Module-2
Word Level Analysis

Prepared By

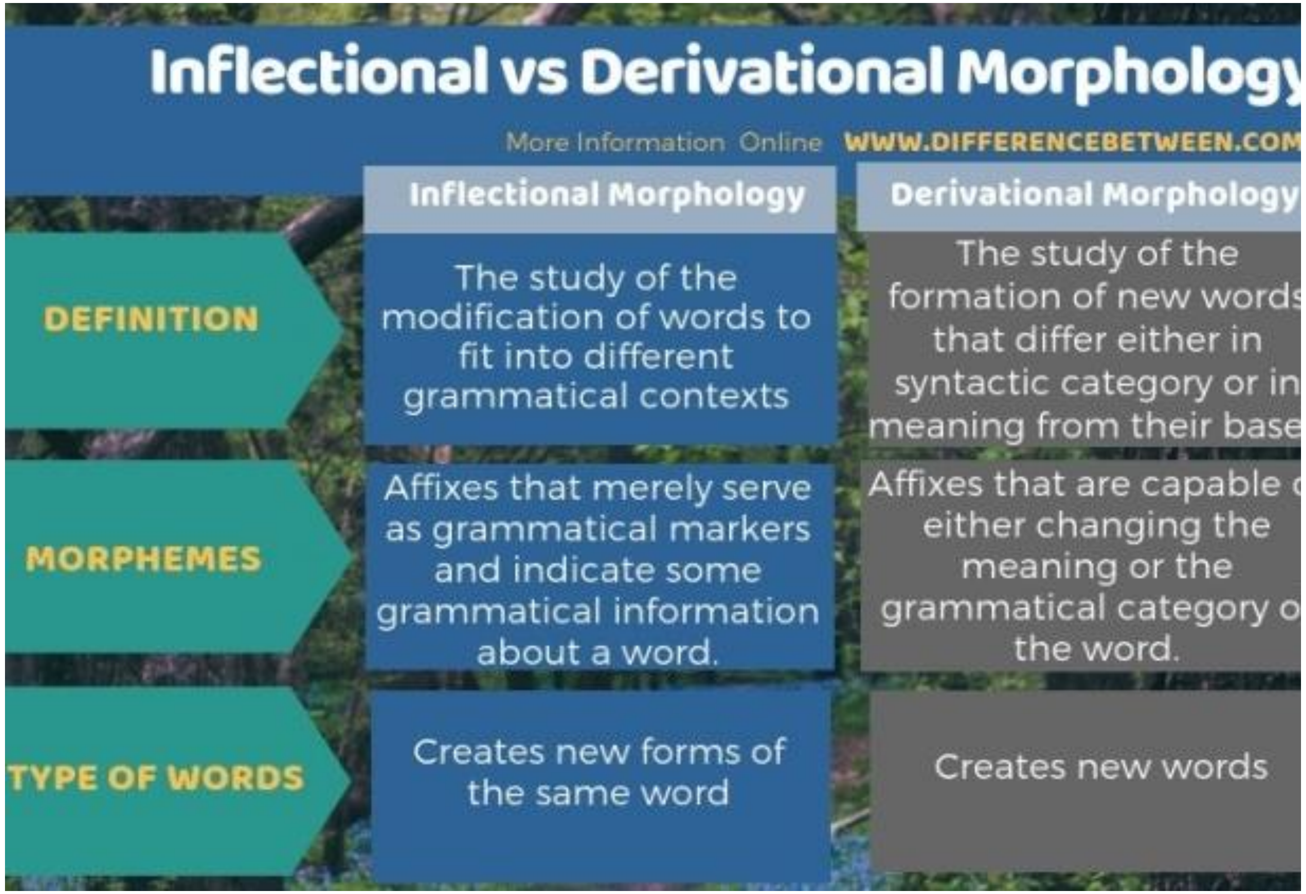
Prof. Suja Jayachandran

Reference :

1.Speech and Language Processing by Daniel Jurafsky,James and Martin

Word: Fundamental building block of any language . Also called orthographic tokens(ie separated by white space). Exception Chinese, and Japanese languages are not separated by white space.

Morphology:deals with syntax of complex words and parts of word called morphemes as well as what semantic property they convey.



- **Tokenization**: Tokenization is the process of tokenizing or splitting a string, or text into a list of tokens. One can think of tokens as parts like a word is a token in a sentence, and a sentence is a token in a paragraph. It is important because the meaning of the text can be interpreted through analysis of the words present in the text.

➢ Token: Non-unique words of a sentence

➢ Type: Unique words of a sentence.

- **Stopword Removal**: Stop word removal is one of the most commonly used preprocessing steps across different NLP applications. The idea is simply to remove the words that occur commonly across all the documents in the corpus. Typically, articles and pronouns are generally classified as stop words. These words have no significance in some of the NLP tasks like information retrieval and classification.

- Normalization is the process of converting a token into its base form.
- It helps in :

➢ Reducing the number of unique tokens present in the text.

➢ Reducing variation of words in the text.

