Unit 2

Syntax is not Strictly Local Over Strings

1 Definitions

Definition 2.1 (k-grams). Let k be some natural number. A *k-gram*, or *k-factor*, over alphabet Σ is an element of $(\Sigma \cup \{\rtimes, \ltimes\})^k$. Given a string w over Σ, its *k-augmented* counterpart $\hat{w}_k := \rtimes^{k-1} \cdot w \cdot \ltimes^{k-1}$ consists of w with $k-1$ left edge markers and $k-1$ right edge markers, and its set k-grams(w) of k-grams is defined as k-grams(w) := $\left\{s \in (\Sigma \cup \{\rtimes, \ltimes\})^k \mid \exists u, v \in \Sigma^* \text{ s.t. } u \cdot s \cdot v = \hat{w}\right\}$.

Definition 2.2 (Strictly Local Languages). A finite set of k-grams is called a *strictly k-local grammar*. A positive strictly k-local grammar G generates the language $L(G) := \{w \mid k\text{-grams}(w) \subseteq G\}$. A negative strictly k-local grammar G generates the language $L(G) := \{w \mid k\text{-grams}(w) \cap G = \emptyset\}$. A language L is *strictly k-local* iff it is generated by some strictly k-local grammar. The class of *strictly local languages* is given by $\bigcup_{k \geq 1} \{L \mid L \text{ is strictly } k\text{-local}\}$.

Positive SL grammars can be translated to negative ones and *vice versa*.

Definition 2.3 (Local Substring Substitution Closure). A language L satisfies *k-local substring substitution closure* iff there is some $k \geq 1$ such that if L contains both $u \cdot x \cdot v$ and $u' \cdot x \cdot v'$, where x has length $k-1$, then L also contains $u \cdot x \cdot v'$.

Theorem 2.4. A language is in SL_k iff it satisfies k-substring substitution closure. ⌟

2 Application to Syntax

At first glance, substring substitution closure seems to be a welcome property because it naturally gives rise to iteration of patterns.

(1) a. John thinks John likes Mary.

b. (John thinks)$^+$ John likes Mary.

(2) a. His father's father died

b. His (father's)$^+$ father died

But in fact substring substitution closure also adds many ungrammatical sentences. This shows that syntax is not strictly local.

(3) a. John likes Mary.

b. Mary likes John.

c. * John likes Mary likes John.

(4) a. The man is dangerous.

b. The men are dangerous.

c. The man that assaulted the men (that assaulted the men)$^+$ is dangerous.

d. The men that assaulted the man (that assaulted the man)$^+$ are dangerous.

e. * The man ... are dangerous.

f. * The man ... is dangerous.

Note that we can also give examples that do not rely on overt agreement (which some linguists might be inclined to regard as extra-syntactic).

(5) a. (John thinks that)$^+$ Mary left.

b. (John thinks that)$^+$ Mary left which man.

c. Which man does John think that (John thinks)$^+$ Mary left.

d. * Which man does John think that (John thinks)$^+$ Mary left which man.

Exercise 2.1. There are many more constructions in English and other natural languages that can be used to show that natural languages are not closed under suffix substitution closure. Give an example using the format familiar from above. ⊙

Our argument against SL grammars hinges on the assumption that the string sets to be generated are infinite. With finite languages the problems above can be circumvented by picking a sufficiently large k. But in practice those ks will have to be huge, and that creates an enormous learnability problem. Suppose English had a measly 1000 words (its lexicon is much larger than that). Then the number of possible k-grams scales rapidly.

k	**Formula**	**Total**	**exceeds...**
2	1000^2	1,000,000	
3	1000^3	1,000,000,000	
5	1000^5	1 quintillion	number of cells in human body
10	1000^{10}	10^{30}	Avogadro constant
30	1000^{30}	10^{90}	number of atoms in the universe

Even Google with its massive corpora and enormous statistical know-how can only handle 5-grams at this point. It strains credibility that a human learner could operate with the large k-values that would be required to capture the grammaticality distinctions native speakers are capable of.

Exercise 2.2. Write an SL grammar that generates the finite language containing all of the sentences below, and only those. Try to keep the size of the k-factors as small as possible.

- Mary likes John.
- John thinks John likes Mary.
- John thinks Mary thinks John likes Mary.
- John thinks John thinks Mary likes John. ⊙

Exercise 2.3. SL grammars can be made more powerful by introducing a notion of locality via tiers. Such a tier-based SL grammar (TSL grammar) over alphabet Σ is a pair $\langle G, T \rangle$ such that

- $T \subseteq \Sigma$ is the tier alphabet, and
- G is a strictly local grammar.

From T we construct a function del_T that deletes all symbols from a string that do not belong to the tier alphabet. If the remainder is a string generated by G, the original string is generated by the TSL grammar. More formally, the generated language is $\{s \mid \text{del}_T(s) \in L(G)\}$.

Explain why the problems with SL grammars pointed out above also exist with TSL grammars. ⊙

References and Further Reading

Steedman, Mark. 2001. *The syntactic process*. Cambridge, MA: MIT Press.