Automatic Acquisition of Feature-Based Phonotactic Resources

Julie Carson-Berndsen & Robert Kelly & Moritz Neugebauer

Department of Computer Science University College Dublin

Dublin 4, Ireland

{julie.berndsen,robert.kelly,moritz.neugebauer}@ucd.ie

Abstract

Automata induction and typed feature theory are described in a unified framework for the automatic acquisition of feature-based phonotactic resources. The viability of this data-driven procedure is illustrated with examples taken from a corpus of syllable-labelled data.

1 Introduction

This paper combines two hitherto distinct areas of research, namely automata induction and typed feature theory, for the purposes of acquiring phonotactic resources for use in speech technology. In order to illustrate the methodology a small annotated data set for Italian has been chosen¹; however, given annotated data, the techniques can be applied to any language thus supporting language documentation at the phonotactic level and eventually building up a catalogue of reusable multilingual phonotactic resources.

There are numerous ways in which phonotactic information has been represented for use in speech technology applications ranging from phrase structure rules to *n*-grams. In this paper, the featurebased phonotactic automaton of the Time Map model (Carson-Berndsen, 1998) is used as the representational device. A phonotactic automaton describes all permissible sound combinations of a language within the domain of a syllable in terms of a finite state automaton, describing not only actual lexicalised syllables but also idiosyncratic gaps which would be considered well-formed by a native speaker of a language. The advantage of this representation of phonotactic constraints in the context of speech recognition is that it allows outof-vocabulary items (new words) to be classified as well-formed if they adhere to the constraints. Furthermore, since the phonotactic automaton constrains with respect to the syllable domain, it provides a more flexible and linguistically motivated

¹We use phonemically annotated data from the EUROM1 Multilingual European Speech Database. context than *n*-grams which restrict their context to a domain of fixed length (the *n*-1 preceding units).

A phonotactic automaton describes languagespecific constraints. Therefore, in order to develop multilingual phonotactic resources, phonotactic automata for different languages must be produced. Phonotactic automata for German and English have already been constructed for the Time Map model using manual techniques (Carson-Berndsen, 1998; Carson-Berndsen and Walsh, 2000). Since manual construction of phonotactic automata is time consuming and laborious, more recently focus has been placed on combining manual and automatic techniques in order to reduce the level of required human linguistic expertise. This will become more important when lesser-studied languages are addressed when an expert may not always be available. The techniques presented here are regarded as support tools for language documentation which allow inferences to be made based on generalisations found in an annotated data set. The linguist is free to accept or reject the suggestions made by the system.

In what follows, a technique is described in which phonotactic automata are acquired automatically given annotated data for a language. While this technique describes all forms found in the data, acquired automata cannot be considered complete since the data is likely to be sparse (in this paper we illustrate this using a small data sample). However, by combining phonotactic automata with a typed feature classification of sounds encountered in the data, it is possible to highlight not only distributional similarities, but also phonetic similarities which can be used to predict gaps in the representation. These can be presented to a user (at least a native speaker of a language) who can accept or reject these. Accepted forms are then integrated into the phonotactic automaton.

2 Automatic Acquisition of Phonotactic Automata

The approach described in this section is one alternative to a fully manual construction of phonotactic automata whereby they are rapidly acquired automatically and at a low cost. Given a corpus of well-formed syllables for the language in question, it is assumed here that the phonotactics for the language is implicit in the syllables themselves and can be automatically extracted by examining each syllable structure in turn. An extracted phonotactics is assumed to describe at least the syllables in the corpus and is also assumed to be an approximation of the complete phonotactics for the language from which the data was drawn. Since the phonotactics in question are finite-state structures and the data available for acquiring phonotactics is a corpus of positive examples of well-formed syllable structures, the approach adopted here is to apply a regular grammatical inference procedure which can learn from positive data alone. The field of grammatical inference has yielded many important learnability results for different language classes. A full discussion of these results is beyond the scope of this paper however see Belz (2000, Chapter 3) for a concise summary and discussion and Angluin and Smith (1983) for a survey style introduction to the field. Suffice to say that since the formal language of well-formed syllables in a given natural language is finite, it is possible to learn the structure of a regular grammar i.e. the required phonotactic automaton from positive data alone i.e. the corpus of well-formed syllables.

