Morphological Parsing

- Morphology-to understand how a word is formed
- Morphological Parsing-It is used to find morphemes form a word.
- Morphemes contain Stem(Root word) and Affix(Prefix(e.g. reform),infix(e.g. passersby) and suffix(e.g. nationalist).
- Morphological Parser decides the order of words:

➢ Lexicons-Stem,affix,part of speech(noun,adjective,verb)

➢ Morphotactics:decides which morphemes should come based on rules.

➢ Orthographic rules: lady+s=ladys(wrong)

lady+ies=ladies(true)

- Types of Morphemes:

1. Free Morphemes: Independent word having its own meaning(e.g. camera)

a) Lexical Morphemes:Adjective/Noun/Verb/Picture word(e.g. Yellow)

b) Grammatical Morphemes:Conjunction(e.g. and, or)

2. Bound Morphemes: No meaning of its own.(E.g. –ing (running))

a) Inflectional Morphemes:Words when combined with free morphemes it will not change the part of speech.IT will be always added as suffix.

Example: cat + s=cats

b) Derivational Morphemes:Words when combined with free morphemes it will change the part of speech.

Example: danger+ ous= dangerous

3. Allomorphes: antonym

Happy x unhappy

Rational x irrational

Possible x impossible

- Finite State Morphological Parsing

Let's now proceed to the problem of parsing English morphology. Consider a simple example: parsing just the productive nominal plural (*-s*) and the verbal progressive (*-ing*). Our goal will be to take input forms like those in the first column below and produce output forms like those in the second column.

Input	Morphological Parsed Output
cats	`cat +N +PL`
cat	`cat +N +SG`
cities	`city +N +PL`
geese	`goose +N +PL`
goose	(`goose +N +SG`) or (`goose +V`)
gooses	`goose +V +3SG`
merging	`merge +V +PRES-PART`
caught	(`catch +V +PAST-PART`) or (`catch +V +PAST`)

The second column contains the stem of each word as well as assorted morphological **features**. These features specify additional information about the stem. For example the feature `+N` means that the word is a noun; `+SG` means it is singular, `+PL` that it is plural. We will discuss features in Chapter 11; for now, consider `+SG` to be a primitive unit that means 'singular'. Note that some of the input forms (like *caught* or *goose*) will be ambiguous between different morphological parses.

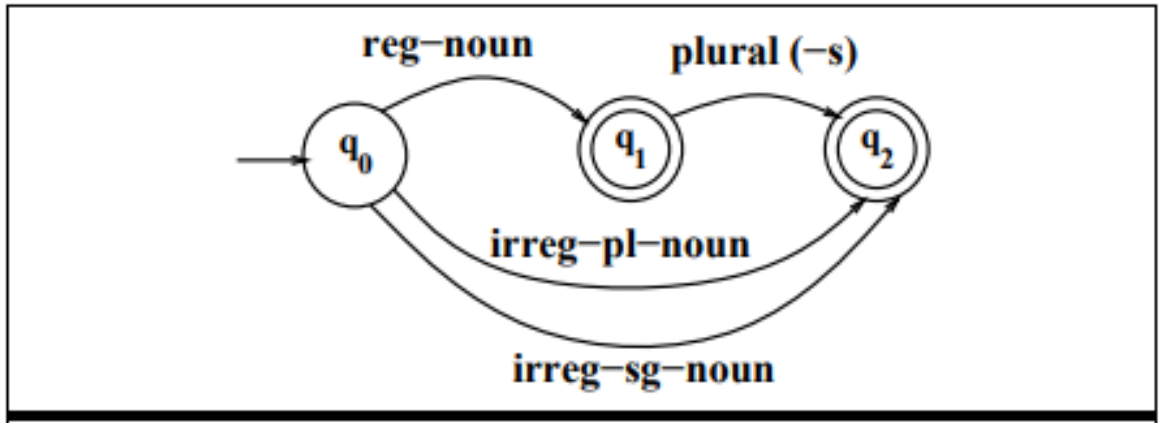

Figure 3.2 A finite-state automaton for English nominal inflection.

reg-noun	irreg-pl-noun	irreg-sg-noun	plural
fox	geese	goose	-s
cat	sheep	sheep	
dog	mice	mouse	
aardvark			

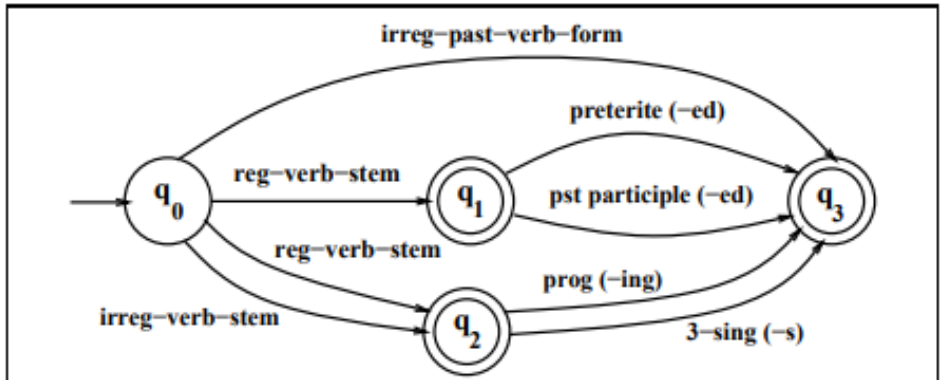

Figure 3.3 A finite-state automaton for English verbal inflection

Morphological Parsing with Finite-State Transducers(Pg 102 Jurafsky ebook)

