regular punctuation free by means of the mechanism in (4.3.a). In orders of magnitude, we observed size fall from an average 9.73 on stability after constraint (1) to 0 after (2), invariably. In specific, at this stage slightly over 75% of the cases were completely determined (size = 1), and the maximum observed size was 8. That explains the several instances whose search depth is 0, since a completely determined alignment vector also implies a determined *Penalty* variable and thus no distribution at all is performed. The remaining instances have depth equal to 1, corresponding to the first time (7) was posed, which was enough to produce a solution.

Time performance was also very good, the whole experiment being carried out in little more than 15 seconds. Although all these results are usually regarded as indicative of a very successful application of CCP, further experimentation with problem instance parameters is necessary before more definitive conclusions can be drawn. In specific, it would be interesting to investigate the impact of source length and compression rate on performance.

6. Conclusions and Future Work

In this paper we tackled the alignment of extracts with their respective sources, which adds value to available source annotation whose projection onto abstracts serves a purpose. One possible such purpose is the evaluation of extractive summarization methods for the criterion of textuality.

We present a constraint-based solution and the promising results of a preliminary evaluation experiment. A more representative corpus is under construction that will provide not only a better platform for experimentation but also the opportunity to apply *Sumalign* in a real-case scenario.

Acknowledgements

We would like to thank Brazilian funding agencies CAPES and CNPq for supporting this work.

7. References

Gasperin, C., Vieira, R., Goulart, R., and Quaresma, P. (2003). Extracting XML chunks from Portuguese Corpora. In *Proceedings of the Workshop on Traitement automatique des langues minoritaires*. Batz-sur-Mer: ATALA, pp. 223-232.

Mani, I., and Maybury, M.T. (eds.) (1999). *Advances in Automatic Text Summarization*. MIT Press.

Pardo, T.A.S., Nunes, M.G.V., Rino, L.H.M. (2004). DiZer: An Automatic Discourse Analyzer for Brazilian Portuguese. In *Proceedings of the 17th Brazilian Symposium on Artificial Intelligence – SBIA*. Springer, pp. 224-234.

Rich, E. and Knight, K. (1991). *Artificial Intelligence*, 2nd edition, McGraw-Hill.

Russell, S. and Norvig, P. (1995). *Artficial Intelligence: A Modern Approach*, Prentice-Hall, 1995.

Schulte, C. (2002). *Programming Constraint Services: High-Level Programming of Standard and New Constraint Services*, Lecture Notes in Computer Science Series, Springer-Verlag.

Seno, E.R.M., and Rino, L.H.M. (2005). Co-referential chaining for coherent summaries through rhetorical and linguistic modeling. In *Proceedings of the Workshop on Crossing Barriers in Text Summarization Research*.

Vieira, R., Gasperin, C., Goulart, R., and Salmon-Alt, S. (2003). From concrete to virtual annotation markup language: the case of COMMOn-REFs. In *Prooceedings of ACL 2003 Workshop on Linguistic Annotation: Getting the Model Right*. Sapporo: ACL, pp. 6-13.

Van Roy, P. and Haridi, S. (2004). *Concepts, Techniques, and Models of Computer Programming*, MIT Press.

Semi-Automatic Phonological Annotations of Speech by Grammatical Inference

Robert Kelly and Julie Carson-Berndsen

UCD School of Computer Science and Informatics
Belfield, Dublin 4, Ireland
{robert.kelly, julie.berndsen}@ucd.ie

Abstract

This paper describes a technique for automatically generating multiple levels of linguistic annotation for a corpus of speech utterances. Using a training corpus of multilevel annotations, a corresponding finite-state representation is automatically constructed by grammatical inference. This finite-state description is then employed as a knowledge component to automatically generate a new multilevel annotation for an unseen utterance. The approach is evaluated on a small corpus of English speech utterances annotated over four levels of phonological description.

1 Introduction

This paper addresses the problem of automatically generating multilevel annotations for a corpus of speech utterances. A multilevel annotation of a speech utterance consists of a number of independent parallel tiers of time-aligned annotation tokens where each tier represents an annotation of the utterance at some level of linguistic description. The levels of annotation provided by a particular corpus will vary depending on the purpose for which the corpus is intended but typically range from a phonetic labelling of segments, such as the phonetic transcriptions provided with TIMIT (Garofolo et al., 1993), to an annotation level labelling some semantic category, e.g. speech acts (Leech and Weisser, 2003).

Multilevel annotations of speech are now recognised as valuable linguistic resources. In particular, the different levels of annotation provided with a corpus can be used as a "Gold Standard" for evaluating the hypotheses generated by automatic speech recognition systems. Also, annotated corpora are used in unit-selection approaches to speech synthesis. However, the process of manually producing high quality linguistic annotations of speech utterances is time consuming and requires much effort and linguistic expertise. While many computational tools have been developed in an attempt to ease the annotation burden, e.g. Praat (Boersma and Weenik, 2000), Emu (Cassidy and Harrington, 2001), and MATE (McKelvie et al., 2001), the effort required to produce any significant amount of annotated data by human annotators is still immense[1]. Also, such tools provide limited, if any, automatic annotation and/or verification support. On the other hand, a number of annotation tools have been developed which automatically annotate speech corpora at different levels of linguistic annotation (see, for example, the inventory of different automatic annotators used by Leidner et al. (2003)), however such tools generally operate at or above the level of the word and typically generate an annotation at only one level of linguistic description.