The choice of regular inference algorithm is in fact arbitrary. Many algorithms have been developed which can perform this learning task. For the purposes of this paper however the ALERGIA (Carrasco and Oncina, 1999) regular inference algorithm is used. This algorithm as applied to the problem of inferring phonotactic automata is described in detail elsewhere (Kelly, 2004b). Here, the workings of the algorithm are described by example. Note that ALERGIA in fact treats any positive data sample as having been generated by a stochastic process. Thus, learned automata are in fact stochastic automata i.e. automata in which both states and transitions have associated probabilities, however traditional automata can be obtained by simply ignoring these probabilities. Table 1 shows a small subset of the Italian data set consisting of 14 well-formed Italian syllables each consisting of 3 segments and transcribed using the SAMPA phonetic alphabet 2 .

The ALERGIA inference algorithm takes as input a sample set of positive strings S (representing well-formed syllables in this case) together with a confidence value α and outputs a deterministic stochastic automaton A which is minimal for

/v e n/	/r a n/	/b e n/	/m e n/	/t w a/
/k a n/	/n o n/	/t o n/	/f j o/	/r a n/
/d j o/	/s t o/	/s t e/	/t s a/	/p l o/

Table 1: Training set of Italian syllables.

the language it describes. ALERGIA proceeds by building a Prefix Tree Automaton (PTA) from the strings in S. The PTA is a deterministic automaton with a single path of state-transitions from its unique start state for each unique prefix which occurs in S. Also, the PTA has a single acceptance path, i.e. path of state-transitions from the start state to some final state, for each unique string in S where an initial subset of the transitions in acceptance paths for strings are shared if those strings have a common prefix. Thus, common prefixes are essentially merged giving the PTA its tree shape. The PTA for S accepts exactly the strings in S and each state of the PTA is associated with a unique prefix of S. ALERGIA also assigns each transition in the PTA a frequency count dependent on the number of prefixes in S which share that transition. Similarly, each state has an assigned frequency dependent on the number of strings from S which are accepted (or equivalently, generated) at that state. The PTA for the Italian syllables of table 1 is shown in figure 1. Note that final states are denoted by double circles and the single start state is state 0. In figure 1 final state 26 has a frequency of 2 since the two occurrences of the syllable /r a n/ terminate at this state. All other final states have a frequency of 1 since exactly one syllable terminates at each final state. All other states have a frequency of 0. Similarly, the transition from state 0 to state 13 has a frequency of 3 since three of the syllables in the training set begin with t/and the transitions from state 0 to 17 and from state 17 to 18 have a frequency of 2 since two syllables begin with the segment combination /s t/. The frquencies associated with the states and transitions of the PTA can be used to associate a stochstic language with each state. The set of acceptance paths from a given state determine the set of strings in its associated language and the probability for a given string in the language is easily derived from the frequencies of the states and transitions on the acceptance path for that string.

ALERGIA uses the PTA as the starting point for constructing a canonical, i.e. minimal deterministic, automaton for S. The canonical automaton is identified by performing an ordered search of the automata derivable from the PTA by partitioning and merging subsets of states of the PTA. Using the stan-

²http://www.phon.ucl.ac.uk/home/sampa/



Figure 1: Prefix Tree Automaton for the syllables in table 1.

dard order on the prefixes associated with the states of the PTA, pairs of states are subsequently examined to determine if they generate a similar stochastic language within a statistical significance bound dependent on the supplied confidence value α . If a pair of states are deemed to statistically generate the same language then they are merged into a single state and the state-transitions of the automaton are altered to reflect this merge. The canonical automaton is identified when no more state merges are possible. Figure 2 shows the canonical automaton derived from the PTA in figure 1.

