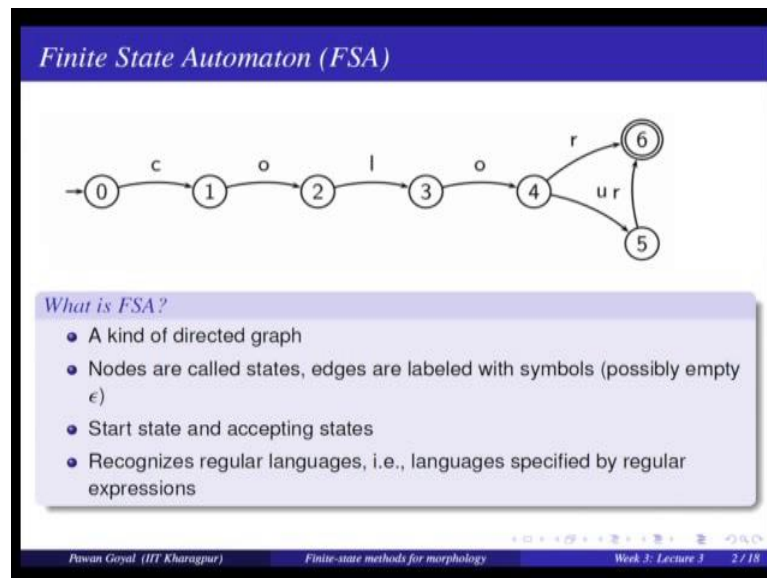


Natural language Processing
Prof. Pawan Goyal
Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur

Lecture - 13
Finite State Methods for Morphology

So welcome back for the third module of this week. So in the last lecture we were talking about computational morphology. So in this module we will talk about how do we use financial math methods for doing this morphological analysis. So I will just go in briefly into what these models do.

(Refer Slide Time: 00:42)



So starting with what is a finite state automaton. So if you have taken a course on formal languages automaton theory or theory of computation, you might already be aware of what is a finite automaton. I will just briefly tell what is this. If you have not taken a course, that course you might just want to quickly see that in one of the one of the books for that course.

So what is a finite state automaton? So it is kind of a directed graph. So in this figure you are seeing there is a finite automaton that is having 6 different nodes. So these are the nodes here are called the states. And there are edges between the nodes and they are the edges are labeled with certain symbols. So for example, you are seeing between node 0

to 1 there is in there is an edge with the label of c. And they may also be empty if for certain category of this automaton.

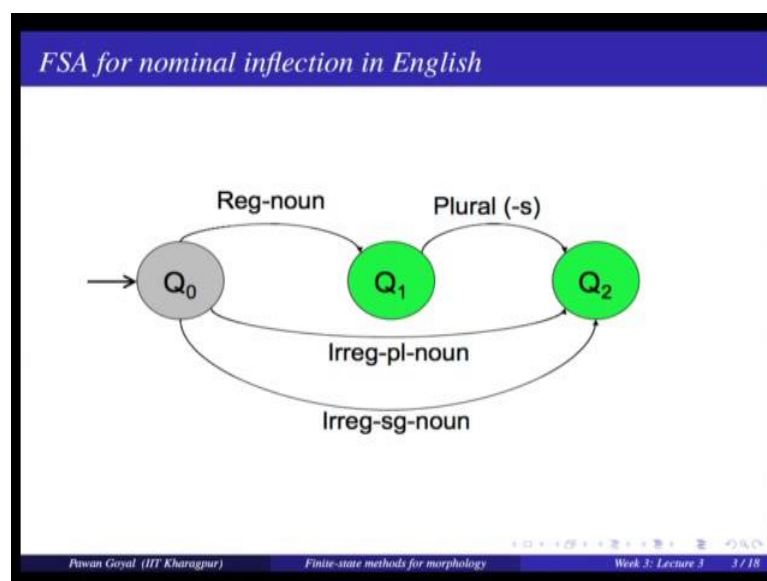
There are certain started states. So for example, here the node 0 is the started state, and there are certain accepting or final states that are noted by maybe having a double circles like on the node 6, you are having double circle this is an accepting state or the finite state. So they can be more than one final state very easily. So now, what do the finite state automaton do? So they recognize a regular language that is the language that is specified by the regular expressions. So any regular expression you can always convert into an automaton financial automaton.