In this paper, a *semi-automatic* approach to linguistic annotation is proposed based on grammatical inference techniques. Given a training corpus of manually produced multilevel annotations and a new utterance to be annotated, a generalised finite-state transducer is constructed from the training corpus using a regular inference algorithm. The constructed transducer is then used as a knowledge resource to automatically generate a multilevel annotation for the new utterance[2]. The accuracy of the automatically generated annotations cannot be guaranteed since the content of a multilevel annotation will vary with respect to a number of sources; in particular, the annotator and the speaker. Therefore, automatically generated annotations serve only as a first attempt which of course must be verified by a human annotator. Thus, while it is recognised that this annotation strategy does not completely eliminate the need for manual annotation, an evaluation based on a small training corpus of multilevel annotations for English speech utterances shows that a great deal of tedious annotation marking and labelling can potentially be performed in an automatic fashion.

The remainder of this paper is organised as follows. Section 2 introduces the Ostia transducer inference algorithm. Section 3 presents a scheme for encoding a training corpus of multilevel annotations as a positive sample of input-output pairs. Such a sample can then be supplied to Ostia to infer a corresponding finite-state transducer. Section 4 discusses how an inferred transducer can be used to automatically generate a new multilevel annotation. The quality of these automatically generated annotations are evaluated in section 5 in the context of a small corpus of English speech utterances annotated over four levels of phonological description. Finally, section 6 presents concluding remarks and some directions for future work.

2 Constructing Annotation Transducers

The basis of the annotation strategy discussed in this paper is the construction of a finite-state transducer from a sup-

[1] For example, Barras et al. (2001, §. 4.3) report that the total time required to produce annotations on four levels (orthographic, speech turn, topic, background noise) for one hour of broadcast news, including verification, using the Transcriber tool took around 50 hours for a single human annotator.

[2] This is somewhat similar to the approach adopted by Roche and Schabes (1995) for constructing a part-of-speech tagger. However, in this paper the annotation problem is generalised to multilevel annotations over an arbitrary number of different linguistic levels.

plied training sample of multilevel annotations. A finite-state transducer is a generalisation of a finite-state automaton whereby a number (possibly zero) of output symbols are emitted whenever a state-transition is effected on an input symbol. Thus, a finite-state transducer defines a regular mapping from strings of input symbols to strings of output symbols.

Ostia, the *Onward Subsequential Transducer Inference Algorithm* (Oncina et al., 1993), is a grammatical inference technique (cf. Fu and Booth (1975)) that identifies a *subsequential* finite-state transducer from a *positive* sample of input-output pairs (i.e., no *negative* examples of the required transduction are assumed to be available). A subsequential transducer is a restriction of a finite-state transducer requiring that it's input tape be deterministic, that is, for each combination of state and input symbol a unique next state is defined, but allowing a number (possibly zero) of final output symbols to be emitted when the entire input string has been consumed.

Given a sample S of input-output pairs, Ostia infers a transducer from S in two stages. Firstly, a tree subsequential transducer is constructed from S. The tree transducer describes exactly the pairs in the sample S and is constructed such that the consecutive symbols of the output strings which appear in S are assigned to those state-transitions which are as "close" as possible to the unique initial state of the tree without causing a conflict in the input-output mappings defined in S. This is illustrated in Figure 1 which shows an initial tree transducer constructed from a small sample of phoneme-CV pairs extracted from the LeaP annotated speech corpus (Gut, 2004) (cf. section 5.1)[3].

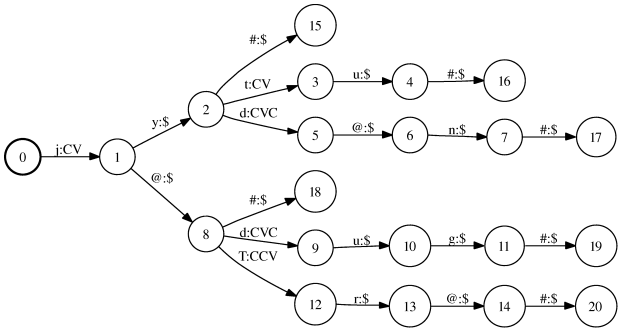

Figure 1: Initial subsequential transducer constructed by Ostia from the sample $\{(jy, CV), (jytu, CVCV), (jyd@n, CVCVC), (j@, CV), (j@dug, CVCVC), (j@Tr@, CVCCV)\}$.

Following the construction of the tree subsequential transducer Ostia identifies a solution subsequential transducer by continually *merging* pairs of compatible states. A pair of states is compatible if merging them results in a new transducer which is (or can be made) subsequential. For example, in Figure 1 the states 2 and 8 can be merged since the result can be made subsequential (by subsequently merging the state pairs 5 and 9 and 15 and 18). The subsequential transducer resulting from this merge is illustrated in Figure 2.

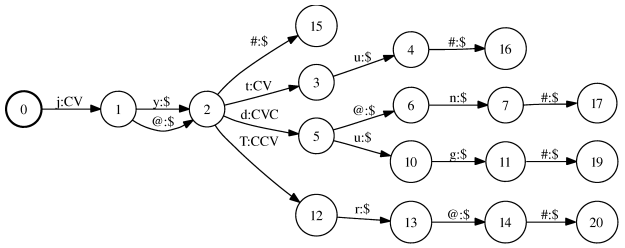

Figure 2: Subsequential transducer resulting from the merge of states 2 and 8 in Figure 1.