Since automata are derived from training sets of syllables through the use of a language independent regular inference algorithm, the procedure described above is generic and language independent. However, the procedure is of course dependent on the existence of a corpus of training syllables for the language in question and since it is entirely data



Figure 2: Canonical Automaton for the PTA in figure 1.

driven, the quality of the resulting phonotactics will be dependent on the quality and completeness of the syllable corpus. Thus, firstly the corpus must have high quality annotations. Fortunately, the need for high quality annotations in corpora is now recognised and has become an essential part of speech technology research and we assume here that high quality annotations are available. Secondly, if valid sound combinations are not detailed in the training corpus then they may never be represented in the learned phonotactics. In order to be complete the learned automaton must model all valid sound combinations, however. In this case, generalisation techniques must be applied in conjunction with the inference algorithm in order to identify and rectify gaps in the training corpus. This ensures that the acquired phonotactics describes as close an approximation as possible to the complete phonotactics for the language. One such approach to generalisation which operates independently of the chosen regular inference algorithm is described in Kelly (2004a). An alternative technique is discussed in section 3.

Finally, note that learned automata represent the first stage in the development of multilingual phonological resources called *Multilingual Time Maps* (MTMs) (Carson-Berndsen, 2002). An MTM extends the single tape model of a phonotactic automaton to a multitape transducer whereby the different transition tapes detail linguistic information

of varying levels of granularity and related to the original segment label. An MTM might have individual tapes detailing the segment, the phonological features associated with that segment, the average duration of the segment in a particular syllabic position etc. In particular, the segment tape of the learned phonotactic automata can be augmented with additional tapes detailing feature type labels associated with the segments. These additional type label tapes are discussed in more detail in the following section.

3 Phonotactic Automata and Typed Feature Structures

Lexical knowledge representation in computational phonology has already made extensive use of inheritance hierarchies to model lexical generalisations ranging from higher level prosodic categories to the phonological segment. In contrast to the approach presented in this section, the work described in (Cahill et al., 2000) is set in an untyped feature system using DATR to define inheritance networks with path-value equations (Evans and Gazdar, 1996). The merits of applying a type discipline even to untyped feature structures is considered in Wintner and Sarkar (2002) from a general perspective and in Neugebauer (2003b) with special reference to phonological lexica.

Previous proposals to cast phonological structure in a typed feature system can be found in Bird and Klein (1994) and Walther (1999). However, there are two major differences with regard to our work. First, while types may denote sets of segments, we go beyond the idea of sets as arc labels in finite-state automata (Bird and Ellison, 1994; Eisner, 1997; van Noord and Gerdemann, 2001) which says that boolean combinations of finitely-valued features can be stored as a set on just one arc, rather than being multiplied out as a disjunctive collection of arcs. This choice has no theoretical consequences but is merely a convenience for grammar development (Bird and Ellison, 1994). The difference in our approach consists in the hierarchical ordering of types (or sets) which relates each arc label to any other type in a given phonological typed feature system; such type-augmented automata have been formally defined in Neugebauer (2003c). Second, inheritance of type constraints is assumed to govern all subsegmental feature information (Neugebauer, 2003b). Since here the crucial inheritance relationships are induced automatically, we elaborate on work by Walther (1999) where a complex hand-crafted type hierarchy for internal segmental structure is mentioned instead of simple appropriateness declarations (Bird and Klein, 1994).

The interaction of finite-state automata and typed feature structures is depicted in figure 3. Transitions are exhaustively defined over a set of type labels which are characterised by a unique position in the underlying type hierarchy. This hierarchy is key to the compilation of well-formed segment definitions which are achieved by unification of partial feature structures. In the simplest case, only atomic types appear on the arcs which means that types correspond to singleton sets. This can be achieved for a phonemically annotated corpus (just like the corpus in figure 1) by replacing all occurrences of a phoneme with its appropriate atomic type label.



Figure 3: Example of the type-augmented automaton for [traf].

The semantics of the type system assumed here are extremely simple: the denotation of a parent type in the directed acyclic graph that constitutes a type hierarchy is defined as the union of the denotation of its children, whereas a type node without children denotes a unique singleton set (Aït-Kaci et al., 1989). Complex type formulae - as constructed by logical AND – are implicitly supported for the case of intersections since the greatest lower bound condition (Carpenter, 1992) is assumed: its the formal definition (also known as meet) states that in a bounded complete partial order that is an inheritance hierarchy, two types are either incompatible or compatible. While in the first case, type constraints are shared, in the latter case we require them to have a unique highest common descendant. As suggested in the LKB system (Copestake, 2002), these types will in our approach be generated automatically if a type hierarchy does not conform to this condition. These types – such as *glbtype2* in figure 3 - do not have their own local constraint description and thus do not rely on purely linguistic motivation.

