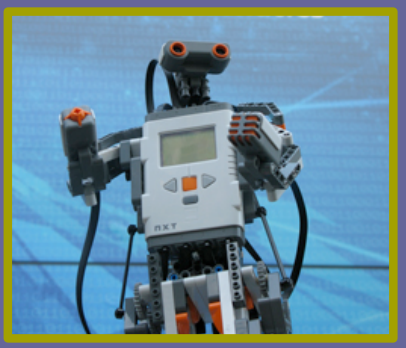
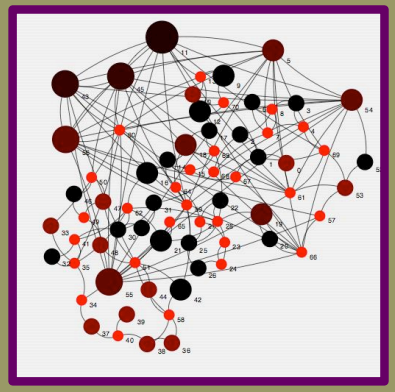
COMP40020 Human Language Technologies

Cats, Flys & Tomatos

March 2019





Prof. Julie Berndsen
School of Computer Science
Julie.Berndsen@ucd.ie

HLT13

HLT10

Contents:

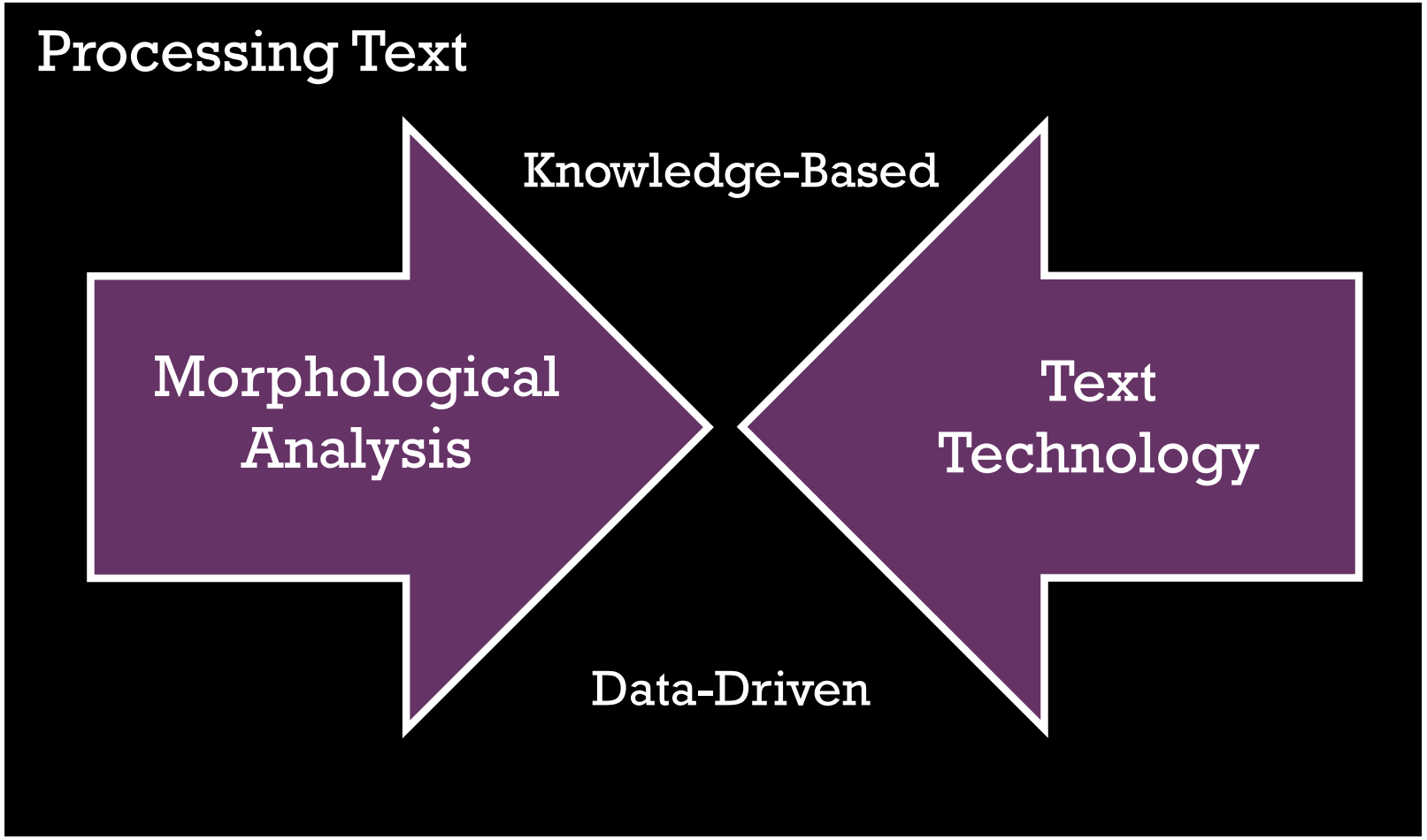
- Stemming and Lemmatisation
- Context for Computational Morphology
- Finite-State Morphology
- Spelling Errors