The solution transducer identified by Ostia is guaranteed to describe the mappings defined in S but will typically generalise over S to define a larger class of transductions. Thus, applying Ostia to a training corpus of multilevel annotations will typically yield a transducer which generalises from the annotations seen in the training corpus to describe additional (but similar) annotation structures. Therefore, an inferred transducer can potentially be used to generate a new annotation for an utterance which was not seen in the supplied training corpus. However, in order to use Ostia to infer a transducer from a training corpus of multilevel annotations, the corpus must first be encoded as a positive sample of input-output pairs. This aspect of the transducer inference approach to automatic annotation generation is discussed in detail in the following section.

3 Encoding Multilevel Annotations

This section presents an encoding scheme which allows a training corpus of multilevel annotations over an arbitrary number of different linguistic levels to be encoded as a positive sample of input-output pairs for use with Ostia. The annotation strategy described here assumes that each level of annotation in a training corpus consists of a continuous sequence of annotation tokens where each token is labelled with the name of the linguistic object it represents and is time-aligned with the original speech signal by augmenting the token with a temporal start and end time referring to a start and end point for the token with respect to the corresponding speech signal[4]. A multilevel annotation of this type over four levels of linguistic description is illustrated in Figure 3[5]. This shows a small portion of a manually pro-

[3]The final output emissions are modelled here by defining a special end-of-string transition labelled with # on the input tape and the corresponding final output symbols on the output tape. When an input string is consumed by the transducer it makes an end-of-string transition from the current state (if one exists). For this paper, the symbol $ represents the empty string while : is used to separate the input and output constituents of a state-transition label in a transducer.

[4]Note that annotations of this type necessarily exclude those levels of annotation describing instants, for example, tonal or pitch markers. However the encoding scheme presented here can be easily extended to cater for these types of annotation since the notion of temporal overlap between instants and between instants and intervals is well defined (cf. (Bird and Klein, 1990)). Extending the encoding to these levels is left to future work however.

[5]All annotations in this paper are illustrated using Praat (Boersma and Weenik, 2000).

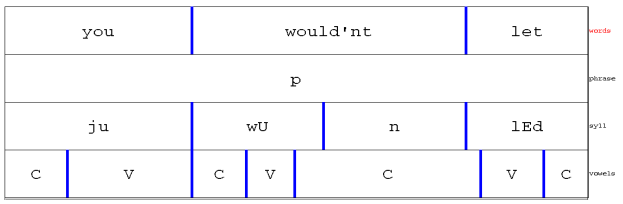

Figure 3: Portion of a speech utterance annotated at four levels of linguistic description; the word level, the phrase level, the syllable level, and the vowels (skeletal CV) level.

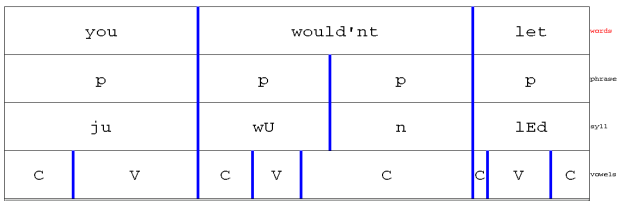

Figure 4: Segmentation imposed on the multilevel annotation of Figure 3 when encoded with the word level as domain annotation and the syllable level as input annotation.

duced annotation for an utterance in the LeaP corpus.

Multilevel annotations of the type described above can be encoded as a positive sample of input-output pairs by examining the overlap relations which occur between the different levels of annotation. The encoding requires that one of the levels of annotation be designated as the **domain annotation** while a second and distinct level be designated as the **input annotation**. The remaining (i.e., non-domain and non-input) levels of annotation are referred to here as the *output annotations*.

The encoding works by constructing an input-output pair (d_i, d_o) for each token d of the designated domain level annotation. The input string d_i is constructed by concatenating (following the temporal order of the annotation tokens) the labels of those tokens on the input annotation which *temporally overlap* d[6]. The corresponding output string d_o is constructed in two stages. Firstly, an individual output string is constructed for each output annotation by concatenating (again, following the temporal order of the annotation tokens) the labels of those tokens on the output annotation which temporally overlap d. Secondly, the individual output strings are themselves concatenated together according to some predefined ordering on output annotations; usually the top-to-bottom order as they appear in the multilevel annotation. Each of the individual output strings involved in this concatenation is separated from one other by a special delimiter symbol. This makes explicit the contribution of each output annotation to a given output string.

The encoding of multilevel annotations presented above essentially segments each annotation level using the tokens of the domain and input annotations in order to construct the corresponding sample of input-output pairs. Thus, applying the encoding scheme to the multilevel annotation illustrated in Figure 3, for example, using the word level as the domain annotation and the syllable level as the input annotation, imposes the segmentation illustrated in Figure 4. Using "-" as the special delimiter symbol separating the individual output strings (constructed from the phrase and vowels level annotations here), and using "." to separate concatenated symbols (required since multicharacter symbols occur in the syllable level annotations), the encoding constructs the following sample of input-output pairs from the multilevel annotation shown in Figure 3.

$$\{(ju, p - C.V), (wU.n, p - C.V.C), (lEd, p - C.V.C)\}$$

[6] A token t_1 with start time s_1 and end time e_1 overlaps a token t_2 with start time s_2 and end time e_2 iff $e_1 > s_2$ and $e_2 > s_1$.

