An algorithm to assign features to a set of natural classes

Mayer, Connor Joseph
connor.joseph.mayer@gmail.com

Daland, Robert
r.daland@gmail.com

December 9, 2017

Abstract

This squib describes a dynamic programming algorithm which assigns features to a set of natural classes. The input consists of a set of classes, each containing one or more segments; in other words, a subset of the powerset of a segmental alphabet Σ. If a class can be generated as the union of existing features (= intersection of already-processed classes), those features are propagated to every segment in the class. Otherwise, a new feature/value is assigned. The algorithm comes in 4 flavors, which differ with respect to complementation and how negative values are assigned. We show that these variants yield *privative specification*, *contrastive underspecification*, *contrastive specification*, and *full specification*, respectively. The main text sets out necessary background, and illustrates each variant of the algorithm. The Appendix formally proves that each algorithm is sound.

1 Introduction

merge what Connor wrote

2 Definitions and notation

Let Σ denote an alphabet of segments. We will use the term *class* to mean a subset of Σ. A *natural class system* $\mathcal{C}$ is a set of classes over Σ, $\mathcal{C} = \{C_i\}_{i=1}^{N}$, which includes Σ itself, and the empty set (i.e. $\varnothing, \Sigma \in \mathcal{C}$). Readers who are familiar with the notion of *lattice* will note that every natural class system forms a lattice under the subset relation. To illustrate, a vowel harmony lattice is shown below (the empty set is suppressed):

A *feature system* is a tuple $(\mathcal{F}, \Sigma, \mathcal{V})$ where

- Σ is a segmental alphabet,
- $\mathcal{V}$ is a set of values, and
- $\mathcal{F} = \{f_j : \Sigma \to \mathcal{V}\}_{j=1}^{M}$ is a set of feature functions mapping segments to feature values

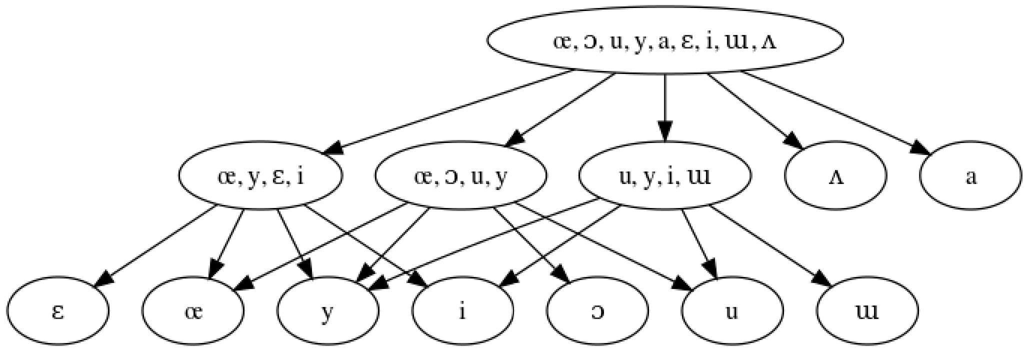

Figure 1: Vowel harmony lattice

We say that a feature system has *privative specification* if $\mathcal{V} = \{+, 0\}$, *full specification* if $\mathcal{V} = \{+, -\}$, and *contrastive specification* if $\mathcal{V} = \{+, -, 0\}$. We do not consider other value sets here.

A (fully specified) feature system for the vowel harmony lattice shown in Fig. 1 is shown below:

σ	front	back	low	high	round
i	+	−	−	+	−
y	+	−	−	+	+
ɯ	−	+	−	+	−
u	−	+	−	+	+
ɛ	+	−	−	−	−
œ	+	−	−	−	+
ʌ	−	+	−	−	−
ɔ	−	+	−	−	+
a	−	+	+	−	−

Table 1: Example of a (fully specified) featurization.

Let $(\mathcal{F}, \Sigma, \mathcal{V})$ be a feature system. The following definitions will prove useful:

- We will refer to the set of feature functions $\mathcal{F} = \{f_j\}_{j=1}^{M}$ as a *featurization* (of Σ).
- A *featural descriptor* $\mathbf{e}$ is a subset of $(\mathcal{V} \setminus \{0\}) \times \mathcal{F}$
  - in other words, $\mathbf{e}$ is a set of feature/value pairs, where the value cannot be 0
  - an example is [+front, −low]