So now if you see the automaton that is provided on this slide, what is the language that it recognizes? What are the words that will be passed through this? And by seeing that you can also see how the automaton actually works. So if you give an input like color to this automaton what will happen? So it will start with the started state, you will take the first character c from a started state on the input character c you will go to the state 1 from this graph. Now state 1 you take an input o that is the next in in the input and you move to state 2 and So on. And as you end with the with the phoneme character r you end up with the state 6 in this graph.

So given an input like see color it is accepted by this automaton, but what happens if I give a word like colour you can see that there is still a path. You can go from 4 to 5 and 5 to 6 and this automaton you can accept colour. So it will accept 2 words color colour, but what happens if you give it is some other word like colr. So you will see col will go to state 3, but if you take r there is no path from 3 that takes an input r. So it cannot move further.

So this input will not be accepted by this automaton. So given an automaton final state automaton what it does? It recognizes if a finite, so not finite a regular language. So if you give any string from the regular language it will accept; that means, this will end up in one of the accepting states, but you should give it any other string that is not in the language it will not accept. So this will not end up in any of the accepting states.

(Refer Slide Time: 04:18)

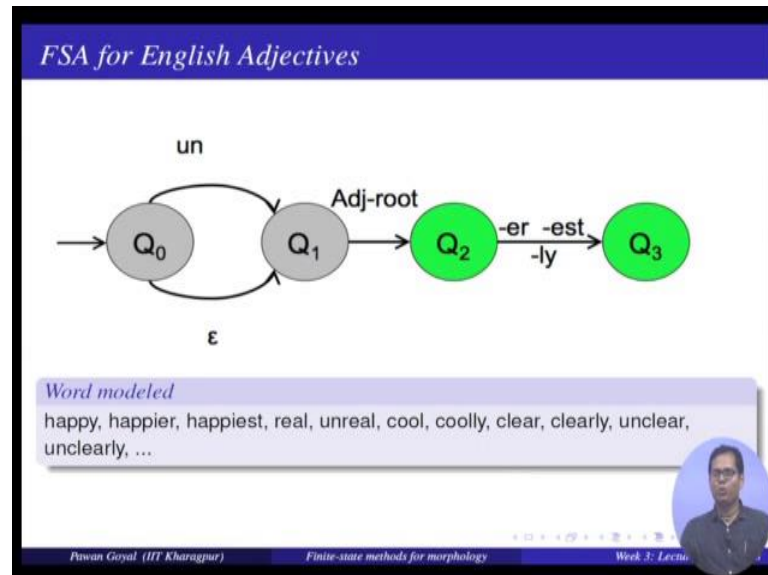


Now how are this finite state automaton used for doing the morphological analysis? So what is the idea? The idea is that when you combine various morphemes there are certain changes that happen at the boundary mainly concatenation. And this is very regular phenomenon. So if you want to capture that 2 words like boy and car can be made converted into plural by adding an s, I can simply have a state for the noun where boy and car both come together and then I have a single edge from there with s marking that there is a plural. And now you think about all the possible plural that you can make in English. All the singular nouns can come to the same state and then you have a single arrow that with s and that will convert into plural. So that very efficiently captures the process of converting in from singular to plural and as long as this is regular this is very straightforward then how do you do that.

So here is an example in this slide. So only 3 states are shown here Q0, Q1, Q2 - Q0 is the starter state and Q1 and Q2 are other accepting or final states. Many other intermediate states are not shown we will see that in some of the later slides. So from Q0 to Q1 you have the regular nouns like car, boy, bag etcetera and then so all of these are also words in English. You can now add a plural morpheme s to convert them into plural. So going from Q1 to Q2, but as such what happens if you have an irregular noun like goose to geese. So that cannot go to Q1 because to goose if you apply s you will not get its plural form. So for the irregular nouns you have a different path that goes from Q0 that it goes to Q2 again there will be some intermediate nodes

for individual characters of phonemes in in that in the language in the world. So both the singular plural will go to Q. So Q 1 and Q 2 both will accept the words in the English.