Aim:

- To give an overview of computational models of morphology which underlie knowledge-based and data-driven processing of text.

UCD DUBLIN

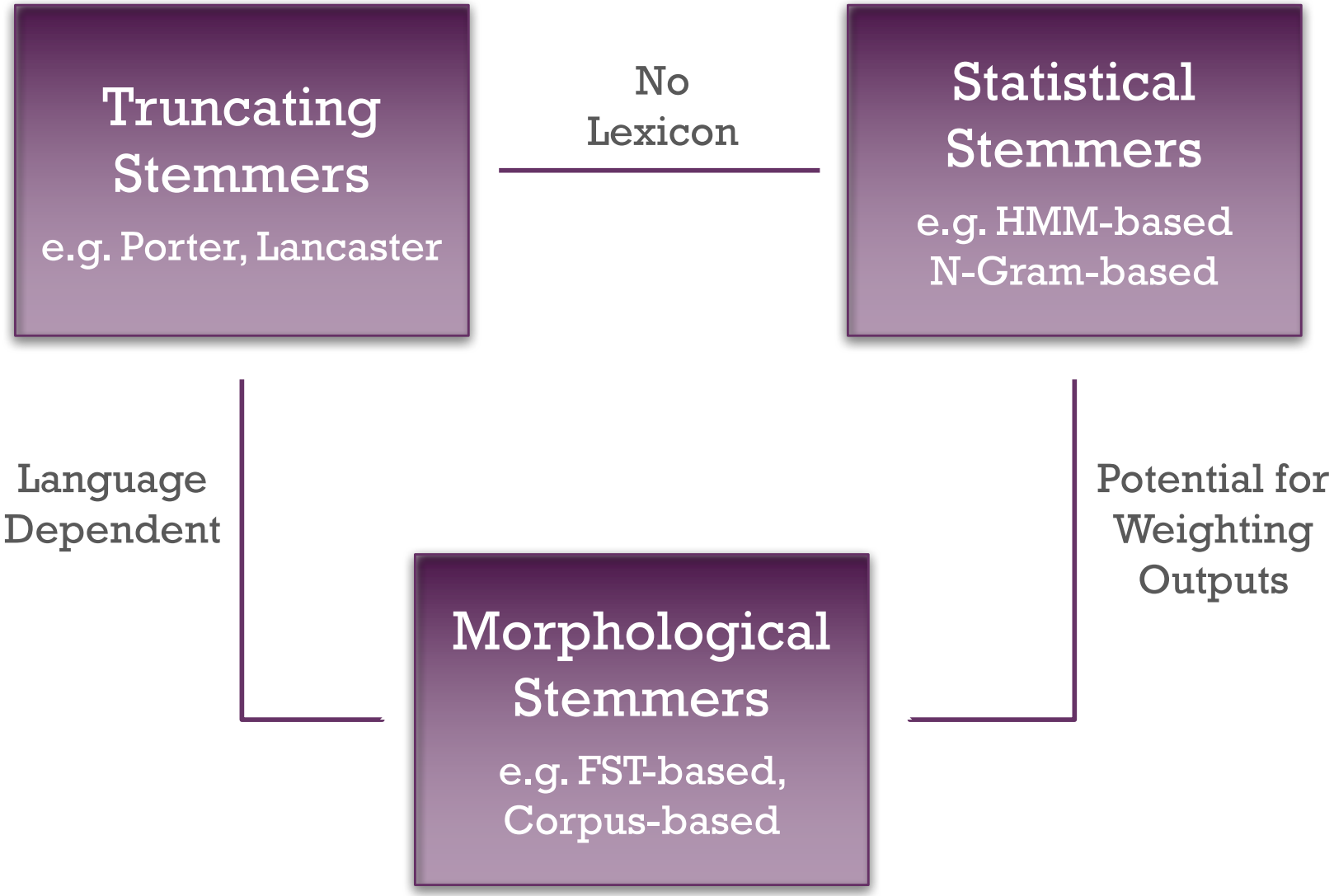
Computational Morphology



Stemming… without a Lexicon

- E.g. Porter Stemmer
  - Works by truncating the orthographic form of a word → language dependent
  - [C](VC)^m[V] – where m is the measure of the word (or word part) and [] denotes arbitrary presence of the contents
  - To remove suffixes a cascade of handwritten rules are applied of the form (condition) S1 → S2 which states that S1 is replaced by S2 if the condition applies to the stem which precedes S1
  - An explanation of the algorithm can be found at http://snowball.tartarus.org/algorithms/porter/stemmer.html

Stemming Approaches



Lemmatisation… with a lexicon