The correspondences between annotation tokens which are captured by these input-output pairs can be easily seen from the segmentation illustrated in Figure 4.

The encoding scheme presented in this section can be used to encode an entire corpus (or part thereof) of multilevel annotations over n levels ($n > 1$) as a positive sample of input-output pairs. If a new utterance is annotated at one of the levels of linguistic description given by the training corpus then encode the training corpus using this level as the input annotation and use Ostia to infer a corresponding generalised transducer[7] A new multilevel annotation can then be generated by using the inferred transducer to parse the tokens of the manual annotation. This is discussed in detail the following section.

4 Generating Multilevel Annotations

In order to generate a new multilevel annotation for an unseen utterance using an inferred transducer, the utterance must first be (manually) annotated at the linguistic level described by the input tape of the transducer. The inferred transducer is then used to parse the sequence of token labels of this supplied annotation level; dividing the sequence into a number of well-formed token combinations. Each well-formed token combination is then used to construct a new annotation token for the linguistic level corresponding to the domain of the transducer. The start time and end time of each new domain-level token is derived from the start time of the first token and the end time of the last token respectively in the corresponding well-formed combination. Finally, the label of each new domain-level token is obtained by concatenating the individual labels in the corresponding well-formed combination.

The remaining levels of annotation are constructed by observing the outputs which are produced as the transducer parses the supplied sequence of input-level annotation tokens. Thus, as each well-formed combination is identified, a corresponding sequence of outputs is generated. Each output will describe a (possibly empty) concatenation of annotation tokens over a number of levels (where each level

[7] The choice of domain annotation in this case is arbitrary. Note, however, that the corresponding inferred transducer will describe well-formed combinations of input annotation tokens with respect to the domain annotation (i.e., the domain annotation determines the *linguistic domain* of the inferred transducer). Thus, a coarser level of annotation (e.g., phrase or word) may be a more appropriate domain annotation compared with a finer-grained level of annotation (e.g., syllable or segment).

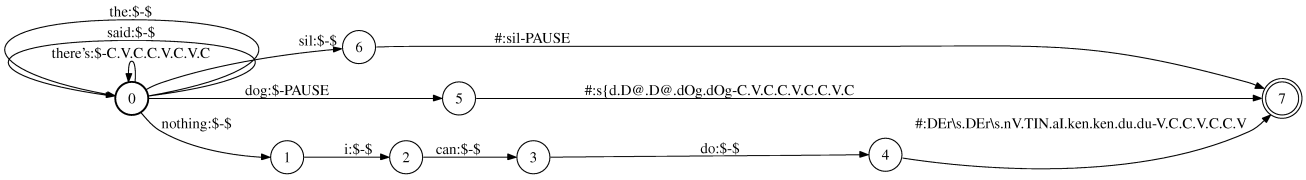

Figure 5: Portion of a transducer inferred using Ostia. The input tape accepts strings of word level tokens while the outputs are tokens over the syllable and skeletal (CV) level. Note that \$ represents the empty string, . represents concatenation, − delimits annotation levels on outputs and # represents the special end-of-string marker (cf. section 2).

is separated from the next by the special delimiter symbol). Given a sequence of outputs corresponding to some well-formed combination, an annotation is constructed for each level as follows. Firstly, the concatenations of symbols for each of the individual levels in each output is extracted. The extracted concatenations of symbols are then themselves concatenated together in sequence. A number of identical symbols may occur in sequence in the resulting concatenation (this occurs as a result of the segmentation imposed by the encoding scheme presented in section 3 which may split a single annotation token across multiple transitions in the corresponding inferred transducer) and these are *smoothed* into a single token. Each individual symbol which remains following smoothing is used to construct an annotation token for the current level with a label equal to that symbol. Augmenting these newly constructed annotation tokens with meaningful start and end times is difficult since the input-output mappings defined by inferred transducers may not preserve linguistic correspondences between levels. In particular, transducers inferred by Ostia may delay producing non-empty output until a parse is complete (cf. Figure 5). Here, *approximate* start and end times are assigned to each token which allots an equal portion of the duration of the corresponding well-formed combination of input-level tokens.

```
35.624217 35.782928 there's
35.782928 36.093914 nothing
36.093914 36.269782 i
36.269782 36.417769 can
36.417769 36.582913 do
36.582913 36.771649 said
36.771649 36.886152 the
36.886152 37.190704 dog
37.190704 37.619651 sil
```

Figure 6: Manual annotation of new utterance at the word level.

In order to illustrate this automatic annotation process Figure 5 shows a portion of a transducer constructed using Ostia from a corpus of multilevel annotations over four levels (cf. section 5.1 below). The domain of the transducer describes phrase level annotations, the input tape describes word tokens, and the output tape describes syllable and skeletal (CV) annotation levels. This portion of the transducer can be used to generate a multilevel annotation over four levels for a new utterance which is manually annotated at the word level. Thus, supplied with the manual word level annotation of Figure 6, the individual tokens of this annotation are parsed into the following three well-formed combinations.