(Refer Slide Time: 06:35)



Similarly, this is in FSA finite state automaton for English adjectives. So you have some prefix like un then you have the actual adjective root like happy and then you have certain suffixes like er, est, ly. So you see here we are also showing some lot of morpho tactics that what kind of morphemes follows other kind of morphemes. So now, what kind of words that you can generate by using this automaton? So you can say un happy - unhappy. A starting from Q naught or from Q naught you taking absolute transition go to Q 1 and you have happy and er – happier, happiest, happily and so on. So sick is generate words like happy, happier, happiest real from Q naught you take an absolute transition go to Q 1 then you have the adjective root real that can Q 2 is a final state. So you can generate real. If you have generated unreal from Q naught you take un that will give you un plus have the word real and go to Q 2 you become, you get unreal. Similarly, all other words you can generate by using this particular automaton.


(Refer Slide Time: 07:49)

Morphotactics

- The last two examples model some parts of the English morphotactics
- But what about the information about regular and irregular roots?

Lexicon

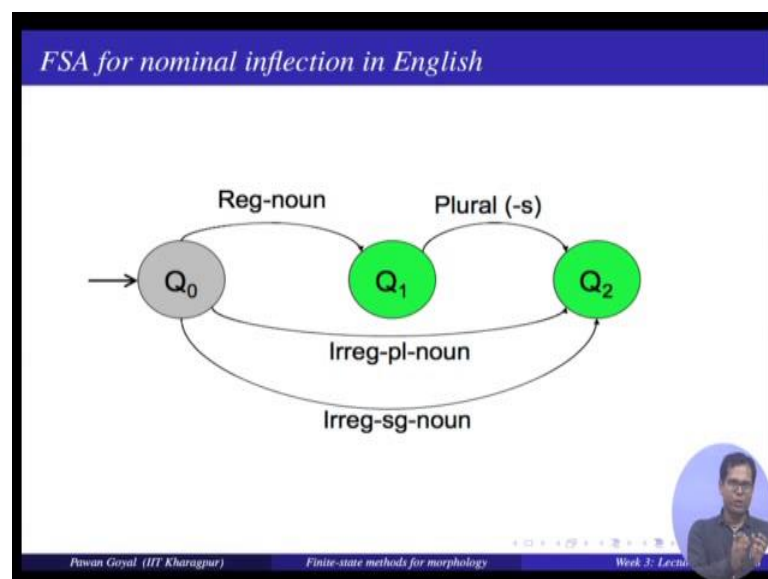
Can we include the lexicon in the FSA?



Pawan Goyal (IIT Kharagpur) Finite-state methods for morphology Week 3: Lecture 3

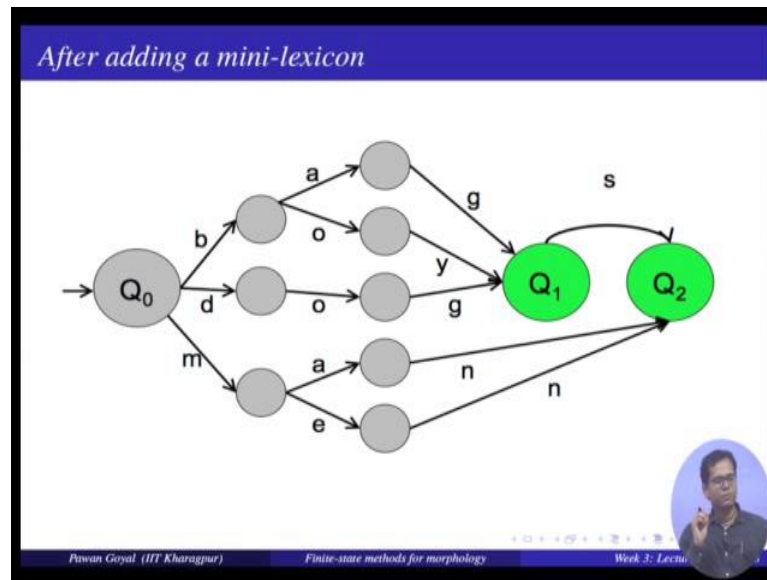
So as we already said in these examples we also seen some English morphotactics, that is what kind of morphemes come after another kind of morphemes in in English. Now but what do I do about regular in regular roots in English, how do I capture that information in this automaton? So can we include the lexicon also in my automaton? So that is what we will see. If I also want to include word like words like car, cars, bag everything in inside my automaton, how do I do that?