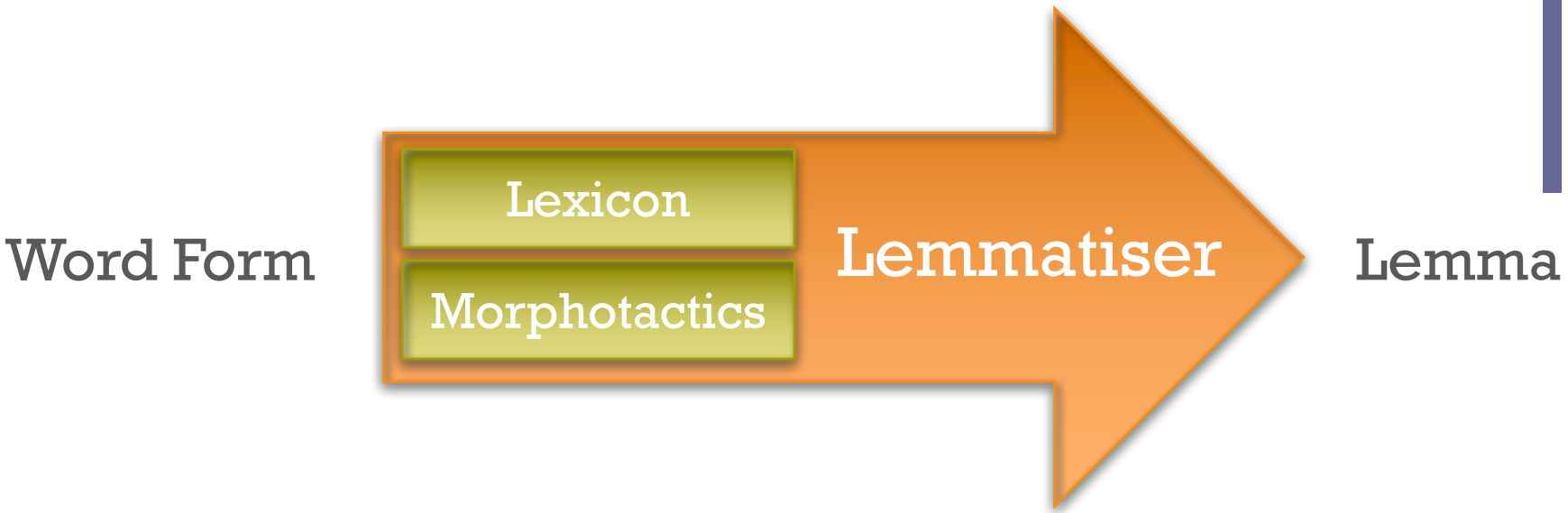
HLT10

Lexeme					Lemma
see	sees	seeing	saw	seen	**see**
sleep	sleeps	sleeping	slept	slept	**sleep**
good	better	best			**good**
woman	women				**woman**
be	is	are	was	were	**be**

- Lemmatisation is the process of determining to which lexeme a word form belongs.
- The *lemma* is the form used to refer to the lexeme and is not necessarily a substring of the word form.

UCD DUBLIN

Lemmatisation… with a lexicon



Lexicon	Morphotactics
e.g. WordNet • "a large lexical database of English. Nouns, verbs, adjectives and adverbs are grouped into sets of cognitive synonyms (synsets), each expressing a distinct concept".	e.g. morphy • A set of morphological functions for WordNet which can be used to determine the lemma given the word form and category of the word (i.e. noun, verb, adjectives and adverbs).