(1) there's nothing i can do (2) said the dog (3) sil

which are used to derive the following phrase level annotation (where the start and end times of each token are derived from the word tokens of Figure 6 and an underscore "_" is used to separate the constituent word labels).

```
35.624217 36.582913 there's_nothing_i_can_do
36.582913 37.190704 said_the_dog
37.190704 37.619651 sil
```

Considering the well-formed combination "there's nothing i can do" a syllable level annotation is generated by examining the corresponding sequence of outputs

\$-C.V.C.C.V.C.V.C \$-\$ \$-\$ \$-\$ \$-\$
DEr\s.DEr\s.nV.TIN.aI.ken.ken.du.du-V.C.C.V.C.C.V

Extracting the syllable level symbols from these outputs and concatenating (smoothing where required) yields the following.

DEr\s.nV.TIN.aI.ken.du

The duration of the well-formed combination "there's nothing i can do" is 0.958696 ($= 36.582913 - 35.624217$). Therefore, a duration of ≈ 0.159783 ($= 0.958696/6$) is alloted to each of the six syllables in the generated annotation yielding the following syllable level annotation.

```
35.624217 35.784 DEr\s
35.784 35.943783 nV
35.943783 36.103566 TIN
36.103566 36.263349 aI
36.263349 36.423132 ken
36.423132 36.582913 du
```

A skeletal annotation for the well-formed combination "there's nothing i can do" can be automatically generated in a similar manner.

Again, note that the start and end times assigned to the syllable and skeletal level tokens by the automatic annotation are approximations only and will need to be verified by a human annotator. In addition, the annotation tokens which are generated are subject to human verification since variation may occur between speakers and/or annotators. Also,

some levels of annotation, e.g. semantic, may not be easily predictable from the content of other levels, e.g. phonetic, leading to errors in automatically generated annotation tokens. Therefore, the grammatical inference approach discussed here embodies a *semi-automatic* approach to generating linguistic annotation since automatically generated annotations are subject to verification by human annotators. Thus, while it is recognised that this annotation strategy does not eliminate the need for manual annotators, a great deal of tedious annotation work (e.g., token marking and labelling) can potentially be performed in an automatic fashion; alleviating at least some of the burden of annotating a speech utterance at multiple levels. The following section presents an experimental evaluation of the approach on a small corpus of English speech utterances annotated over four levels of phonological description.

5 Experimental Evaluation

5.1 The LeaP Corpus

The LeaP corpus (Gut, 2004) was collected in order to investigate the acquisition of prosody by non-native speakers of German and English and consists of both read and free speech manually annotated over six levels of annotation[8]. The manual annotations were carried out using ESPS/waves+ and Praat producing six levels as follows. The *phrase* level divides the speech into intonational phrases, the *words* level divides the speech into individual words, the *syllable* level divides the speech into individual syllables, the *segments* level divides the speech into vocalic and consonantal intervals, the *tones* level marks the pitch accents and boundary tones in the speech and finally the *pitch* level marks the variant pitch heights in the speech. Note that some utterances are also annotated at the part-of-speech and lemma level but these levels are generated automatically using annotation tools and are not considered for the evaluation presented here.

In order to minimise speaker variability a training corpus was constructed using only the annotated utterances for native speakers of read speech. For the evaluation reported here, the LeaP subcorpus consisting of the four read utterances by the native English speakers were used. Each of the four utterances in this subcorpus consists of approximately 268 words corresponding to a reading of a short story (cf. Gut (2004) for further details concerning the nature and vocabulary of the read speech used for this evaluation). Three of the four utterances were used as training data while the remaining utterance was used as test data. The multilevel annotations provided for the training utterances were encoded using the *phrase* level annotation as the domain annotation and the *words* level annotation as the input annotation[9]. Note that, since the encoding scheme presented in section 3 cannot at present encode levels of annotation describing instants, neither the *tones* nor the *pitch* levels are considered for this evaluation. However, as discussed previously, the encoding can be easily extended to cater for annotations of this type.

The encoding of the training utterances produced a positive sample consisting of 234 input-output pairs. Due to speaker variation in the corpus the sample contained ambiguous pairs having identical input but differing outputs. Such a sample cannot be modelled as a subsequential transducer and a more powerful class of transduction is required: the p-*subsequential* transducers. The p-subsequential transducers generalise the subsequential transducers by allowing $p \geq 0$ final outputs at a given state. The p final outputs are nondeterministic and ensure that ambiguous transductions can be accurately modelled. Therefore, in order to facilitate transducer inference from ambiguous samples, the Ostia inference algorithm is modified such that multiple nondeterministic final outputs are permitted. However, when partitioning the states of the tree (p-)subsequential transducer corresponding to an ambiguous sample, only those states which have identical final outputs are considered for merging. These modifications to Ostia are sufficient for our needs and allow generalised p-subsequential transductions to be inferred from ambiguous input-output pairs. Note, however, that these modifications may not be sufficient to guarantee that the modified Ostia algorithm identifies (in the limit) the class of p-subsequential transductions. Using the sample of input-output pairs corresponding to the encoded training corpus, the modified Ostia algorithm inferred a p-subsequential transducer with 369 states and 505 transitions. This transducer was then used to generate an annotation over the *phrase*, *syllable*, and *segments* level from the *words* level annotation of the test utterance. A brief comparison of this automatically generated multilevel annotation and the manually produced annotation for this utterance is now presented.