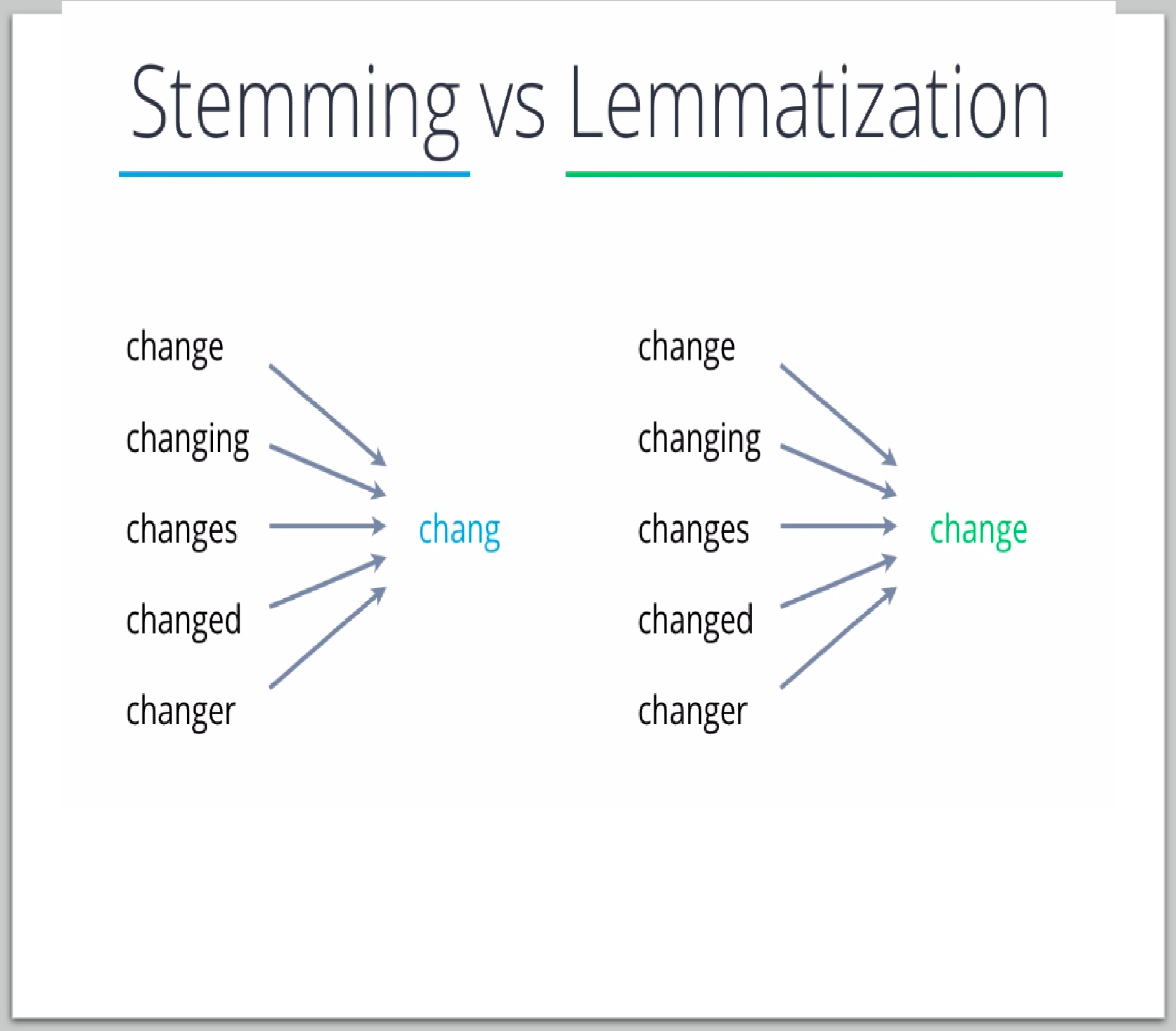
➢ Removing redundant information.

- Stemming: It is an elementary rule-based process for removing inflectional forms from a token and output are the stem/root of the word.
- Eg: “Laughing”, “Laughed” , “Laughs” , “Laugh” will all become “laugh”
- It can produce words that are not part of the dictionary.

Eg: “His teams are not winning”.

After stemming: “Hi”, “team”, “are”, “not”, “winn”

- Lemmatization: It is a systematic step-by-step process for removing inflection forms of a word.
- It makes use of vocabulary, word structure, part of speech tags, and grammar relation.
- Output of lemmatization is the root word called a lemma.
- E.g. “Running”, ”Run”, and “Ran” will be Run.



Types of Stemming Algorithm

- **1. <u>Porter Stemmer – PorterStemmer()</u>**
- Martin Porter invented the Porter Stemmer or Porter algorithm in 1980. Five steps of word reduction are used in the method, each with its own set of mapping rules. Porter Stemmer is the original stemmer and is renowned for its ease of use and rapidity. Frequently, the resultant stem is a shorter word with the same root meaning.
- PorterStemmer() is a module in NLTK that implements the Porter Stemming technique.
- **2. <u>Snowball Stemmer – SnowballStemmer()</u>**
- Martin Porter also created Snowball Stemmer. The method utilized in this instance is more precise and is referred to as “English Stemmer” or “Porter2 Stemmer.” It is somewhat faster and more logical than the original Porter Stemmer.
- SnowballStemmer() is a module in NLTK that implements the Snowball stemming technique.
- **3. <u>Lancaster Stemmer – LancasterStemmer()</u>**
- Lancaster Stemmer is straightforward, although it often produces results with excessive stemming. Over-stemming renders stems non-linguistic or meaningless.
- LancasterStemmer() is a module in NLTK that implements the Lancaster stemming technique.
- **4. <u>Regexp Stemmer – RegexpStemmer()</u>**
- Regex stemmer identifies morphological affixes using regular expressions. Substrings matching the regular expressions will be discarded.
- RegexpStemmer() is a module in NLTK that implements the Regex stemming technique.