See https://wordnet.princeton.edu/

Example:Verbs

Regular

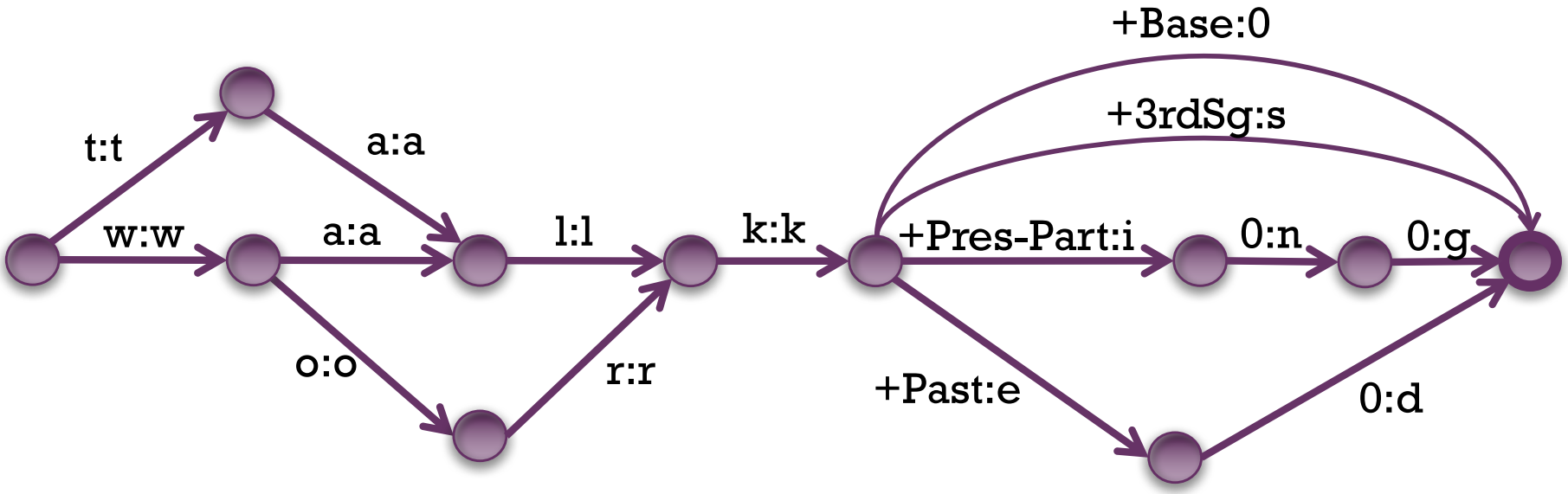
Lemma	Inflected Form			
Root	**3rdSg**	**Past**	**Past-Part**	**Pres-Part**
talk	talks	talked	talked	talking
walk	walks	walked	walked	walking
work	works	worked	worked	working

Irregular

Lemma	Inflected Form			
Root	**3rdSg**	**Past**	**Past-Part**	**Pres-Part**
catch	catches	caught	caught	catching
go	goes	went	gone	going
eat	eats	ate	eaten	eating
be	is	was	been	being

