Prerequisites

- sets (notation, operations)
- strings (notation)

Combining *n*-gram grammars

So far, our short expedition has resulted in a few formal insights into *n*-gram grammars, but also a variety of examples of how they can be used in linguistics. However, all the examples were limited to specific phenomena. If *n*-gram grammars are supposed to provide a model of phonotactics or morphotactics, it isn't enough to show that individual phenomena in those domains can be described with those grammars. We already know that word-final devoicing can be handled by *n*-gram grammars, and the same is true for penultimate stress. But a speaker whose language displays both phenomena must have some grammar that takes care of both. Is this combined grammar also an *n*-gram grammar? It is not immediately obvious that the answer to that is yes.

There are other types of grammars for which the answer is no. Later on, we will encounter context-free grammars (CFGs). For CFGs, it is not the case that what can be jointly accomplished by two grammars can be done by a single grammar. The problem is mostly limited to artificial languages (which is interesting in itself — apparently all natural languages have some special property that avoids this problem). But it nonetheless shows that one cannot model each phenomenon with its own grammar and simply assume that they can all be combined into a single grammar of the same type.

Example 1 Consider the language $L_{ab} := a^n b^n c^+$. This language only contains strings of the form *abδ*, *aabbδ*, *aaabbbδ*, and so on, where δ is a shorthand for 1 or more *c*s. So this language contains strings like *aabbc*, *abcccc*, or *aaaabbbbcc*, but not *abbc*. Intuitively, L_{ab} is the language of strings where all *a* occur before all *b*s, all *b*s occur before all *c*s, and the number of *a*s and *b*s must match. A very similar language is $L_{bc} := a^+ b^n c^n$, where the number of *a*s is irrelevant and instead the number of *b*s and *c*s must match. It is easy to show that there is a CFG that generates L_{ab}, and similarly that there is a CFG that generates L_{bc}. Since we haven't discussed CFGs yet, we won't show those grammars here. The important thing is just that both L_{ab} and L_{bc} can be generated by CFGs.

But now consider what happens if we combine the conditions of both languages. This yields the language $L_{abc} := a^n b^n c^n$, where the number of *a*s must match the number of *b*s and the number of *b*s must match the number of *c*s. So *aabbcc* would be a string of L_{abc}, but not *abbcc*, *abbc*, or *aaabbc*. This language cannot be generated by any CFG. Even though each individual matching condition can be captured by a CFG, their combination cannot.

Fortunately, *n*-gram grammars are not like CFGs and can be combined in a very simple fashion.

Combining negative grammars

Suppose you have two negative bigram grammars that you want to combine into a single bigram grammar. The first thing to be clarified here is what we mean by combine. Given two grammars G and G', their combination could accept every string that is generated by both grammars, or every string that is generated by at least one grammar. The former is much more restrictive than the latter. It is like saying "each one of these grammars imposes a constraint, and we combine the constraints into a single grammar that enforces them all at once". The other option instead allows for violations: "there's a bunch of phenomena, and as long as a string exhibits at least one of them, it is well-formed". This is not how natural languages work. If a language has both word-final devoicing and intervocalic voicing, then a word has to satisfy both. You cannot say "Oh, I already did the intervocalic voicing, so let's not bother with the word-final devoicing". Language is like an exam where you get an F unless you answer every single question correctly.

Because of this, we only have to consider the stricter way of combining grammars, the one where the resulting grammar only generates strings that are well-formed with respect to each one of the original grammars. We also say that the language of this grammar is the **intersection** of the languages generated by the original grammars. Such a combined grammar is actually very easy to build: construct a new negative n-gram grammar that contains every n-gram that belongs to at least one of the original negative grammars. In set-theoretic terms, this amounts to taking the **union** of the two negative grammars.

Example 2

Suppose that we have a negative bigram grammar G_1 for word-final devoicing that contains only the bigram $s⋉$. In set notation, $G_1 := \{s⋉\}$. This grammar rules out every string that ends in an s. Furthermore, let G_2 be a negative trigram grammar for intervocalic voicing. It contains the trigrams *asa*, *asi*, *isa*, and *isi* (a realistic grammar would of course require more than just those four trigrams as the language would have more than just those two vowels). So G_2 forbids every string where s occurs between the vowels a and i.