A useful application of the greatest lower bound condition seems to be the possibility that we can refer to a set of compatible types simply by reference to their common descendant. As indicated by the hierarchical structure which is built over *type09* in figure 3, atomic types encode maximal information whereas non-atomic types characteristically contain only partial information. Thus, by defining transitions over types such as *type34* we might elegantly capture phonotactic generalisations over a subset of fricative sounds. This naturally raises the question as to how the hierarchies are actually determined; a suitable algorithm is described below, a detailed specification is provided in Neugebauer (2003a).

Given a set of phonological feature bundles, an inheritance hierarchy may be generated in the following way. For each feature which is defined for a linguistic object (here: a phoneme) we compute the corresponding extent or set description. The algorithm then inserts these set descriptions into a lattice and looks them up at the same time: it asks for the smallest description that is greater than a singleton set o with respect to the total order \prec used inside the tree. Every fully specified feature structure for a given phoneme will deliver such a singleton set, given that no two segments have been defined using the identical feature structure.

The algorithm can be employed to recursively compute all set descriptions of a feature system by starting from the smallest set description of the lattice. We need the lattice structure to encode the inheritance relationships between sets; in the context of lattice computation we will refer to these sets in terms of nodes. Every set description o has two lists associated with it: the list o^* of its upper nodes and the list o_* of its lower nodes. One node may be shared by two different set descriptions as their upper node. While the algorithm processes each of those two set descriptions, their shared upper node must be detected in order to configure the relationships correctly. To this end, all set descriptions are stored in a search tree T. Every time the algorithm finds a node it searches for it in the tree T to find previously inserted instances of that set description. If the description is found, the existing lists of nodes are updated; otherwise the previously unknown set description is entered into the tree. Figure 4 demonstrates this procedure for the feature bundle {fricative, labiodental, voiceless}. Once the smallest set description [labiodental] is not able to include one of segments which are successively added to it a new upper node is created.

To make sure that all set descriptions that are inserted into the tree are also considered for their up-



Figure 4: Induction of subsumption hierarchies.

per nodes, the total tree order \prec must relate to the partial lattice order \leq in the following way: $o_1 < o_2$ implies $o_1 \prec o_2$. This is how recently inserted nodes are greater than the actual set description with respect to \prec and will be considered later.³

Once we have computed all set descriptions, we finally assign types and type constraints to all nodes in the hierarchy. Therefore, the set of feature structures which constituted the starting point of our algorithm has now been computed into a data structure which supersedes the previous level of information in terms of a type inheritance network. The last step consists of the insertion of greatest lower bounds thus generating a well-formed lattice. Figure 5 visualises the final type inheritance hierarchy for an Italian corpus containing 22 phonemes, each corresponding to a unique atomic type (*type01*, ..., type22). While the types numbered 23 to 39 are generated by our set-theoretic algorithm, the greatest lower bounds (the glb-types) are required by the formal characteristics of our type system.

Any of the non-atomic types may be used to express generalisations over sets of phonological segments since each partial feature structure subsumes all compatible fully specified segment entries. Additionally, non-atomic nodes may be associated with constraints which define appropriateness declarations for linguistic signs of a particular type. For example, all segments are at least characterised with respect to four attributes (phonation, manner, place and phonetic symbol). The next section sketches an application of typed feature structures addressing

³Just computing the (ordered) set descriptions turns out to be more effective since no hierarchy has to be computed. Computing set descriptions as well as their hierarchical structure takes twice as long for the same input when the algorithm is used. This is also due to memory usage: while the computation of all set descriptions is only based on a single predecessor, the integration of a lattice algorithm stores all set descriptions in a persistent search tree.



Figure 5: Complete generated type hierarchy.

data sparseness in automatically learned phonotactic automata.