Example: Regular Verbs

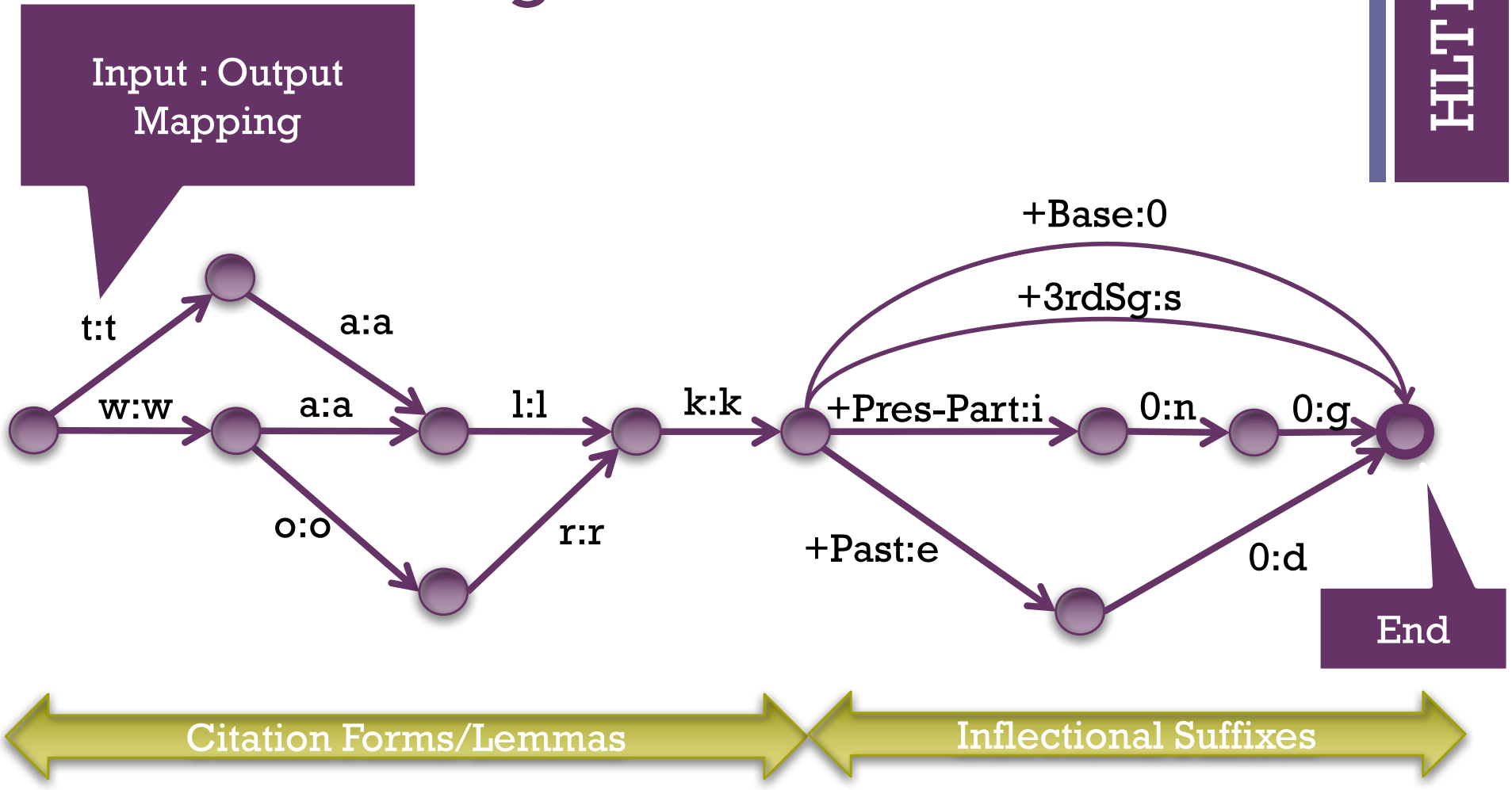
HLT10



Example based on Karttunen (2005) LSA slides

UCD DUBLIN

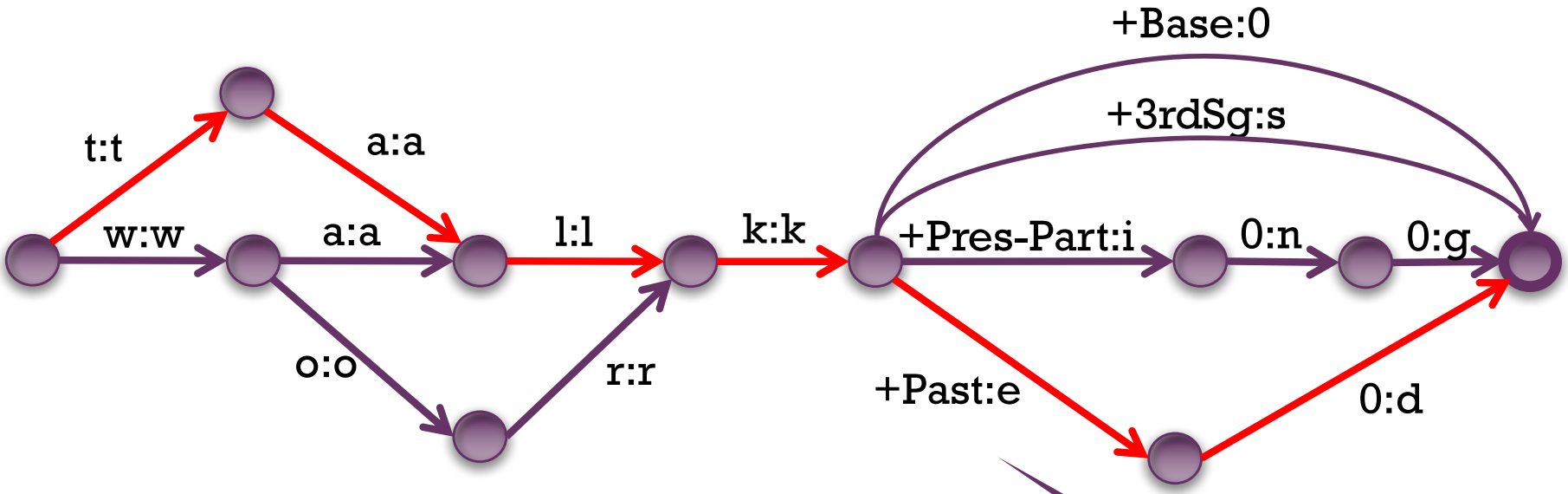
Example: Regular Verbs



Finite State Transducer (FST)