Note that, due to the size of the training and test data sets described above, the following sections describe only a preliminary evaluation of the grammatical inference approach to linguistic annotation. Naturally, an evaluation on a larger corpus of annotations is required in order to determine if the approach scales up to larger data sets. However, such an evaluation requires that a large corpus providing annotations over a large number and variety of linguistic levels. In contrast, speech corpora typically only provide annotations over a small and fixed number of levels.

5.2 Test Annotation Analysis

An initial attempt to produce an annotation for the test utterance failed because the manual *words* level annotation for the utterance contained tokens which were not seen in the training annotations. Thus, for example, a *words* token labelled "couldn't" appears in the test utterance which does not appear in the *words* level annotation for any of the training utterances (the corresponding expanded form consisting of two consecutive tokens labelled "could" and "not" does appear however). While identifying such variation between the test and training annotations may prove useful for automatic verification or consistency checking procedures, it

[8] Thanks to Ulrike Gut at the University of Freiburg for making the LeaP Corpus available.

[9] The *words* level was designated the input annotation since it encodes the minimum of variation. Thus, each speaker will utter the same words albeit with different phonetic realisations (resulting in varying annotations on the remaining levels). This maximises the likelihood that the inferred transducer can parse the *words* level tokens of the new utterance.

# Tokens Marked		# Tokens Marked with Correct Boundaries		# Tokens Marked with Correct Labels	
Non-speech	Speech	Non-Speech	Speech	Non-Speech	Speech
16	57	16	12	1	0

Table 1: Token analysis for the automatically generated *phrase* level annotation.

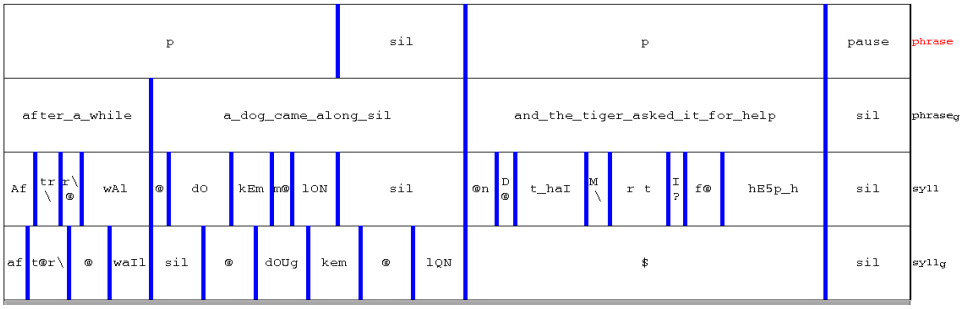

Figure 7: Comparison of the manual and automatic annotations for a portion of the test utterance (from $\approx$ 31.200642 seconds to $\approx$ 34.950642 seconds).

limits the ability of an inferred transducer to generate new annotations for unseen utterances. Note that this kind of variation between annotation tokens will be problematic for any automatic learning procedure applied to the annotation task since novel token labels appearing in the test data cannot be easily predicted from those seen in the training data. For this evaluation, problematic word tokens such as these (only three such instances occurred in the test utterance discussed here) were replaced by a special *match-all* symbol which forced them to be matched against their corresponding expanded forms in the training annotations. This enables the inferred transducer to parse the *words* level tokens of the test utterance into well-formed combinations which are then used to construct the remaining levels of annotation (cf. section 4). Note that, since the inferred transducer is in fact p-subsequential, a given well-formed combination may have a number of corresponding outputs. If a well-formed combination is identified with more than one corresponding output then a single output is chosen nondeterministically.

5.2.1 Phrase Level Annotation

Table 1 summarises the annotation tokens which were automatically generated at the *phrase* level for the test utterance. As can be seen, a total of 73 phrase level tokens were marked by the automatically generated annotation comprised of 16 non-speech intervals and 57 speech intervals. In contrast, the manual annotation for the test utterance marks 65 phrase level tokens; 33 of which represent non-speech intervals consisting of pauses, breaths, and silences while 32 represent intonational phrase intervals. The second column of Table 1 shows that 16 of the 33 non-speech intervals marked by the manual annotation were marked with the correct start and end times by the automatic annotation. The remaining 17 non-speech intervals were incorrectly integrated into an adjacent annotation token. This is illustrated in Figure 7 which shows a comparison of the actual manual annotation produced by a human annotator on the phrase and syllable levels (indicated by *phrase* and *syll* respectively) and the corresponding automatically generated levels (indicated by $phrase_g$ and $syll_g$) for a portion of the test utterance. As can be seen, an interval of silence at the phrase level has been incorrectly incorporated into the second token of the automatically generated annotation. Figure 7 also illustrates a case where a single phrase level token marked by the manual annotation has been incorrectly divided into two tokens by the automatic annotation. This type of annotation error, where a single intonational phrase is incorrectly marked as a sequence of multiple phrases, occurs a number of times and accounts for the additional 8 tokens generated by the automatic annotation at the phrase level.

Returning to the second column in Table 1, only 12 of the 32 speech intervals marked by the manual annotation are marked with the correct start and end times by the automatic annotation. The 20 incorrectly marked phrase level tokens arise as a result of an inaccurate phrase level parse by the inferred transducer. Thus, a number of non-speech intervals are incorrectly parsed as constituents of intonational phrases, while tokens which should have been parsed into a single intonational phrase are parsed into multiple subphrases.