(Refer Slide Time: 08:29)



So this is what we have seen. So from Q_{naught} I take all the regular nouns to Q_1 all the irregular nouns directly go to Q_2 , but now I want to include the lexicon. So let us take a very simple and a small lexicon.

(Refer Slide Time: 08:42)

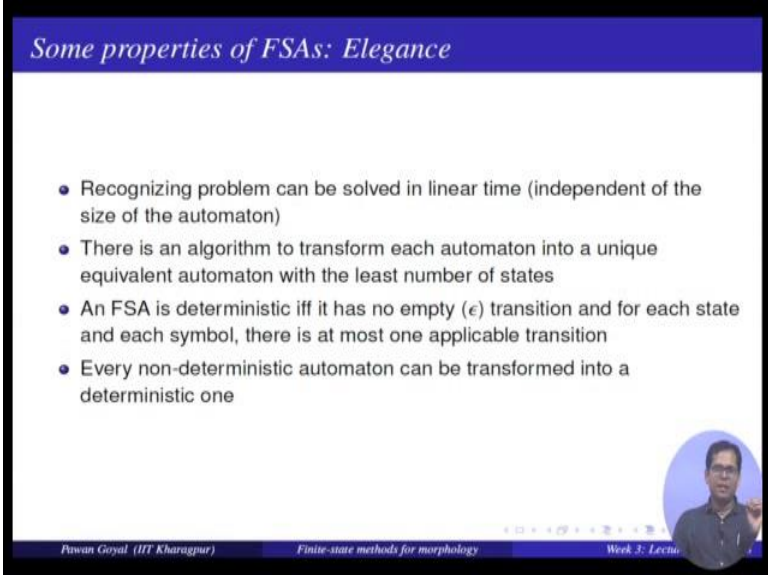


So I have the words like bag boy and dog as the regular noun and man as the irregular noun. And we want to generate all the, so I want to recognize all the singular and plural form. So what do I do? So you see here from the state Q_{naught} I am now having all the possible other individual states that take the regular singular nouns to Q_1 . So I have different nodes for b and then the b shared among bag and boy. Similarly, there is a different state for going to doc. And finally, all of these they go to Q_1 all the regular nouns in the singular form and then you get you can have s added to that this becomes plural.

So you have dogs, bags and boys together. What do you do for the irregular nouns you have a different path? Man and m a n and m e n they again go to Q_2 directly. Now given this figure can you recognize the words like boys easily, yes I start with Q_{naught} when I get the word b I go to the next state o, I go to another state y I go to Q_1 and s I go to Q_2 . So I can recognize boys I can recognize boy m a n, m e n all this can be recognized by this automaton. I can further expand it to include other words my vocabulary. But the education here is this; the goal of morphological asks that we started with. So what is this automaton doing? Given a word in English it will tell me whether it is a singular or

plural word in English. Assuming that, you have taken care of all the words in your vocabulary while building this automaton, so it can recognize various words.

(Refer Slide Time: 10:36)



The slide is titled "Some properties of FSAs: Elegance" in a blue header. It contains a bulleted list of four properties. In the bottom right corner, there is a small circular inset video of a man speaking. The footer of the slide includes the name "Pawan Goyal (IIT Kharagpur)", the course "Finite-state methods for morphology", and "Week 3: Lecture 3".

- Recognizing problem can be solved in linear time (independent of the size of the automaton)
- There is an algorithm to transform each automaton into a unique equivalent automaton with the least number of states
- An FSA is deterministic iff it has no empty (ϵ) transition and for each state and each symbol, there is at most one applicable transition
- Every non-deterministic automaton can be transformed into a deterministic one

Pawan Goyal (IIT Kharagpur) Finite-state methods for morphology Week 3: Lecture 3

So that is what we have written here. So what are the properties of FSAs? So they are very elegant. So that is the recognition problem can be solved in linear time. What do I mean by that? Given give me any input string. So you can find out whether automaton recognizes the string or not a linear time linear in the length of the string. Because every time you are making if you are having a word like boys you are checking if from the start state if you go, if you take input b where do you go o y s and then finally, if you end up in the accepted state, you will accept the string otherwise you will not, of course, this will happen only for the deterministic automaton, but as in the elegance of itself we know, that every non-deterministic finite automaton can be contributed to a deterministic finite automaton and there is in a simple algorithm for that.