Example based on Karttunen (2005) LSA slides

Example: Regular Verbs



Bidirectional Finite State Transducer

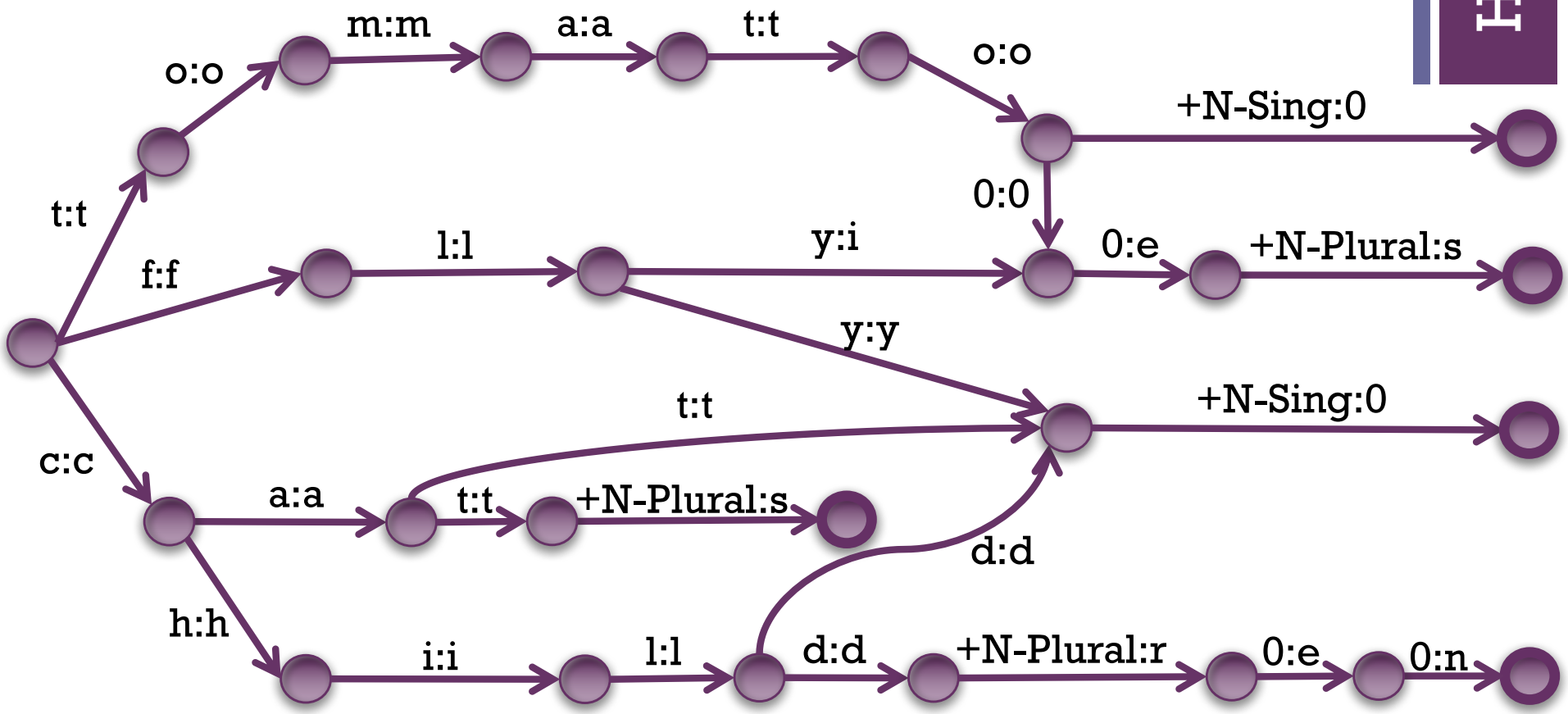
talked → talk+Past