Finally, examining the third column of Table 1, it can be seen that only one token marked by the automatic annotation for the phrase level is labelled correctly. The reason for this is that phrase level token labels are constructed directly from parsed word sequences. Thus, for example, all non-word tokens are labelled with "sil" in the supplied *words* level annotation. Therefore, all non-speech intervals marked in the automatically generated annotation will be labelled "sil" while the non-speech tokens in the manual annotation are labelled with different types of non-speech labels (e.g., sil, pause, breath, etc.). Similarly, all intonational phrases marked by the automatically generated annotation are labelled with the word tokens which constitute the phrase rather than a meaningful phrase label. These labelling errors at the phrase level can be seen in Figure 7.

The task of predicting intonational phrase boundaries from

# Tokens Marked	# Phrases Marked with Correct Number of Boundaries	# Tokens Marked with Correct Labels
377	25	67

Table 2: Token analysis for the automatically generated *syllable* level annotation.

a sequence of word tokens alone represents a difficult problem, and the automatic annotation marks acceptable phrase level tokens. The annotation errors discussed above could be overcome by integrating simple corrective heuristics as a post processing operation whereby, for example, a period of silence is always parsed as a single token. Such heuristics would be specific only to phrase level annotations and may not apply if the transducer used to generate the annotation is constructed with respect to some other domain. Therefore, domain-specific heuristic measures aimed at improving annotation accuracy are not applied as part of the annotation strategy outlined here.

5.2.2 Syllable Level Annotation

Considering the 57 intonational phrases marked by the automatically generated annotation (cf. Table 1), the second column of Table 2 shows that 25 of the phrases marked have associated *syllable* level annotations which mark the correct number of syllables when compared with the corresponding manual syllable level annotation. Considering again the annotation shown in Figure 7, the first two phrases marked by the automatically generated annotation mark the correct number (8) of syllable tokens when compared with the manual syllable annotation. Furthermore (ignoring the misplaced silence), one of the labels of the automatically marked tokens matches the label of its corresponding token while the labels of the remaining automatically marked tokens differ by only one insertion or substitution (typically a vowel) from the labels of the corresponding tokens of the manual annotation.

For the 32 intonational phrases marked by the automatically generated annotation which did not mark the correct number of syllable boundaries, 9 were marked with one more or one less syllable when compared with the manual annotation, 8 were marked with two more or two less syllables, and 5 were marked with three more or three less syllables. The remaining 10 phrases contained more extreme annotation errors at the syllable level such as the third phrase marked in Figure 7. As can be seen, 8 syllable boundaries which were marked by the manual annotation are marked as a single syllable labelled with "$" (representing the empty string). This spurious syllable marking occurs as a result of a combination of word tokens in the test utterance which is not seen in the training utterances. This causes the inferred transducer to parse this subsequence of word tokens via a path of transitions which generates no corresponding syllable (or segment) level annotation. The reason for this is that the acceptance path for the subsequence is in fact a subpath in a larger acceptance path for a longer sequence of word tokens. The associated syllable (and segment) level annotation for this longer sequence is generated by the output of an earlier transition on its acceptance path. Crucially, this earlier transition is not included in the acceptance subpath which is traversed for the subsequence under consideration here and so no output is generated. A similar problem occurs when a subsequence of word tokens which should be parsed as a single phrase is in fact parsed into a number of smaller subphrases. While this in itself causes incorrect phrase level annotations, as discussed previously, it also causes spurious annotations to be generated at the syllable level where a number of extra tokens are erroneously inserted.

The above annotation errors illustrate a potential drawback of using Ostia to infer transducers from multilevel annotations: inferred transducers do not in general maintain linguistic correspondences between input and output annotation tokens. The reason for this is that the inferred transducers are *onward*; producing a corresponding output string as soon as enough of the input string has been seen to uniquely identify it. For some input strings an output will only be produced when the entire input has been consumed whereas for others an output string may be produced when the first input symbol has been consumed while for others still outputs may be produced in stages as the input is progressively consumed. Thus, the nature of the inferred transducers can lead to cases where annotation tokens are erroneously inserted and/or deleted for those levels of annotation generated by transduction. Some of the extremely spurious syllable level annotations automatically marked for the test utterance could have been avoided if alternative paths were followed when parsing portions of the supplied word tokens. Others, however, could not have been avoided since no alternative paths were available. On the other hand, the training corpus used for the evaluation presented here is very small. If a larger training corpus is available and, if a *stochastic* transducer is inferred making use of the statistical information contained in the training annotations, then more accurate parses would be expected. This in turn would improve the automatically generated annotations for transduced levels of annotation (syllable and segment here). A stochastic transducer would also provide a more principled means for selecting a single output for ambiguous transductions. This is however left to future work.

Despite the fact that the automatically generated annotation marks spurious syllable boundaries for some portions of the test utterance, other portions of the test utterance are marked in a highly accurate manner when compared with the manual annotation. In any case, as for the phrase level, the automatically generated syllable level annotation is intended only as a first attempt which must be verified and corrected where required by a human annotator.

6 Conclusion

This paper has presented an approach based on grammatical inference techniques for automatically generating multilevel annotations of speech. An encoding scheme was described which transforms a training corpus of multilevel annotations over $n > 1$ linguistic levels into a positive sample

of input-output pairs. This sample can then be supplied to the Ostia inference algorithm which infers a corresponding finite-state transducer[10]. The inferred transducer is then used to automatically generate a new multilevel annotation for an unseen utterance.