In order to obtain a grammar that enforces both constraints, one simply constructs a new grammar G that contains all these n-grams: $s⋉$, *asa*, *asi*, *isa*, *isi*. Every string that is ruled out by G_1 or G_2 is also ruled out by G. And every string that is allowed by G is allowed by both G_1 and G_2. If desired, we can convert G from a mixed grammar to a fixed one using the familiar padding procedure from an earlier unit.

We can express this insight in terms of a formal theorem.

Theorem 1. Let G_1 be a negative m-gram grammar, G_2 a negative n-gram grammar (m and n may be the same). Then $L(G_1) \cap L(G_2) = L(G_1 \cup G_2)$.

Let us look in detail at how this theorem uses mathematical notation to express our idea about combining grammars. Recall that $L(G)$ is the set of strings that are well-formed with respect to G. The intersection $A \cap B$ of two sets A and B contains all elements that occur in both A and B. This is in contrast to the union $A \cup B$, which contains all elements that occur in at least one of the two, A or B. So $L(G_1) \cap L(G_2)$ is the set of strings that belong to both $L(G_1)$ and $L(G_2)$. In other words, these strings

are well-formed with respect to both G_1 and G_2. The theorem tells us that these are exactly the strings that are well-formed with respect to some other grammar that is the union of G_1 and G_2. This makes sense because we have defined n-gram grammars as finite sets, so set-theoretic operations like union can be applied to these grammars like to any other set.

That negative n-gram grammars are this easy to combine is a huge advantage. It means that when faced with a complex system like natural language phonotactics, we can safely decompose it into simpler subsystems and design a grammar for each one of them. The overall system is then obtained in a purely mechanical fashion by taking the union of the grammars for these subsystems. Not only does this greatly simplify the linguistic analysis step, it also helps with real-world implementations. For each phenomenon one designs a small grammar that can be easily tested for correctness, and only once all components have been individually verified does one combine them into the full model.

Exercise 1 Write two negative grammars such that the first only generates *ab* and *ba*, whereas the latter generates all strings of the form *ab*, *aab*, *aaab*, and so on. Then build the corresponding combined grammar. What is the language generated by the combined grammar? Is this the correct result?

Extension to positive grammars via De Morgan's Law

Combining grammars by taking their union only gives the desired result for negative grammars. For positive grammars, one has to take their intersection instead. (And since positive grammars always require a fixed n-gram length, one has to make sure first that the n-grams have the same length across all grammars.)

Theorem 2. Let G_1 and G_2 be positive n-gram grammars. Then $L(G_1) \cap L(G_2) = L(G_1 \cap G_2)$.

Think about this theorem for a moment to figure out the intuition behind it.

...

...

...

Done? Okay, then here is a very different way to think about it. The set operations of intersection and union are intimately connected via another operation called relative complement. The complement of B with respect to A, written $A - B$, contains all elements of A that aren't also members of B. Given some fixed universe U, one usually writes $\overline{A}$ instead of $U - A$. It then holds that $A \cap B = \overline{\overline{A} \cup \overline{B}}$ and $A \cup B = \overline{\overline{A} \cap \overline{B}}$. This is known as **De Morgan's law**.

Example 3 Let $A = \{a, b, c\}$ and $B := \{b, c, d\}$, while the fixed universe is $U := \{a, b, c, d, e\}$. Then $A \cup B = \{a, b, c\} \cup \{b, c, d\} = \{a, b, c, d\}$. Following De Morgan's law, we can also compute it in a stepwise fashion as $\overline{\overline{A} \cap \overline{B}}$:

- $\overline{A} = U - A = \{a, b, c, d, e\} - \{a, b, c\} = \{d, e\}$
- $\overline{B} = U - B = \{a, b, c, d, e\} - \{b, c, d\} = \{a, e\}$
- $\overline{A} \cap \overline{B} = \{e\}$

- $\overline{\overline{A} \cap \overline{B}} = U - (\overline{A} \cap \overline{B}) = \{a, b, c, d, e\} - \{e\} = \{a, b, c, d\}$

As you can see, $A \cup B = \{a, b, c, d\} = \overline{\overline{A} \cap \overline{B}}$.