- Regular Expression

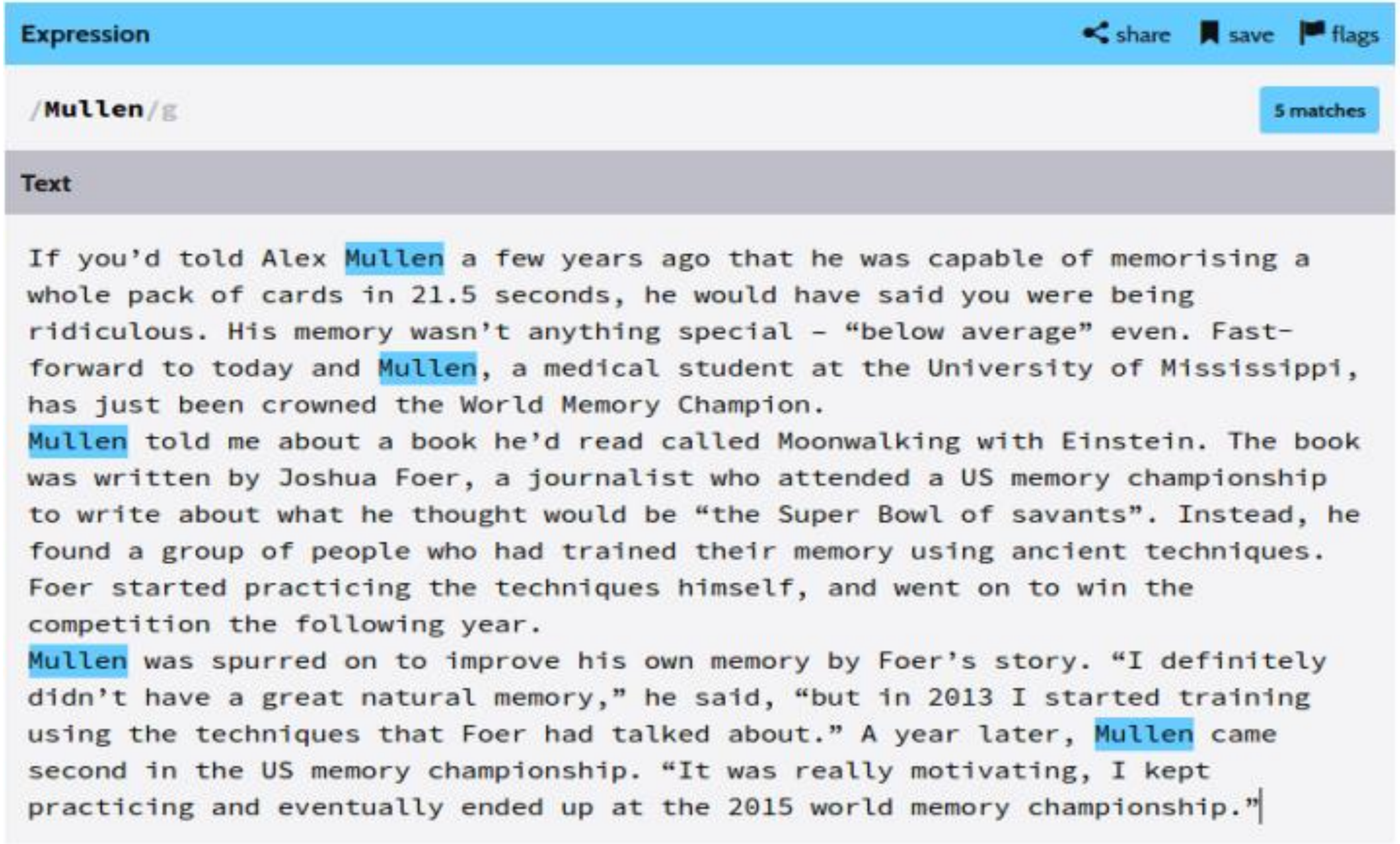
- A regular expression (RE) is a language for specifying text search strings. RE helps us to match or find other strings or sets of strings, using a specialized syntax held in a pattern.
- Regular expression search requires a pattern that we want to search for, and a corpus of texts to search through.
- A regular expression search CORPUS function will search through the corpus returning all texts that contain the pattern
- Properties of Regular Expressions

➢ American Mathematician Stephen Cole Kleene formalized the Regular Expression language.

➢ RE is a formula in a special language, which can be used for specifying simple classes of strings, a sequence of symbols. In other words, we can say that RE is an algebraic notation for characterizing a set of strings.

➢ Regular expression requires two things, one is the pattern that we wish to search and other is a corpus of text from which we need to search.