- The natural class described by a featural descriptor $\mathbf{e}$, written $\langle \mathbf{e} \rangle$, consists of every segment which has *at least* the feature/value pairs in $\mathbf{e}$
  - featural descriptors can be written in the form $[\alpha_k f_k]_{k \in K}$ for some index set K
  - $\mathbf{e} = [\alpha_k f_k]_{k \in K}$ if and only if $\langle \mathbf{e} \rangle = \{x \in \Sigma \,|\, \forall k \in K\, [f_k(x) = \alpha_k]\}$
  - for the feature system in Table 1, the natural class described by [+front, −low] is {i, y, ɛ, œ}

- Let $\mathcal{V}^{\mathcal{F}}$ denote the set of all licit featural descriptors over $(\mathcal{F}, \Sigma, \mathcal{V})$
  - Formally, $\mathcal{V}^{\mathcal{F}} = \mathcal{P}((\mathcal{V} \setminus \{0\}) \times \mathcal{F})$, where $\mathcal{P}(X)$ is the powerset of X
  - Define $\langle \mathcal{V}^{\mathcal{F}} \rangle = \{\langle e \rangle \,|\, e \in \mathcal{V}^{\mathcal{F}}\}$
  - In other words, $\langle \mathcal{V}^{\mathcal{F}} \rangle$ is the set of all natural classes that can be generated by featural descriptors over $(\mathcal{F}, \Sigma, \mathcal{V})$

Note that while every featural descriptor in $\mathcal{V}^{\mathcal{F}}$ picks out a class in $\langle \mathcal{V}^{\mathcal{F}} \rangle$, the two are not in 1-1 correspondence. This is because the same class can often be described by multiple featural descriptors. For example, under the the vowel feature system shown in Table 1, the featural descriptor [+front] picks out the same class as the featural descriptor [+front, −low] (the front vowels); and the featural descriptors [+front, −front] and [+high, +low] both pick out the empty set.

Let $(\mathcal{F}, \Sigma, \mathcal{V})$ be a feature system with featurization $\mathcal{F} = \{f_j\}_{j=1}^{M}$.

- The *feature vector* of a segment x is the tuple $F(x) = (f_j(x))_{j=1}^{M}$.
- Two segments x, y are *featurally distinct* if and only if $F(x) \neq F(y)$; in other words, if they do not match on at least feature.
- The feature system is *well-formed* if every pair of segments in Σ is featurally distinct.[1]
- A feature f_j is *redundant* if $\mathcal{F}' = \mathcal{F} \setminus \{f_j\}$ is well-formed.
- A featurization is *efficient* if it contains no redundant features.

It is straightforward to show that if $(\mathcal{F}, \Sigma, \mathcal{V})$ is a well-formed feature system, then it generates a natural class system. Our goal in the remainder of this paper is to go the opposite direction: starting with a natural class system $\mathcal{C}$ over an alphabet Σ, can we assign an efficient, well-formed feature system $(\mathcal{F}, \Sigma, \mathcal{V})$ that is rich enough to generate $\mathcal{C}$?

[1] It is always possible to make ill-formed systems become well-formed. For example, suppose that [ptk] are not given distinct place features. One way to make the system well-formed is to add place features. Another way is to replace instances of [ptk] with a meta-symbol [T] in Σ, yielding a new segmental alphabet $\Sigma' = \Sigma \setminus \{p, t, k\} \cup \{T\}$.

3 Intersectional closure

In this section we define the *intersectional closure* of a natural class system $\mathcal{C}$. We prove that if a feature system is expressive enough to generate all the classes in $\mathcal{C}$, it generates the intersectional closure. Then we give a dynamic programming algorithm which efficiently computes the intersectional closure of a natural class system, as well as the intersection relation. It turns out that these structures are exactly what are needed to assign an efficient feature system.

The *intersectional closure* of $\mathcal{C}$, denoted $\mathcal{C}_\cap$, is the natural class system consisting of every class that can be generated by the intersection of finitely many classes in $\mathcal{C}$. In other words, $\mathcal{C}_\cap$ consists of every class in $\mathcal{C}$, as well as any class that can be generated by the intersection of two or more classes in $\mathcal{C}$.

Theorem: Let $\mathcal{C} = \{C_i\}_{i=1}^n$ be a natural class system and $(\mathcal{F}, \Sigma, \mathcal{V})$ a feature set. If $\mathcal{C} \subset \langle \mathcal{V}^\mathcal{F} \rangle$, then $\mathcal{C}_\cap \subset \langle \mathcal{V}^\mathcal{F} \rangle$. In other words, if $(\mathcal{F}, \Sigma, \mathcal{V})$ is rich enough to generate $\mathcal{C}$, it generates the intersectional closure.