4 Examples

The integrated approach utilising both automata induction and typed feature theory as presented in the previous sections requires a phonemically annotated corpus. Each phoneme is then mapped to a canonical feature bundle which is based on the phonetic characteristics specified in the International Phonetic Alphabet (IPA); the features used in Figure 3 serve as an example. Our set-theoretic algorithm operates on these feature structures thus deriving a type inheritance network for the corpus in question. Note that phonemes and features are not corpus-specific but rather a subset of a languageindependent set of linguistic descriptions that is the IPA. As a result, we obtain a representation of our annotation alphabet (phoneme and feature labels) which exclusively refers to (sets of) linguistic objects via their corresponding types. This is exemplified in figure 3 for an individual sound and in figure 5 for the full corpus.

Once the complete type hierarchy has been generated the inheritance relationships described can be used to construct more compact finite-state structures than the automata learned over the original data set. In addition, the linguistic generalisations described by the hierarchy can be used to address data sparseness in the training corpus. To illustrate this, the automata are learned over type labels rather than segments. Since all transitions of the learned automata will now be labelled with types the information in the feature hierachy can be used to express generalisations. To ensure that automata are learned over types the segments in the training data must be replaced with type labels that correspond to singleton sets containing only the original segment. For example, the syllable /*r a n*/ in table 1 would be replaced by /*type03 type01 type10*/. Note that in this case the transitions of the learned automaton will be labelled with type labels rather than segments.

In the first case, the type hierarchy allows more compact automata to be constructed by examining the set of transitions emanating from each state of the learned automaton. If each of the transitions emanating from a given state s_1 have the same destination state s_2 then the type labels on each transition are examined to determine if they have a common ancestor node in the hierarchy. If a common ancestor exists and if no other type label is the child of that parent other than those appearing on the set of transitions then they can be replaced by a single transition from s_1 to s_2 labelled with the parent type. The topmost diagram of figure 6 illustrates a small section of the learned automaton for the full Italian data set. In this case there are two transitions from state 22 to state 35, one labelled with type15 and the other labelled with type06. Referring to the hierarchy in figure 5, a common parent of type15 and type06 is type30 and the only children of type30 are type15 and type06. Therefore these two transitions can be replaced by the single transition labelled with type30.

Note that replacements of the kind described above serve only to produce more compact automata and do not extend the coverage of the automaton. However, it is possible to use the type hierarchy to achieve a more complete phonotactics. The middle diagram of figure 6 shows another small section of the learned automaton for the full Italian data set. Referring again to the hierarchy in figure 5, it can be seen that type29 is a parent of type14 which labels the transition from state 0 to state 7 and is also a parent of type18 which labels the transition from state 0 to state 5. Similarly, type25 is a common parent of type06 and type01 which label the transitions from state 7 to state 17 and state 5 to state 25 respectively. Finally, type35 is a common parent of type10 and *type14* which label the transitions from state 17 to state 35 and state 25 to state 35. If each type label is replaced by its common parent then both transitions from state 0 are labelled with type29. Also, the paths emanating from the destination states of these transitions (states 5 and 7) are both labelled with type25. In this case state 5 and state 7 can be merged into a single state. A similar state merging can be performed for states 17 and 35 resulting in a new automaton as shown in figure 6. This process yields a more general phonotactics since type29 actually denotes the segment set $\{p, m, b\}$ and type25denotes the set $\{a, i, e, E\}$. Thus, the segment p has been effectively introduced as a new onset consonant cluster that can precede any vowel in the set denoted by type25. Also, as a result of introducing type25 the additional vowels i and E have been introduced as new vowel clusters. Note however that type35 denotes exactly the set $\{m, n\}$ and so no new coda clusters are introduced.



Figure 6: Finite-state diagrams.

5 Conclusion

An important pre-requisite for the development of robust multilingual speech technology applications is the availability of language resources at varying levels of granularity. This paper has presented generic techniques for acquisition of languagespecific phonotactic resources. The techniques were exemplified using a small data set for Italian.⁴ but scale to larger data sets and can be applied to any language. Although the induction techniques as described here assume that data is annotated at the syllable level, only very few corpora are actually annotated at this level; a more usual annotation is at the phonemic level. As a result, a cyclical learning procedure has been developed which learns as syllable annotation is being performed and uses the phonotactic automaton developed thus far to predict syllable boundaries for annotation support (Kelly, 2004b). The work presented in this paper represents one specific step towards the provision of finegrained representations for speech recognition and

synthesis based on a combination of data-driven and user-driven techniques.

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.