Searching for a particular word


Expression share save flags

/Mullen/g 5 matches

Text

If you'd told Alex Mullen a few years ago that he was capable of memorising a whole pack of cards in 21.5 seconds, he would have said you were being ridiculous. His memory wasn't anything special – "below average" even. Fast-forward to today and Mullen, a medical student at the University of Mississippi, has just been crowned the World Memory Champion.
Mullen told me about a book he'd read called Moonwalking with Einstein. The book was written by Joshua Foer, a journalist who attended a US memory championship to write about what he thought would be "the Super Bowl of savants". Instead, he found a group of people who had trained their memory using ancient techniques. Foer started practicing the techniques himself, and went on to win the competition the following year.
Mullen was spurred on to improve his own memory by Foer's story. "I definitely didn't have a great natural memory," he said, "but in 2013 I started training using the techniques that Foer had talked about." A year later, Mullen came second in the US memory championship. "It was really motivating, I kept practicing and eventually ended up at the 2015 world memory championship."

- Searching a number

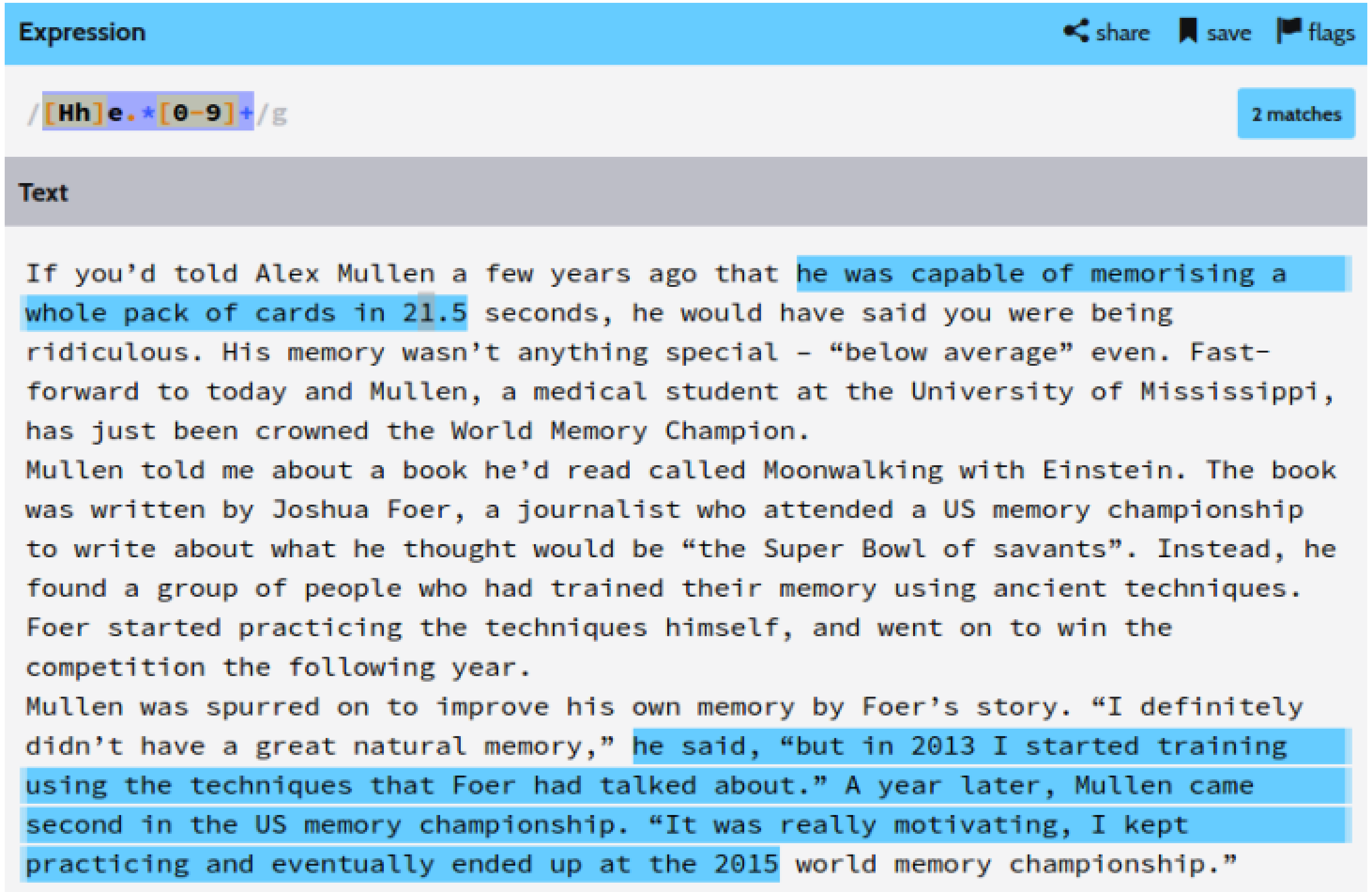
Expression share save flags

`/[0-9]+[.]?[0-9]/g` 3 matches

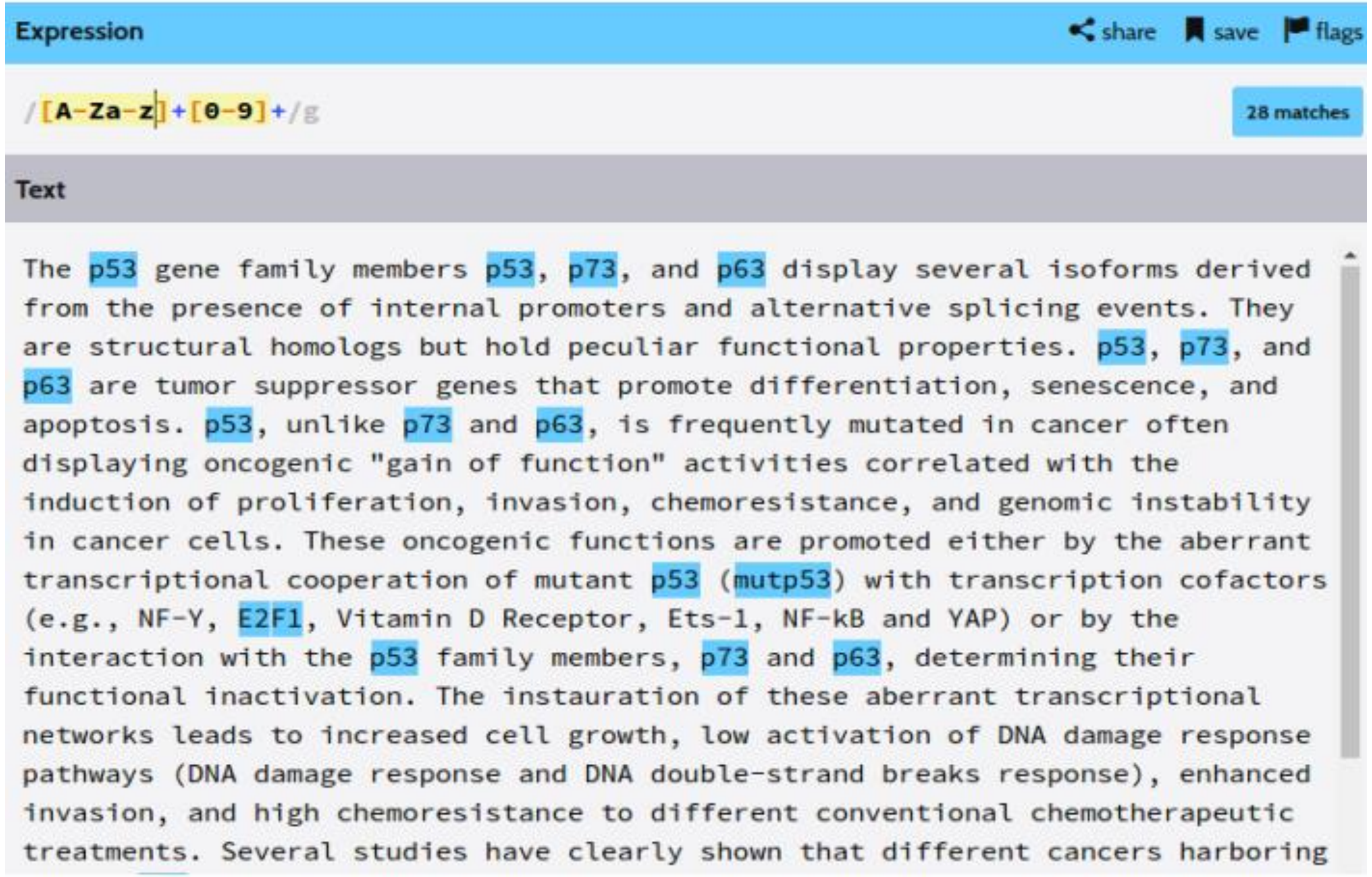
Text

If you'd told Alex Mullen a few years ago that he was capable of memorising a whole pack of cards in 21.5 seconds, he would have said you were being ridiculous. His memory wasn't anything special – "below average" even. Fast-forward to today and Mullen, a medical student at the University of Mississippi, has just been crowned the World Memory Champion.
Mullen told me about a book he'd read called Moonwalking with Einstein. The book was written by Joshua Foer, a journalist who attended a US memory championship to write about what he thought would be "the Super Bowl of savants". Instead, he found a group of people who had trained their memory using ancient techniques. Foer started practicing the techniques himself, and went on to win the competition the following year.
Mullen was spurred on to improve his own memory by Foer's story. "I definitely didn't have a great natural memory," he said, "but in 2013 I started training using the techniques that Foer had talked about." A year later, Mullen came second in the US memory championship. "It was really motivating, I kept practicing and eventually ended up at the 2015 world memory championship."