Proof: Let C_i, C_j be classes in $\mathcal{C}$, so that $C_i, C_j \in \langle \mathcal{V}^\mathcal{F} \rangle$. This means that there exist featural descriptors $\mathbf{e}_i, \mathbf{e}_j \in \mathcal{V}^\mathcal{F}$ such that $\langle \mathbf{e}_i \rangle = C_i$ and $\langle \mathbf{e}_j \rangle = C_j$. Now $\mathbf{e}_i$ and $\mathbf{e}_j$ are set of feature/value pairs. *Claim*: $\langle \mathbf{e}_i \cup \mathbf{e}_j \rangle = C_i \cap C_j$. Proof that $C_i \cap C_j \subset \langle \mathbf{e} \rangle$: If segment $x \in C_i \cap C_j$, then $x \in C_i$. By definition, x must have the features in $\mathbf{e}_i$; similarly, since $x \in C_j$, x must have the features in $\mathbf{e}_j$. Thus, x has the features in $\mathbf{e}_i \cup \mathbf{e}_j$. Proof that $\langle \mathbf{e}_i \cup \mathbf{e}_j \rangle \subset C_i \cap C_j$. Let $x \in \langle \mathbf{e}_i \cup \mathbf{e}_j \rangle$. Then x has all the features of $\mathbf{e}_i$, and so $x \in C_i$. And x has all the features of $\mathbf{e}_j$, so $x \in C_j$. This illustrates that the union operation on a pair of featural descriptors corresponds to the intersection operation on the corresponding pair natural classes. Since union and intersection operations are associative, the extension to any number of finite unions/intersections proceeds by induction (e.g. if $C_i, C_j, C_k \in \mathcal{C}$, $C_i \cap C_j \cap C_k = (C_i \cap C_j) \cap C_k$; $(C_i \cap C_j) \in C_\cap$ and $C_k \in C_\cap$, so $C_i \cap C_j \cap C_k \in \mathcal{C}_\cap$).

Next, we give an algorithm which computes the intersectional closure, a modified variant of Dijkstra's shortest-paths algorithm. As we will show later, the computational benefit of precomputing the intersectional closure is that it efficiently computes the intersection relation, which reduces the computational complexity of the featurization algorithm. We assume that the input is a natural class system $\mathcal{C} = \{C_i\}_{i=1}^N$, whose classes are sorted in decreasing order of cardinality. (This implies that $C_i \not\subseteq C_j$ whenever $j > i$, so there is no need to check the subset relation.)

Require: $\mathcal{C}$ sorted by decreasing size ($|C_i| \geq |C_{i+1}|$ for every $i = 1 \ldots N-1$)
Require: subset matrix S ($N \times N$): $S_{ij} = 1$ if $C_j \subset C_i$, 0 otherwise

closure $\leftarrow \mathcal{C}$

```
pairQueue ← {(i, j)|i < j}
intersections ← ∅

while pairQueue ≠ ∅ do
  (i, j) ← POP(pairQueue)
  if S_ij = 0 then
    c ← C_i ∩ C_j
    if c ∈ closure then {found an existing intersection}
      k ← index such that closure_k = c
      PUSH (i, j, k) → intersections
    else {found a new class!}
      N ← |closure|
      APPEND c to closure
      RESIZE S → (N + 1) × (N + 1)
      for k = 1 to N do
        if c ⊂ C_k then {update subset matrix}
          S_{k,N+1} → 1
        end if
        if C_k ⊂ c then
          S_{N+1,k} → 1
        end if
        PUSH (k, N + 1) → pairQueue
        PUSH (N + 1, k) → pairQueue
      end for
    end if
  end if
end while
```

4 Privative specification

achieved by assigning a new feature [+f] only, to every segment in X

5 Contrastive underspecification

achieved by assigning a new feature [+f] to every segment in X, and if $Y \setminus X$ (the complement of X with respect to Y) is in the input, then [-f] is assigned to every segment in $Y \setminus X$

6 Contrastive specification

achieved by assigning a new feature [+f] to every segment in X, and [-f] to every segment in $Y \setminus X$ (even if $Y \setminus X$ was not in the input)

7 Full specification

achieved by assigning a new feature [+f] to every segment in X, and [-f] to every segment in $\Sigma \setminus X$

A Formal proof of the algorithm

A.1 Privative underspecification

A.2 Contrastive underspecification

A.3 Contrastive specification

A.4 Full specification