So we do not have to worry about it. Even if we have started building an NFA I non-deterministic automaton I can convert it to DFA. Where the linear time you can find out whether a string is accepted or not. Similarly, I do not have to worry about getting the minimum mode of states of the automator, because the again there is an algorithm that converts any automaton into it the equivalent automaton that has the minimum number of states.

So this is why we say this is very elegant you can convert an NFA to DFA. And in DFA you can convert it into the minimum number of states since very elegant. And we have seen that it can work as a language recognizer given a regular language it can tell me a new given a new string whether this is in the language or not, but. So coming to my previous question is it what we need in the morphological analysis. So answer is no, this is not sufficient for the morphological analysis we, so take the word boys. Remember what are the different morphology analysis we talked about, we talked about finding the lambdaization, is like the simplest one lambdaization. I want to find out what is the lemma for the word boys. I want to find out the word boy, can the DFA help me to obtain the word boy? It cannot, it can only tell me that boys are a word in me in my defined regular language, this this is there. So this is either single plural, but it cannot tell whether boys is what is the lemma for the word boys. So I need some other more model that can give a word can also give me what is the lemma or the root form.

(Refer Slide Time: 13:28)

But ...

FSAs are language recognizers/generators.
We need transducers to build Morphological Analyzers

Finite State Transducers

- Translate strings from one language to strings in another language
- Like FSA, but each edge is associated with two strings

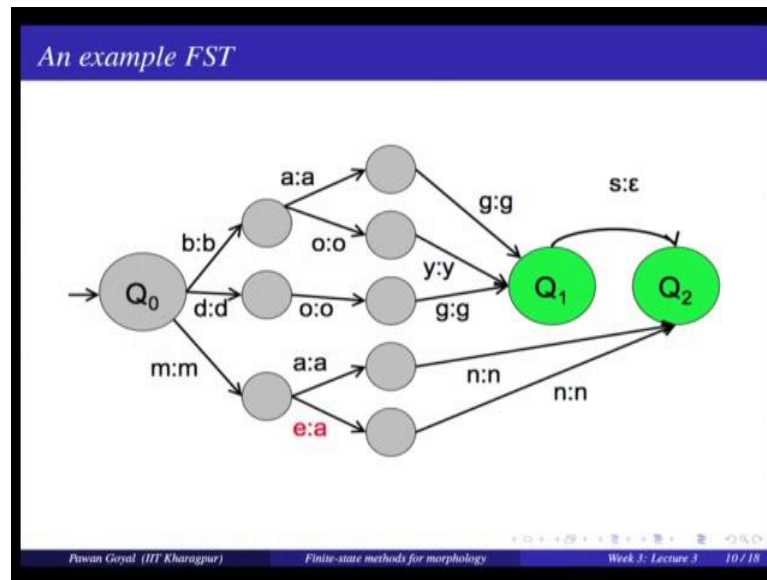
Pawan Goyal (IIT Kharagpur) Finite-state methods for morphology Week 3: Lecture 3 9 / 18

So for that, So FSAs are language recognizer or generators. So we need instead transducers that can help me do morphological analysis.

So what are transducers? So transducers are very similar to final state automaton. Except that they can help me translate one string into another. So now, what happens there? How do they do that? The model is very similar to FSA, but now in each edge of the FSA instead of having a single phoneme as the label I have the input phoneme or input

characters' symbol and the output character symbol. So it translates one-character symbol to another character symbol. So that solves my problem of going from the input string like boys to an output this string like boy.

(Refer Slide Time: 14:23)



So let us take the example on the same lexicon of forwards and they are plural. So earlier we saw the final decision automaton where we were having each edge label within input character only. So finally, it could recognize whether a word is there in the language or not, but now we have the transducer where each other is labeled with the input as well as the output character. So you see here. So here m, m e n and e is the input a is the out. So once you give an input like m e n to this transducer the output you will get is m e n the actual lemma. That is how they can solve the morphological analysis problem.


(Refer Slide Time: 15:11)

Two-level morphology

Given the input *cats*, we would like to output *cat+N+PL*, telling us that cat is a plural noun.

We do this via a version of **two-level morphology**, a correspondence between a lexical level (morphemes and features) to a surface level (actual spelling).