- Searching passages with number



- Named entity recognition



RE:Build an FSA that modeled the subpart of English dealing with amounts of money. Such a formal language would model the subset of English consisting of phrases like ten cents, three dollars, one dollar thirty-five cents and so on

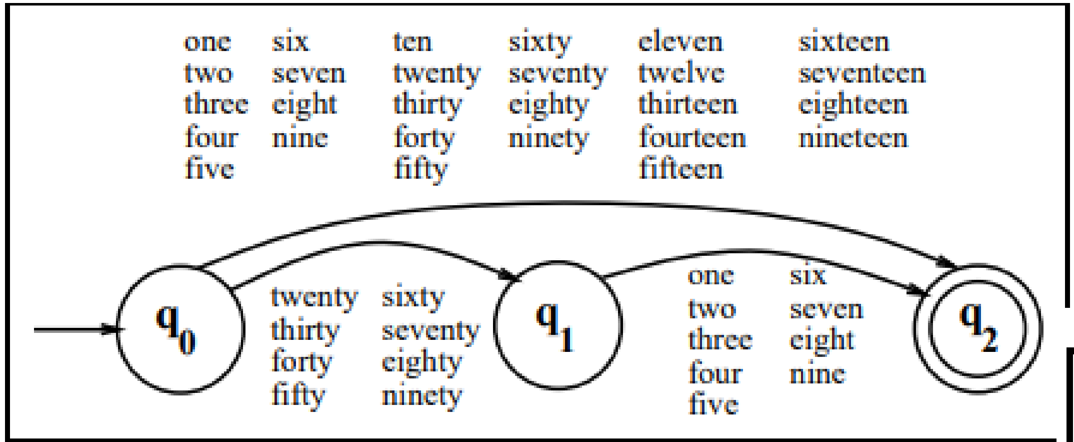


Figure 2.16 An FSA for the words for English numbers 1 – 99.

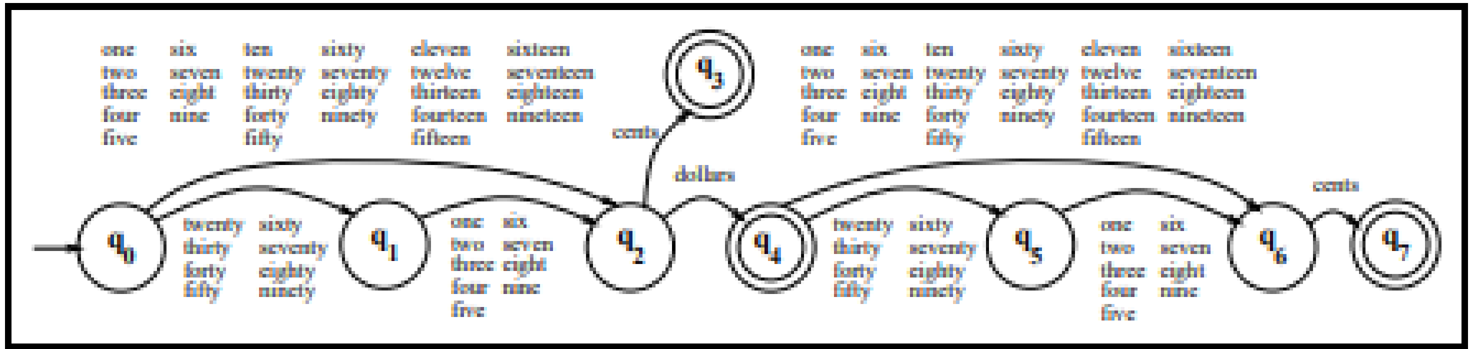


Figure 2.17 FSA for the simple dollars and cents.

Porter’s Algorithm

- Porter (1980) algorithm we define a consonant as a letter other than A, E, I, O, and U, and other than Y preceded by a consonant. Any other letter is a vowel. (This is of course just an orthographic approximation.)
- Let c denote a consonant and v denote a vowel. C will stand for a string of one or more consonants, and V for a string of one or more vowels.
- Any written English word or word part can be represented by the following regular expression (where the parentheses () are used to mark optional elements): (C)(VC) m (V)
- For example the word troubles maps to the following sequence: troubles C V C VC with no final V. We call the Kleene operator m the measure of any word or word part; the measure correlates very roughly with the number of syllables in the word or word part.

- Steps:

Step 1: Plural Nouns and Third Person Singular Verbs

The rules in this set do not have conditions:

SSES → SS	caresses → caress
IES → I	ponies → poni ties → ti
SS → SS	caress → caress
S → ε	cats → cat

Step 2a: Verbal Past Tense and Progressive Forms

(m> 1) EED → EE	feed → feed agreed → agree
(*v*) ED → ε	plastered → plaster bled → bled
(*v*) ING → ε	motoring → motor sing → sing

Step 2b: Cleanup

If the second or third of the rules in 2a is successful, we run the following rules (that remove double letters and put the E back on -ATE/-BLE)

AT → ATE	conflat(ed) → conflate
BL → BLE	troubl(ing) → trouble
IZ → IZE	siz(ed) → size
(*d & !(*L or *S or *Z)) → single letter	hopp(ing) → hop tann(ed) → tan fall(ing) → fall hiss(ing) → hiss fizz(ed) → fizz
(m=1 & *o) → E	fail(ing) → fail fil(ing) → file