Exercise 2 Compute $A \cap B$ in the same fashion.

De Morgan's law is very important as it extends from sets to logic and algebra, where it allows for some very nifty tricks. It can also greatly simplify proofs. But how does De Morgan help us understand that $L(G_1) \cap L(G_2) = L(G_1 \cap G_2)$ when G_1 and G_2 are positive grammars?

Well, it's because positive and negative grammars are connected by relative complement, as we already know. Given a n-gram grammar G of some polarity, $\Sigma_E^n - G$ is the equivalent grammar of opposite polarity. This is all we need to derive the theorem for positive grammars from the one for negative ones. Suppose ${}^+G_1$ and ${}^+G_2$ are positive grammars, and ${}^-G_1 = \Sigma_E^n -^+ G_1 =^- \overline{G_1}$ and ${}^-G_2 = \Sigma_E^n -^+ G_2 =^- \overline{G_2}$ the equivalent negative grammars. The notation is, to put it politely, a teeny-weeny confusing here: we first take a positive grammar ${}^+G_1$, then construct its set-theoretic complement $\overline{G_1}$, and then interpret that as a negative grammar ${}^-\overline{G_1}$.

Exercise 3 Suppose ${}^+G$ is a positive grammar containing the bigrams $\rtimes a$, $\underline{aa}$, and $a\ltimes$. Assume furthermore that all symbols must be a. Compute $\overline{G}$, then verify for yourself that ${}^-\overline{G}$ is the negative grammar that accepts the same strings as ${}^+G$.

So given some grammar ${}^+G$, we can always do polarity conversion to get the equivalent ${}^-\overline{G}$. Similarly, we can convert ${}^-G$ to ${}^+\overline{G}$, and the other way round. Note that we can do that even if G is itself defined via a more complex expression. For instance, ${}^-(G_1 \cup G_2)$ can be polarity converted to ${}^+\overline{G_1 \cup G_2}$. Furthermore, the polarity markers distribute over $\cap$ and $\cup$ in a systematic fashion. For example, ${}^+(G_1 \cap G_2)$ is equivalent to ${}^+G_1 \cap^+ G_2$. The former says that there is a positive grammar that is the intersection of G_1 and G_2, the latter describes the intersection of the positive grammars ${}^+G_1$ and ${}^+G_2$. Since intersection does not change polarity, this is the same as ${}^+(G_1 \cap G_2)$. The same is true if we replace $+$ with $-$, or $\cap$ with $\cup$.

Given all these equivalences, plus De Morgan's law, we have

$$
\begin{aligned}
& L({}^+G_1) \cap L({}^+G_2) && \textit{starting point} \\
=& L({}^-\overline{G_1}) \cap L({}^-\overline{G_2}) && \textit{polarity conversion} \\
=& L({}^-\overline{G_1} \cup^- \overline{G_2}) && \textit{Theorem 1} \\
=& L({}^-(\overline{G_1} \cup \overline{G_2})) && \textit{distributivity of polarity} \\
=& L({}^+\overline{(\overline{G_1} \cup \overline{G_2})}) && \textit{polarity conversion} \\
=& L({}^+(G_1 \cap G_2)) && \textit{De Morgan's Law} \\
=& L({}^+G_1 \cap^+ G_2)) && \textit{distributivity of polarity}
\end{aligned}
$$

Whew, that's quite some symbol salad. You might have to read it several times before it even vaguely starts making sense to you. But that's okay, math often takes

some time before it clicks. And if you still can't work your way through it after half an hour, just put it aside for now and come back later in the semester when you're more comfortable with arguments of this sort.

Recap

- *n*-gram grammars can be combined in an automatic fashion. If each component of a system can be described by an *n*-gram grammar, the whole system can be described by an *n*-gram grammar.
- Negative grammars are combined via union.
- Positive grammars are combined via intersection.
- The duality between negative grammars and union on the one hand and positive grammars and intersection on the other follows from **De Morgan's law**: $A \cap B = \overline{\overline{A} \cup \overline{B}}$