Lexical	c	a	t	+N	+PL		
Surface	c	a	t	s			

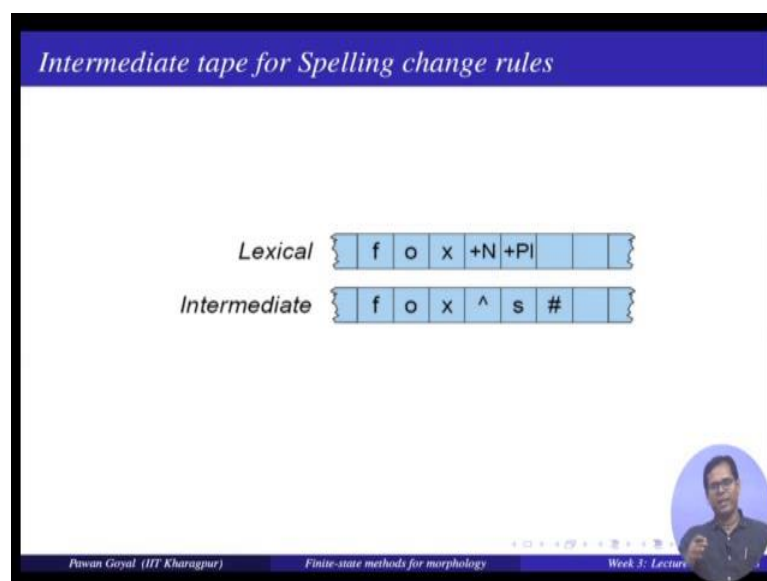


Pawan Goyal (IIT Kharagpur) Finite-state methods for morphology Week 3: Lectures

Now, some small details that what might be the problem that that you might face while doing that. So for example, here is an input. So we are calling it the lexical label, cat I want to convert it to its noun, but a plural form. So I know cat gives me cats, but what happens if I take a word like fox. Fox will give me foxes. So there will be they are changed at the boundary you adding an e between fox and s. So now, there are many ways in which you can handle it. One is you have you take cat has a regular noun and then have a regular plural to that and fox as irregular noun separately this is one possibility.

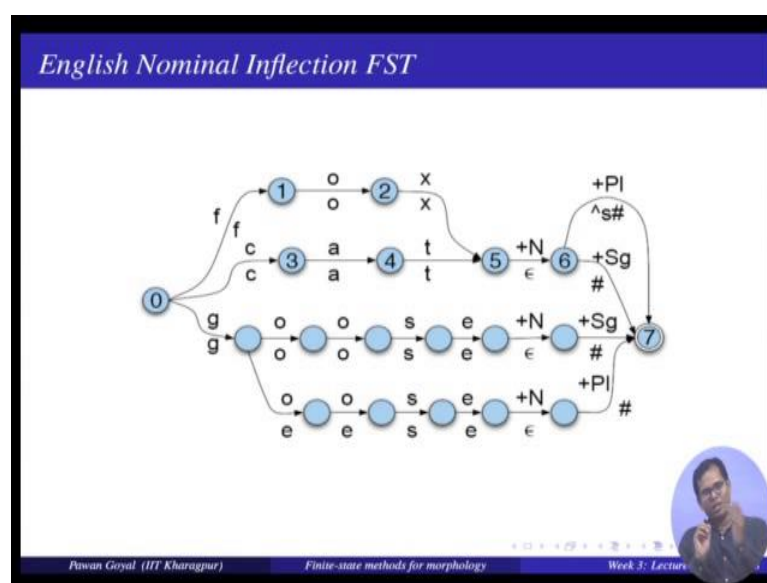
But the interesting idea is if you can use a 2 level morphology that is between the lexical form cat plus noun plus plural and the surface form, that is cats can I define an intermediate form. That is cat there is an s, but in between there is some placeholder that can be null or e depending on what are the phonemes available in the stem and the surface. So that is instead of going from lexical to surface level directly can I go to the individual level, first this is my first label and then I transfer from intermediate label to the surface level in the second level of morphology. So this is my 2 level morphologic.