Step 3: Y → I

(*v*) Y → I	happy → happi sky → sky

Step 4: Derivational Morphology I: Multiple suffixes

(m > 0) ATIONAL → ATE	relational → relate
(m > 0) TIONAL → TION	conditional → condition rational → rational
(m > 0) ENCI → ENCE	valenci → valence
(m > 0) ANCI → ANCE	hesitanci → hesitance
(m > 0) IZER → IZE	digitizer → digitize
(m > 0) ABLI → ABLE	conformabli → conformable
(m > 0) ALLI → AL	radicalli → radical
(m > 0) ENTLI → ENT	differentli → different
(m > 0) ELI → E	vileli → vile
(m > 0) OUSLI → OUS	analogousli → analogous
(m > 0) IZATION → IZE	vietnamization → vietnamize
(m > 0) ATION → ATE	predication → predicate
(m > 0) ATOR → ATE	operator → operate
(m > 0) ALISM → AL	feudalism → feudal
(m > 0) IVENESS → IVE	decisiveness → decisive
(m > 0) FULNESS → FUL	hopefulness → hopeful
(m > 0) OUSNESS → OUS	callousness → callous
(m > 0) ALITI → AL	formaliti → formal
(m > 0) IVITI → IVE	sensitiviti → sensitive
(m > 0) BILITI → BLE	sensibiliti → sensible

Step 5: Derivational Morphology II: More multiple suffixes

$(m > 0)$	ICATE	$\to$ IC	triplicate	$\to$	triplic
$(m > 0)$	ATIVE	$\to$ ε	formative	$\to$	form
$(m > 0)$	ALIZE	$\to$ AL	formalize	$\to$	formal
$(m > 0)$	ICITI	$\to$ IC	electriciti	$\to$	electric
$(m > 0)$	FUL	$\to$ ε	hopeful	$\to$	hope
$(m > 0)$	NESS	$\to$ ε	goodness	$\to$	good

Step 6: Derivational Morphology III: single suffixes

$(m > 1)$	AL	$\to$ ε	revival	$\to$	reviv
$(m > 1)$	ANCE	$\to$ ε	allowance	$\to$	allow
$(m > 1)$	ENCE	$\to$ ε	inference	$\to$	infer
$(m > 1)$	ER	$\to$ ε	airliner	$\to$	airlin
$(m > 1)$	IC	$\to$ ε	gyroscopic	$\to$	gyroscop
$(m > 1)$	ABLE	$\to$ ε	defensible	$\to$	defens
$(m > 1)$	ANT	$\to$ ε	irritant	$\to$	irrit
$(m > 1)$	EMENT	$\to$ ε	replacement	$\to$	replac
$(m > 1)$	MENT	$\to$ ε	adjustment	$\to$	adjust
$(m > 1)$	ENT	$\to$ ε	dependent	$\to$	depend
$(m > 1)$	(*S or *T) & ION	$\to$ ε	adoption	$\to$	adopt
$(m > 1)$	OU	$\to$ ε	homologou	$\to$	homolog
$(m > 1)$	ISM	$\to$ ε	communism	$\to$	commun
$(m > 1)$	ATE	$\to$ ε	activate	$\to$	activ
$(m > 1)$	ITI	$\to$ ε	angulariti	$\to$	angular
$(m > 1)$	OUS	$\to$ ε	homologous	$\to$	homolog
$(m > 1)$	IVE	$\to$ ε	effective	$\to$	effect
$(m > 1)$	IZE	$\to$ ε	bowdlerize	$\to$	bowdler

Step 7a: Cleanup

$(m > 1)$	E	$\to$ ε	probate rate	$\to$ $\to$	probat rate
$(m = 1$ & ! *o)	E	$\to$ ε	cease	$\to$	ceas

Step 7b: Cleanup

$(m > 1$ & *d *L)	$\to$	[single letter]	controll roll	$\to$ $\to$	control roll

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.

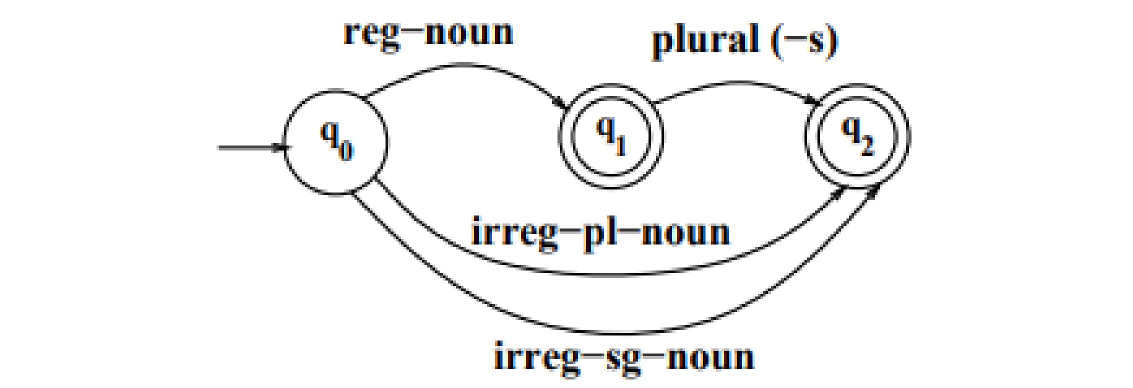


Figure 3.2 A finite-state automaton for English nominal inflection.