Example based on Karttunen (2005) LSA slides

Example: Plural Nouns

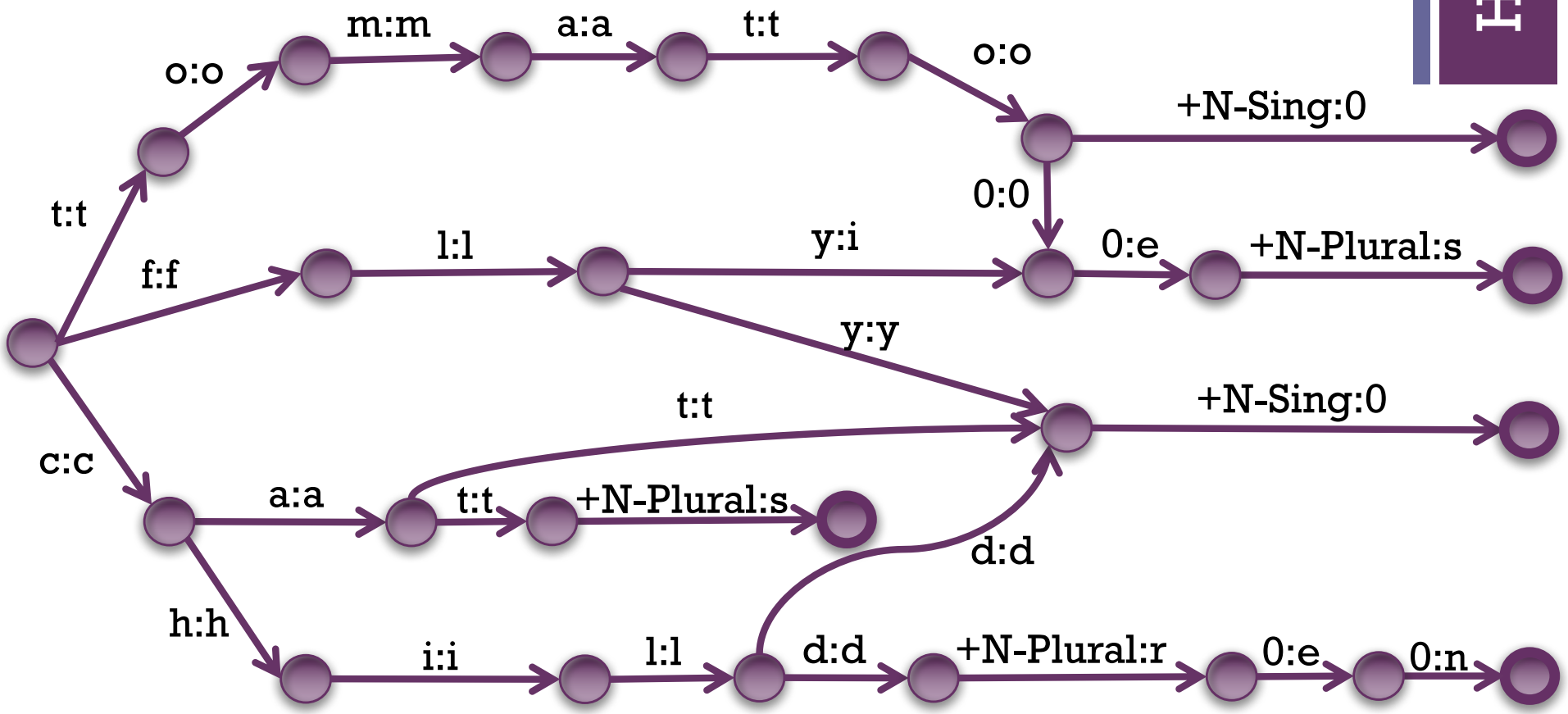
Lemma	Inflected Form
Root	N-Plural
cat	cats
fly	flies
tomato	tomatoes
child	children