The approach was evaluated on a small corpus of speech utterances providing four levels of phonological annotation. An analysis of an automatically generated annotation at the phrase and syllable level showed that inferred transducers are capable of producing very accurate annotations for some portions of the test utterance. On the other hand, some rather spurious annotations were also generated. However, the automatically generated annotations are intended only as a first attempt which is used to outline the predicted structural content on the different levels of annotation together with approximate temporal anchors to the speech signal. This first attempt must of course be verified by a human annotator. Therefore, the annotation procedure outlined here embodies a *semi-automatic* approach to generating linguistic annotation. Note that while traditional approaches to automatic speech recognition (e.g., HMMs) can be used to generate segment, syllable and word level annotations for a speech utterance, the grammatical inference approach outlined in this paper can be used to generate multilevel annotations over an arbitrary number and type of linguistic annotation, from sub-phonetic feature annotations to higher level semantic annotations.

Further evaluation must be carried out to determine if the grammatical inference approach can scale up to larger training samples of multilevel annotations over a greater number and variety of annotation levels. A stochastic extension (or alternative) to the Ostia algorithm which infers stochastic annotation transducers may help in this regard. Also, an enhanced encoding scheme which directly reflects the temporal anchors of multilevel annotations may improve the accuracy of those annotations generated by transduction. On the other hand, traditional finite-state definitions do not account for temporal anchors. A possible solution is to extend the traditional concept of finite-state machine to include temporal information (e.g., by storing temporal anchors as a property of the constituent states similar to the *Annotation Graph* formalism of Bird and Liberman (2001)). An enhanced encoding scheme which constructs *temporal* positive samples of input-output pairs could then be used in conjunction with a suitable inference algorithm (e.g., by extending Ostia) to construct a corresponding temporal finite-state automaton. Finally, general purpose heuristics, such as forcing the inferred transducer to output a parse which minimises the number of marked subsequences, may also improve the accuracy of the automatically generated annotations across all levels.

Acknowledgements

This material is based upon works supported by the Science Foundation Ireland under Grant No. 02/IN1/I100. The opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of Science Foundation Ireland.

Many thanks to Nicholas Kushmerick and John Nerbonne for valuable discussion and suggestions.

[10] In general, representing linguistic objects as finite-state machines (automata and transducers) has many advantages, in particular, space and time efficiency. In fact, the determinism condition of subsequential transducers ensures that they are maximally efficient. Also, Ostia constructs transducers with a minimal number of states ensuring inferred transducers are maximally compact.

7 References

Claude Barras, Edouard Geoffrois, Zhibiao Wu, and Mark Liberman. 2001. Transcriber: Development and use of a tool for assisting speech corpora production. *Speech Communication*, 33(1-2):5–22.

Steven Bird and Ewan Klein. 1990. Phonological events. *Journal of Linguistics*, 26(1):33–56.

Steven Bird and Mark Liberman. 2001. A formal framework for linguistic annotation. *Speech Communication*, 33(1-2):23–60.

Paul Boersma and David Weenik. 2000. Praat, a system for doing phonetics by computer, version 3.4. Technical Report 132, Institute of Phonetic Sciences of the University of Amsterdam. www.praat.org.

Steve Cassidy and Jonathan Harrington. 2001. Multi-level annotation in the Emu speech database management system. *Speech Communication*, 33(1-2):61–77.

King-Sun Fu and Taylor L. Booth. 1975. Grammatical inference: Introduction and survey, parts 1 and 2. *IEEE Transactions on Systems, Man and Cybernetics*, 5(1, 4):95–111, 409–423.

John S. Garofolo, Lori F. Lamel, William M. Fisher, Jonathan G. Fiscus, David S. Pallett, and Nancy L. Dahlgren. 1993. DARPA TIMIT acoustic-phonetic continuous speech corpus CD-ROM. Technical report, National Institute of Standards and Technology.

Ulrike Gut. 2004. The LeaP Corpus. Available at http://www.phonetik.uni-frieburg.de/leap/LeapCorpus.pdf.

Geoffrey Leech and Martin Weisser. 2003. Generic speech act annotation for task-oriented dialogues. In *Proceedings of Corpus Linguistics 2003*, Lancaster.

Jochen L. Leidner, Tiphaine Dalmas, Bonnie Webber, Johan Bos, and Claire Grover. 2003. Automatic multilayer corpus annotation for evaluating question answering methods: CBC4Kids. In *Proceedings of the 3rd International Workshop on Linguistically Interpreted Corpora*, Budapest.

David McKelvie, Amy Isard, Andreas Mengel, Morten Baum Moller, Michael Grosse, and Marion Klein. 2001. The MATE workbench- an annotation tool for XML coded speech corpora. *Speech Communication*, 33(1-2):97–112.

Jose Oncina, Pedro Garcia, and Enrique Vidal. 1993. Learning subsequential transducers for pattern recognition interpretation tasks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(4):448–458.

Emmanuel Roche and Yves Schabes. 1995. Approximating annotated corpora with finite-state transductions: A case study in part of speech tagging. Technical Report TR-95-10, Mitsubishi Electric Research Laboratories.

Author Index