(Refer Slide Time: 16:50)



So example is, so the motivation example is for cats and fox. So instead of going from fox plus n plus p l to foxes I go to fox and some placeholder and an s and this is the end of word. Now I can have a second label rule that says if my (Refer Time: 17:06) fox of x is h, s will there be some addition in the in between will there will be sudden change in between that can be taken care of by a separate module. Similarly cat there is a placeholder an s I can find out what should be the actual surface from given this intermediate form.

(Refer Slide Time: 17:26)



So here is an example. So I have in my vocabulary words like fox, cat, goose and I am generating the singular and plural form. So what are you seeing here? From fox to generate the plural form same with cat I add a placeholder as and the end of the character. So that is here now. So this is my intermediate form so; that means, at this point I am not distinguishing between fox and cat they are behaving very similarly. So I need an additional process to find out given this intermediate form what should be the surface level form.

(Refer Slide Time: 18:23)

Spelling Handling

A spelling change rule would insert an e only in the appropriate environment.

<i>Lexical</i>	f	o	x	+N	+Pl		
<i>Intermediate</i>	f	o	x	^	s	#	
<i>Surface</i>	f	o	x	e	s		

Pawan Goyal (IIT Kharagpur) Finite-state methods for morphology Week 3: Lectures

So this is 2 level morphology. I go to fox s there is something in between that may be null or something else intermediate label and then from intermediate labeled you go to the surface level.

Now, what do you think the transition depend depends on from intermediate or surface label? So it will depend on what are the ending characters of the stem and the starting director of the affix, now given these, what should be the replacement character.

(Refer Slide Time: 18:56)

Rule Handling

Rule Notation
 $a \rightarrow b/c_d$: "rewrite a as b when it occurs between c and d ."

Pawan Goyal (IIT Kharagpur) Finite-state methods for morphology Week 3: Lectures

So you can have these simple rules in this notation. This is also called Catherine Cook notation for condensed context as sensitive rules. So that so rules are that a letter a is converted to b if a is visited by c and followed by d . So this is if you remember the context sensitive grammar.

(Refer Slide Time: 19:24)

$a \rightarrow b/c_d$

$\underline{c} \ a \ \underline{d} \rightarrow \underline{c} \ b \ \underline{d}$

Inter. \rightarrow Surface

So that is what I am saying a is converted to b if preceded by c and followed by d ; that means, whenever a comes between c and d , then you convert a to b . This is the context

sensitive rules. So this kind of spelling change rules you can apply to go from intermediate label to the surface level you can use this rules to do the conversion.

(Refer Slide Time: 20:00)

Morphological Analysis: Approaches

Two different ways to address phonological/graphemic variations

- Linguistic approach: A phonological component accompanying the simple concatenative process of attaching an ending
- Engineering approach: Phonological changes and irregularities are factored into endings and a higher number of paradigms

Pawan Goyal (IIT Kharagpur) Finite-state methods for morphology Week 3: Lecture 3 16 / 18

Now so after talking about this 2 level morphology, in general when people design the morphological analysis for a given language, they follow 2 different kinds of approaches. So they are also called the linguistic approach and engineering approach.

So what is the difference between the two? So in the linguistic approach what will happen? You have the stem, stem is already defined in the lexicon or the language and you have an affix that is also defined. So to the stem you will apply the suffixes h, but the surface changing rules will be taking care of separately by the rules of this kind, that we have just seen. So there will be rules of this kind that will define the surface how, what are the changes that are happening at the surface level.

On the other hand, engineering approaches, you would not have the separate rules for spelling changes. They will try to find out the minimum possible unit of the stem that can be used and to that what other and the affixes can be big and in that case that can be applied to the stem. Let us see one example that makes it clear. So the idea is that engineering approach all the phonetic irregularities will be factored into the endings.

(Refer Slide Time: 21:24)

Different Approaches: Example from Czech						
	woman	owl	draft	iceberg	vapor	fly
S1	žen-a	sov-a	skic-a	kr-a	pár-a	mouch-a
S2	žen-y	sov-y	skic-i	kr-y	pár-y	mouch-y
S3	žen-ě	sov-ě	skic-e	kr-ě	pár-ě	mouch-ě
P2	žen-0	sov-0	skic-0	ker-0	par-0	mouch-0