- Two-level morphology, first proposed by Koskenniemi (1983).
- Two level morphology represents a word as a correspondence between a lexical level, which represents a simple concatenation of morphemes making up a word, and the surface level, which represents the actual spelling of the final word.
- Morphological parsing is implemented by building mapping rules that map letter sequences like cats on the surface level into morpheme and features sequences like cat +N +PL on the lexical level. Figure shows these two levels for the word cats. Note that the lexical level has the stem for a word, followed by the morphological information +N +PL which tells us that cats is a plural noun.

- The automaton that we use for performing the mapping between these two levels is the finite-state transducer or FST.
- A transducer maps between FST one set of symbols and another; a finite-state transducer does this via a finite automaton.
- Thus we usually visualize an FST as a two-tape automaton which recognizes or generates pairs of strings.
- The FST thus has a more general function than an FSA; where an FSA defines a formal language by defining a set of strings, an FST defines a relation between sets of strings. This relates to another view of an FST; as a machine that reads one string and generates another

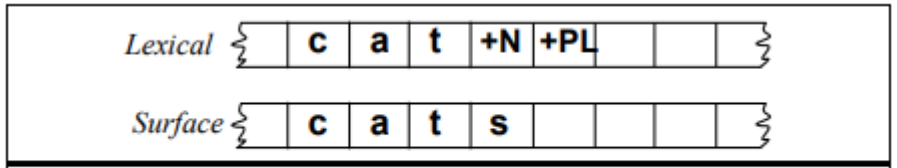

Figure 3.8 Example of the lexical and surface tapes.

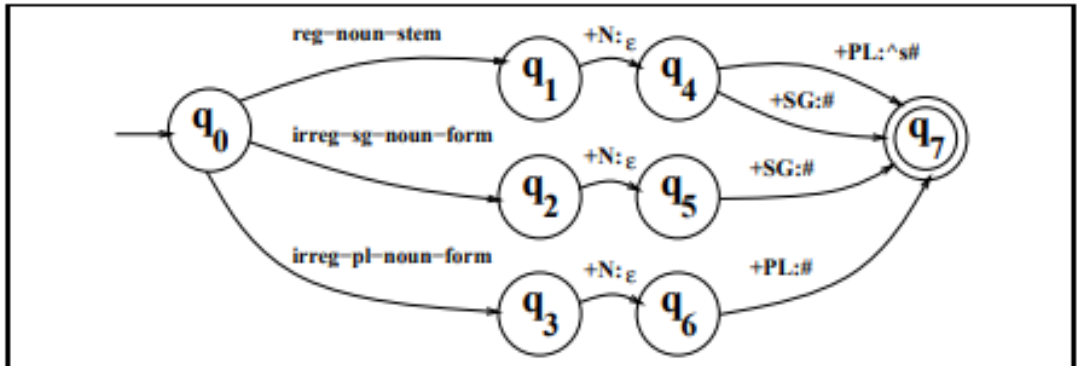

Figure 3.9 A transducer for English nominal number inflection T_{num}. Since both q_1 and q_2 are accepting states, regular nouns can have the plural suffix or not. The morpheme-boundary symbol ˆ and word-boundary marker # will be discussed below.

That is, c maps to itself, as do a and t, while the morphological feature +N (recall that this means 'noun') maps to nothing (ε), and the feature +PL (meaning 'plural') maps to ˆs. The symbol ˆ indicates a morpheme boundary, while the symbol # indicates a word boundary.

This transducer will map plural nouns into the stem plus the morphological marker +PL, and singular nouns into the stem plus the morpheme +SG. Thus a surface *cats* will map to `cat +N +PL` as follows:

```
c:c a:a t:t +N:ε +PL:^s#
```

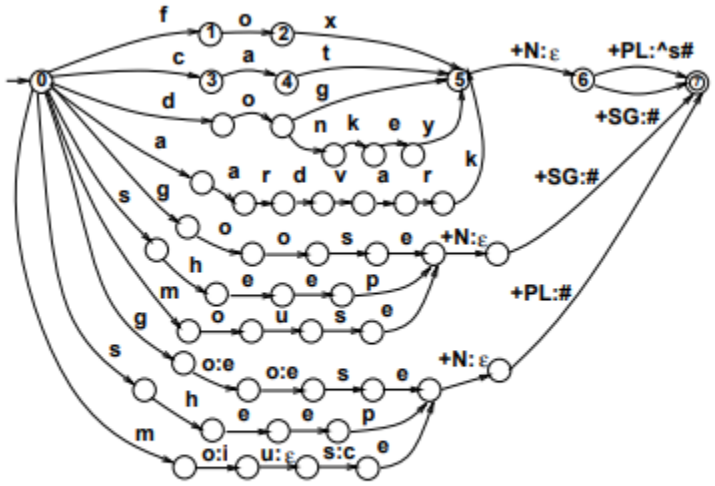

Figure 3.11 A fleshed-out English nominal inflection FST $T_{lex} = T_{num} \circ T_{stems}$.

