N-gram models of grammaticality

Exercise 1.
Consider the word *supercalifragilisticexpialidocious*. For each one of the following, say whether it is a bigram of the word.

- fr
- z
- doci
- pail
- sit
- co
- super

Solution

- fr: yes
- z: no
- doci: no
- pail: no
- sit: no
- co: no
- super: no

Explanation
We assume for this exercise that the relevant symbols are the characters of the English alphabet that are used to spell *supercalifragilisticexpialidocious*, rather than the sounds that are uttered when pronouncing the word.

Bigrams consist of exactly two symbols. Consequently, none of the following are bigrams:

- z (only 1 symbol)
- doci (4 symbols)
- pail (4 symbols)
- sit (3 symbols)
- super (5 symbols)

As these are not bigrams, they cannot be bigrams of *supercalifragilisticexpialidocious*. That leaves us with only two contenders:

- fr
- co

The bigrams of the word *supercalifragilisticexpialidocious* are *ag, al, ca, ce, ci, do, er, ex, fr, gi, ia, ic, id, if, il, io, is, li, oc, ou, pe, pi, ra, rc, st, su, ti, up, us, xp*. As you can see, *fr* is in this list, but *co* is not. Among all the items listed in the exercise, then, only *fr* is a bigram.

Exercise 2.
Consider once more the word *supercalifragilisticexpialidocious*. Which one of the following is among its bigrams (with edge markers):

- fr
- z
- ⋊⋉
- ⋊s
- s⋉⋉

Solution

- fr: yes
- z: no
- ⋊⋉: no
- ⋊s: yes
- s⋉⋉: no

Explanation
Once we consider edge markers, the long list of bigrams from the previous solution gains two new members: ⋊*s* and *s*⋉. Bigrams that were already on the original list without edge markers are still bigrams, so *fr* is still a bigram of *supercalifragilisticexpialidocious*. Similarly *n*-grams that do not consist of exactly two symbols are still not possible bigrams, which rules out *z* (only 1 symbol) and *s*⋉⋉ (3 symbols). This leaves us with ⋊⋉ and ⋊s. The latter is one of the two new members, but the former is not. In fact, ⋊⋉ can never be a bigram of any string that consists of one or more symbols.

Exercise 3.
Consider once more the word *supercalifragilisticexpialidocious*. For each one of the following, say whether it is a bigram of the word (with edge markers), a trigram, a 4-gram, or none of those choices.

- ⋊fr
- z
- do⋉c
- s⋉⋉⋉
- sit
- cious
- ⋊sup

Solution

- ⋊fr: none of those choices

- z: none of those choices
- do⋉c: none of those choices
- s⋉⋉⋉: 4-gram
- sit: none of those choices
- cious: none of those choices
- ⋊sup: 4-gram

Explanation

- ⋊fr: the word starts with *su*, not *fr*, so ⋊*fr* does not occur in the word
- z: this is a unigram, but we are only considering bigrams, trigrams, and 4-grams; and even if we did, the word does not contain *z* at all
- do⋉c: this is a 4-gram but it can never occur in any string because it is impossible to have any symbol after ⋉
- s⋉⋉⋉: this is a 4-gram; for 4-grams, we have to assume at least three edge markers on each side of the string, and since the word ends in *s*, the string with edge markers must indeed contain s⋉⋉⋉
- sit: this is a trigram, but it does not occur in the word
- cious: this is a 5-gram that occurs in the word, but we are only considering bigrams, trigrams, and 4-grams
- ⋊sup: this is a 4-gram and since the word starts with *sup*, ⋊*sup* is indeed a 4-gram of the string once we include edge markers

Exercise 4.
Suppose a language has the vowels *a* and *u*, the voiced consonants *z* and *v*, and the voiceless consonants *s* and *f*. The language also has intervocalic voicing, which means that a voiceless consonant may not occur immediately between two vowels. Write an *n*-gram grammar that expresses this fact.