A linguistic approach

$$\begin{matrix} \text{žen} + \begin{Bmatrix} a \\ y \\ \text{ě} \\ 0 \end{Bmatrix} & \text{sov} + \begin{Bmatrix} a \\ y \\ \text{ě} \\ 0 \end{Bmatrix} & \text{skic} + \begin{Bmatrix} a \\ y \\ \text{ě} \\ 0 \end{Bmatrix} & \text{kr} + \begin{Bmatrix} a \\ y \\ \text{ě} \\ 0 \end{Bmatrix} & \text{pár} + \begin{Bmatrix} a \\ y \\ \text{ě} \\ 0 \end{Bmatrix} & \text{mouch} + \begin{Bmatrix} a \\ y \\ \text{ě} \\ 0 \end{Bmatrix} \end{matrix}$$

An engineering approach

$$\begin{matrix} \text{žen} + \begin{Bmatrix} a \\ y \\ \text{ě} \\ 0 \end{Bmatrix} & \text{sov} + \begin{Bmatrix} a \\ y \\ \text{ě} \\ 0 \end{Bmatrix} & \text{skic} + \begin{Bmatrix} a \\ y \\ \text{ě} \\ 0 \end{Bmatrix} & \text{k} + \begin{Bmatrix} ra \\ ry \\ fe \\ er \end{Bmatrix} & \text{p} + \begin{Bmatrix} ára \\ áry \\ áfe \\ ar \end{Bmatrix} & \text{m} + \begin{Bmatrix} oucha \\ ouchy \\ ouše \\ uch \end{Bmatrix} \end{matrix}$$

So here is one example from Czech. So what you are seeing here. So they are various words women, owl, draft, iceberg, wrapper and fly. For each of these you have the actual word and suffixes that is like for women, žen z e n and owl it is sov. So here the root word and there are various affixes are applied in various grammatical functions that are abstracted using S1, S2, S3 so on and P2, the various grammatical functions. You might treat them as singular plural and so on.

So to a given root word where affixes have been applied. So what you are seeing here are the final forms. So for the initial 2 words they seem pretty regular. So you have the word sov you apply a y e n o for getting different forms similarly, for žen, skic you see some difference right they are in red. So what is the difference that you are seeing? So now, the suffix why has been changed to i in the case of S2. Similarly, the accent is gone over e from me and you are getting the single simple e. So that is why it is in red. So this is an irregularity. This is okay, but what happens if we go to ice work with kr. You see in the case of S3 even the stem that is a change. So on r you are getting an accent. And for P2 you are getting another word another character e in between kr. So you are getting k e r. So now so on you will see such changes even vapor and fly. So now, the question is how does one capture all this - all the changes, all these irregularities in the case of stems and the suffixes.

So now in the linguistic approach what will happen, I will take the same root word. So for example, here I will have skic as the root word and kr as the root word. And the same set of suffixes. So y a y e with an accent you know are my suffixes. That is my root and the suffix pair I will add these and whatever a spelling change rules I will try to enumerate, whatever different spelling change rules, by taking what is the previous character previous to previous character, I will try to define the spelling changes in that way. That is a linguistic approach.

What will happen in engineering approach? I try to find out the part of the stem that is common across all these variations. So it is with the first 3, zen, sov and skic, but with kr, I will take only k as common, I will not take r because r is getting changed into something with an accent or with an e n r. So I will take only k as common is part of my stem, and everything else remaining I will convert into my stem that I might add. And that you see in the last three. So you have only k p n m as the actual strength and everything else goes into the suffix. So this is what is engineering approach. So by doing that you do not have to worry about handling the spelling change rules separately you are you handling them here itself.

(Refer Slide Time: 24:55)

Tools Available

- AT&T FSM Library and Lextools
<http://www2.research.att.com/~fsmtools/fsm/>
- OpenFST (Google and NYU)
<http://www.openfst.org/>

Pawan Goyal (IIT Kharagpur) Finite-state methods for morphology Week 3: Lecture 3

So there are various tool kits set available for doing the morphological pricing, for example, AT and FSM library that is very popular. Again the OPENFST tool is very popular for doing the morphological analysis for a given language. So this is this was

about our computation morphology using finite state methods. So I have very briefly talked about what our financial methods and how do we how do you use that for computational morphology.

So in the next lecture we will talk about in the same process a more popular problem about part of speech tagging. So what is the problem part of speech tagging and what are the different computational models that you can use to handle that.

Thank you.