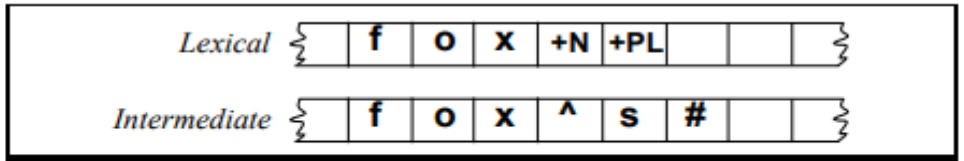

Figure 3.12 An example of the lexical and intermediate tapes.

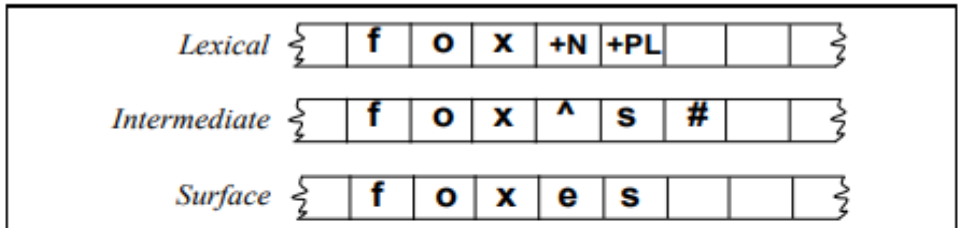

Figure 3.13 An example of the lexical, intermediate and surface tapes. Between each pair of tapes is a 2-level transducer; the lexical transducer of Figure 3.11 between the lexical and intermediate levels, and the E-insertion spelling rule between the intermediate and surface levels. The E-insertion spelling rule inserts an *e* on the surface tape when the intermediate tape has a morpheme boundary ˆ followed by the morpheme *-s*.

Orthographic Rules and Finite-State Transducers

Name	Description of Rule	Example
Consonant doubling	1-letter consonant doubled before *-ing*/*-ed*	beg/begging
E deletion	Silent e dropped before *-ing* and *-ed*	make/making
E insertion	e added after *-s*,*-z*,*-x*,*-ch*, *-sh* before *-s*	watch/watches
Y replacement	*-y* changes to *-ie* before *-s*, *-i* before *-ed*	try/tries
K insertion	verbs ending with *vowel* + *-c* add *-k*	panic/panicked

Language Model

- Language modeling is the way of determining the probability of any sequence of words. Language modeling is used in a wide variety of applications such as Speech Recognition, Spam filtering, etc.
- An N-gram language model predicts the probability of a given N-gram within any sequence of words in the language. A good N-gram model can predict the next word in the sentence i.e the value of $p(w|h)$.
- Two types of Language Modelings:
- **Statistical Language Modelings**: Statistical Language Modeling, or Language Modeling, is the development of probabilistic models that are able to predict the next word in the sequence given the words that precede. Examples such as N-gram language modeling.
- **Neural Language Modelings**: Neural network methods are achieving better results than classical methods both on standalone language models and when models are incorporated into larger models on challenging tasks like speech recognition and machine translation. A way of performing a neural language model is through word embeddings.

- **N-gram**
- N-gram can be defined as the contiguous sequence of n items from a given sample of text or speech. The items can be letters, words, or base pairs according to the application. The N-grams typically are collected from a text or speech corpus (A long text dataset).
- **N-gram Language Model:**
- An N-gram language model predicts the probability of a given N-gram within any sequence of words in the language. A good N-gram model can predict the next word in the sentence i.e the value of p(w|h)
- Example of N-gram such as unigram (“This”, “article”, “is”, “on”, “NLP”) or bi-gram (‘This article’, ‘article is’, ‘is on’,’on NLP’)
- **Perplexity**: Perplexity is a measure of how good a probability distribution predicts a sample. It can be understood as a measure of uncertainty.
- Perplexity=P(S) raise to (-1/n)

- **Maximum Likelihood Estimate**: It is the method of estimating the parameter of an assumed probability distribution,given some observed data.It is the value that makes the observed data the “most probable”.

$$P(Wi|Wi-1)=Count(Wi-1,Wi)/Count(Wi-1)$$

- **Laplace Smoothening:**Also called Add one smoothening.It is a smoothening technique that helps to tackle the problem of zero probability of a word in the text.When a particular bigram never occurred in our corpus data,then we get probability zero for that word.And when probability of any word is zero the overall effect is zero,which waste the contribution of other words.To avoid this we can use Laplace smoothening.

- Practice numericals based on N gram model.