References

- Aït-Kaci, Hassan, R. Boyer, P. Lincoln, and R. Nasr. 1989. Efficient Implementation of Lattice Operations. ACM Transactions on Programming Languages and Systems, 11(1):115–146.
- Dana Angluin and Carl H. Smith. 1983. Inductive inference: Theory and methods. ACM Computing Surveys, 15(3):237–269.
- Anja Belz. 2000. Computational Learning of Finite-State Models for Natural Language Processing. Ph.D. thesis, University of Sussex.
- Steven Bird and T. Mark Ellison. 1994. One– Level Phonology. *Computational Linguistics*, 20(1):55–90.
- Steven Bird and Ewan Klein. 1994. Phonological Analysis in Typed Feature Systems. *Computational Linguistics*, 20:455–491.
- Lynne Cahill, Julie Carson-Berndsen, and Gerald Gazdar. 2000. Phonology–based Lexical Knowledge Representation. In *Lexicon Development* for Speech and Language Processing, pages 77– 114. Kluwer Academic Publishers, Dordrecht.
- Bob Carpenter. 1992. *The Logic of Typed Feature Structures*, volume 32 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, Cambridge.
- Rafael C. Carrasco and Jose Oncina. 1999. Learning deterministic regular grammars from stochastic samples in polynomial time. *ITA*, 33(1):1–19.
- Julie Carson-Berndsen and Michael Walsh. 2000. Interpreting multilinear representations in speech. In *Proceedings of the 8th Australian Conference on Speech Science and Technology*, pages 472–477, Canberra, December.
- Julie Carson-Berndsen. 1998. *Time Map Phonology: Finite State Models and Event Logics in Speech Recognition*. Kluwer Academic Publishers, Dordrecht, Holland.
- Julie Carson-Berndsen. 2002. Multilingual time maps: Portable phonotactic models for speech technology applications. In *Proceedings of the LREC 2002 Workshop on Portability Issues in Human Language Technology*.

⁴Due to space constraints this paper only includes selected examples of the acquired resources. Additional information is publicly available at http://muster.ucd.ie/sigphon/. This includes the complete annotation alphabet (phoneme and feature set), the typed feature system and complete state diagrams for all phonotactic automata.

- Ann Copestake. 2002. *Implementing Typed Feature Structure Grammars*, volume 110 of *CSLI Lecture Notes*. CSLI Publications, Center for the Study of Language and Information.
- Jason Eisner. 1997. Efficient generation in primitive optimality theory. In *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics and the 8th Conference of the European Association for Computational Linguistics*, Madrid.
- Roger Evans and Gerald Gazdar. 1996. DATR: A language for Lexical Knowledge Representation. *Computational Linguistics*, 22(2):176–216.
- Robert Kelly. 2004a. Generalisation in the automatic acquisition of phonotactic resources. To Appear in Proceedings of The University of Cambridge Second Postgraduate Conference in Language Research.
- Robert Kelly. 2004b. A language independent approach to acquiring phonotactic resources for speech recognition. In *Proceedings of the 7th Annual Colloquium for the UK Special Interest Group for Computational Linguistics*, pages 126–133. CLUK04.
- Moritz Neugebauer. 2003a. Automatic Generation of Constraint Hierarchies. Poster presented at the 14th Meeting of Computational Linguistics in the Netherlands, University of Antwerp.
- Moritz Neugebauer. 2003b. Computational Phonology and Typed Feature Structures. In Proceedings of the First CamLing Postgraduate Conference on Language Research. Cambridge. University of Cambridge.
- Moritz Neugebauer. 2003c. Subsumption in Speech Recognition and Feature Theory. In Proceedings of the Twenty-ninth Annual Meeting of the Berkeley Linguistics Society, University of California at Berkeley. Berkeley Linguistics Society.
- Gertjan van Noord and Dale Gerdemann. 2001. Finite State Transducers with Predicates and Identities. *Grammars*, 4(3):263–286.
- Markus Walther. 1999. One–Level Prosodic Morphology. In *Marburger Arbeiten zur Linguistik*, volume 1, Philipps–Universität Marburg.
- Shuly Wintner and Anoop Sarkar. 2002. A note on typing feature structures. *Computational Linguistics*, 28(3):389–397.