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

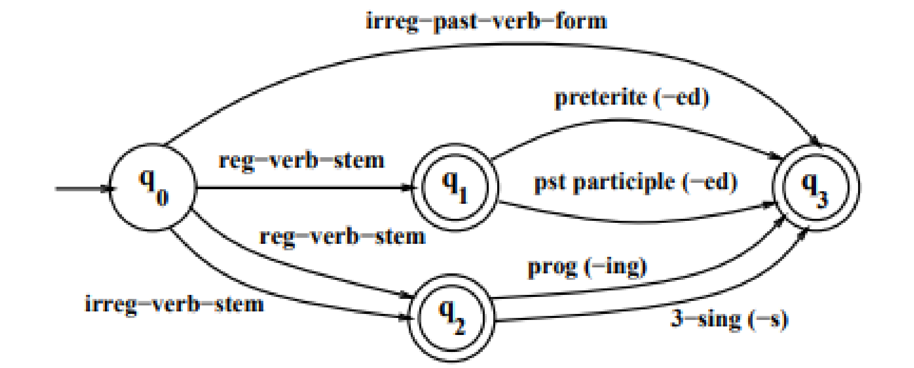


Figure 3.3 A finite-state automaton for English verbal inflection

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

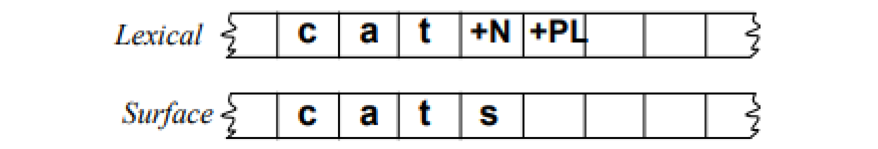


Figure 3.8 Example of the lexical and surface tapes.

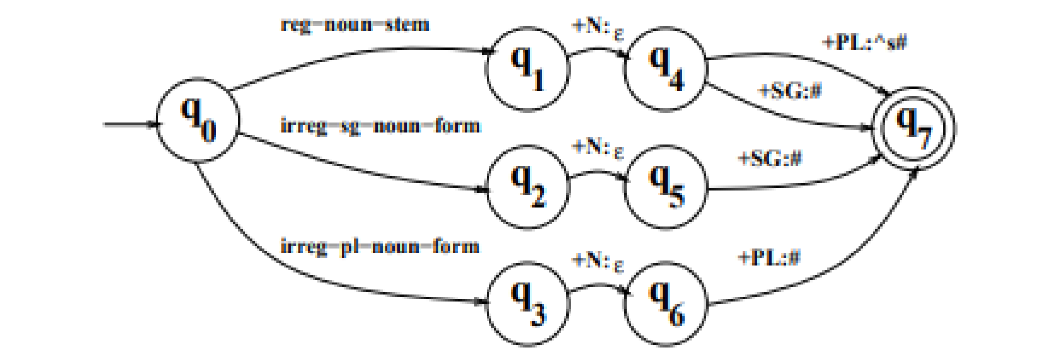


Figure 3.9 A transducer for English nominal number inflection T_{num}. Since both q_1 and q_2 are accepting states, regular nouns can have the plural suffix or not. The morpheme-boundary symbol ˆ and word-boundary marker # will be discussed below.

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:ˆs#
```

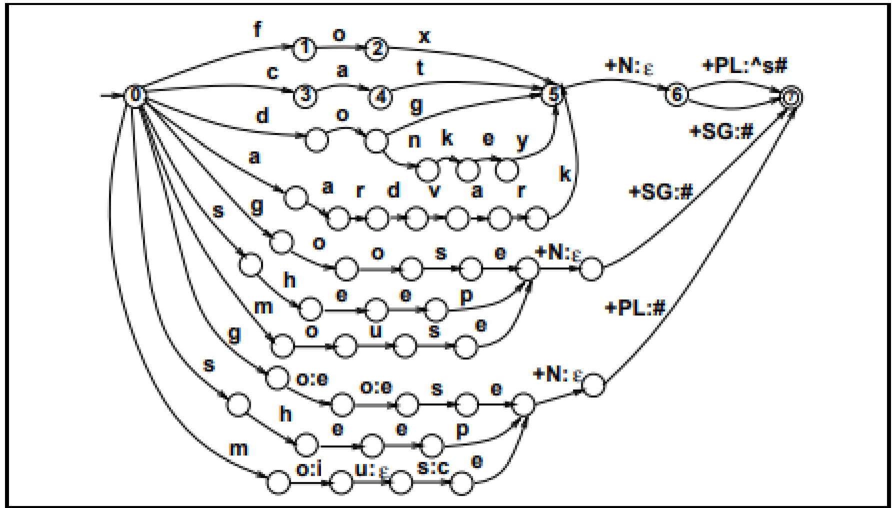


Figure 3.11 A fleshed-out English nominal inflection FST $T_{lex} = T_{num} \circ T_{stems}$.

Lexical	f	o	x	+N	+PL	
Intermediate	f	o	x	^	s	#

Figure 3.12 An example of the lexical and intermediate tapes.

Lexical	f	o	x	+N	+PL	
Intermediate	f	o	x	^	s	#
Surface	f	o	x	e	s	

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”.

$$P(W_i|W_{i-1})=Count(W_{i-1},W_i)/Count(W_{i-1})$$

- **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.

- Practicenumerical based on N-gram model.