Solution

The grammar consists of the following forbidden trigrams: *asa, asu, usa, usu, afa, afu, ufa, ufu*.

Explanation

We want to describe intervocalic voicing by a list of forbidden *n*-grams. The first question is what value we should pick for *n*. In general, we want to forbid strings that contain patterns of the form VsV and VfV, where V is a shorthand for our vowels *a* and *u*. Crucially, there is nothing wrong with patterns like Vs, sV, Vf, fV — *s* and *z* are allowed to occur next to a vowel, it is only a problem when they are sandwiched between vowels. Strings like *af*, *afzi*, or *afsi* should be allowed, but not *afi*. With bigrams, we cannot make this distinction. For example, the ill-formed *afi* contains only bigrams that also occur in the well-formed *afzizfi*, which makes it impossible to write a bigram grammar that rules out *afi* but still permits *afzizfi*.

Hence we take a step up from bigrams to trigrams, and then it becomes a simple matter of listing all trigrams where the first and third position are filled by vowels, and the second is filled by a voiceless consonant.

Exercise 5.
Consider the formal language where all strings are sequences of a, b, and c such that

- every string starts with a
- no string ends with c
- a and c are always separated by at least two symbols.

Write a negative n-gram grammar for this language such that all n-grams have the same length.

Solution

- $\rtimes\rtimes$b
- $\rtimes\rtimes$c
- c$\ltimes\ltimes$
- ac$\ltimes$
- aca
- acb
- acc
- ca$\ltimes$
- caa
- cab
- cac
- abc
- cba

Explanation

This exercise is a little tricky because we have to consider all three constraints first before we can decide on a value for n.

Requiring that every string starts with a would only require bigrams: $\rtimes b$ and $\rtimes c$ make it impossible to start with b and c, and $\rtimes\ltimes$ rules out the empty string, so we can only start with a. And the bigram $c\ltimes$ is all that is needed to ensure that no string ends with c.

The third condition, however, cannot be expressed with bigrams. If a and c are always separated by at least two symbols, then we cannot have a and c next to each other, and we cannot have them in positions with only one symbol between them. For example, we cannot have abc as part of any string. This requires us to use trigrams. The grammar has to contain every trigram where a is in position one and c is in position two, or the other way round:

- ac⋉
- aca
- acb
- acc
- ca⋉
- caa
- cab
- cac

We could also add all trigrams where *a* is in position two and *c* in position three, or the other way round. But this would be redundant. Suppose our sliding window moves through the string from left to right, and that at step *s* the sliding window sees a trigram with *a* in the second position and *c* in the third position. But then at the very next step, $s + 1$, the sliding window moves one step to the right and *a* is now in the first position and *c* in the second position. In other words, if *a* and *c* are ever adjacent in the string, then it is guaranteed that the string contains some trigram that has *a* and *c* as its first two symbols, and hence it is sufficient to rule out just these trigrams.

Our trigram grammar now rules out adjacent *a* and *c*, but we also have to list all the cases where *a* and *c* are separated by one symbol, whatever that symbol may be. Once again, though, we can be smart about this. We do not need to consider any trigrams of the form *axc* or *cxa* where *x* is *a* or *c* since those are already ruled out by the trigrams that forbid adjacent *a* and *c*. But then the only remaining option for *x* is *b*, giving us just two additional trigrams:

- abc
- cba

Alright, these 10 trigrams handle the third condition that *a* and *c* must be separated by at least two symbols. But since we had to use trigrams to handle this condition, the exercise requires us to state the first two conditions in terms of trigrams, too. Fortunately, that's not too difficult. Instead of ⋊*b* and ⋊*c*, we forbid ⋊⋊*b* and ⋊⋊*c*. And instead of *c*⋉, we forbid *c*⋉⋉. Again we do not need to add trigrams like *ac*⋉ because every string that contains this trigram also contains *c*⋉⋉, which means that the former is redundant if the grammar already contains the latter.