Example: Plural Nouns



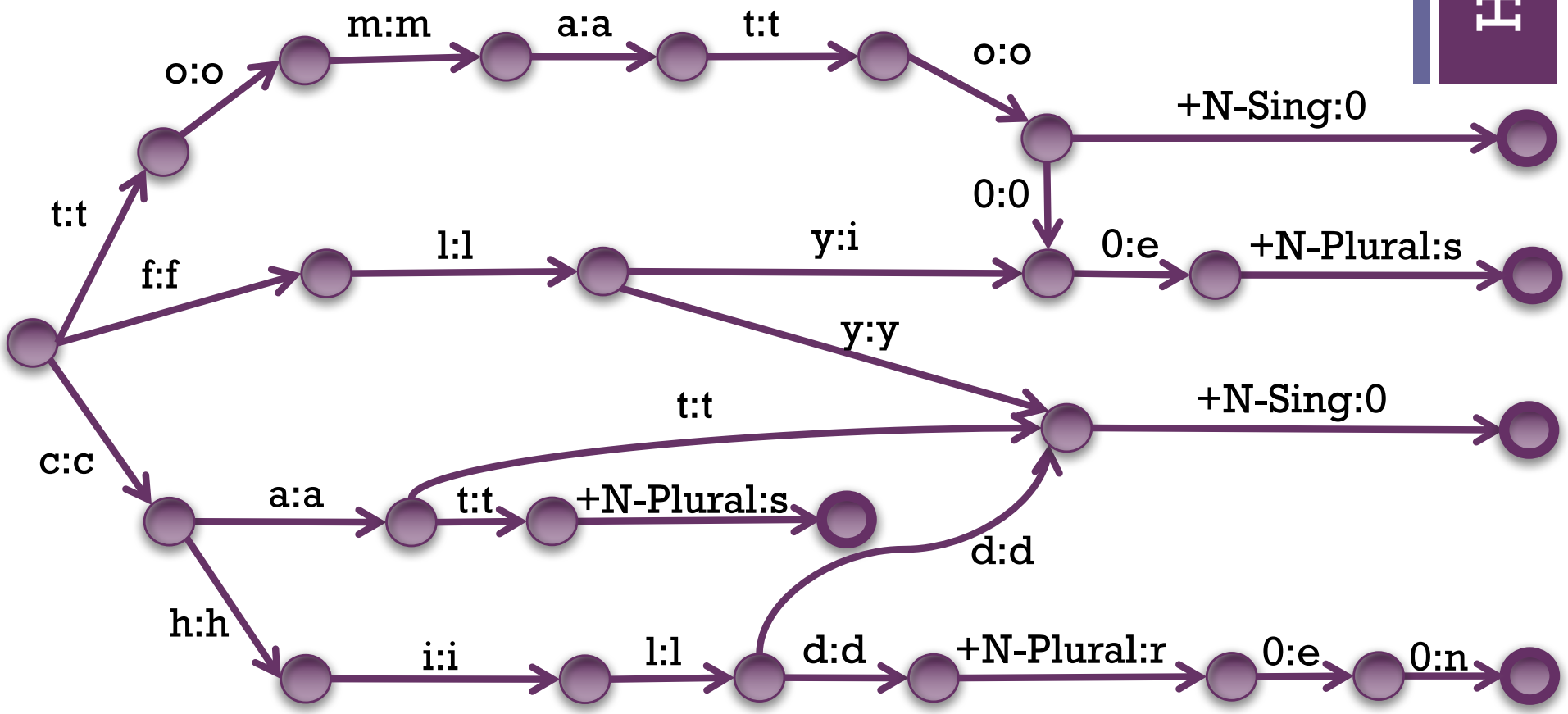
tomato+N-Plural → tomatoes

Example: Plural Nouns



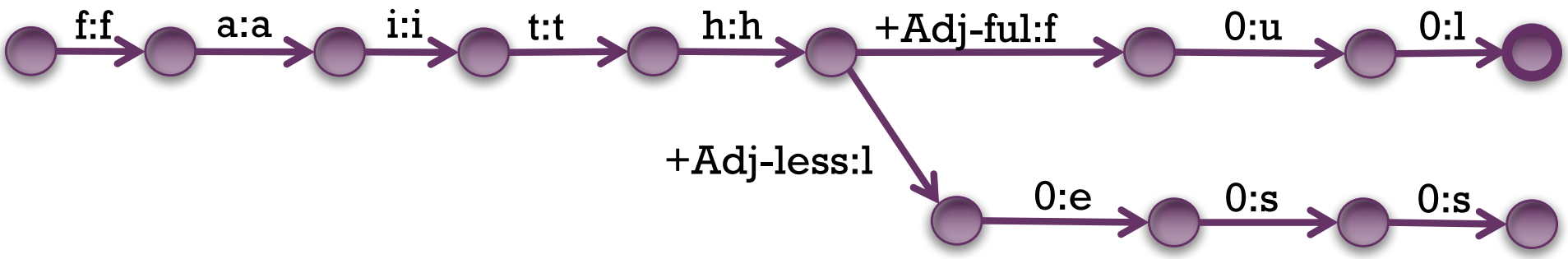
cat → cat+N-Sing

Example: Plural Nouns

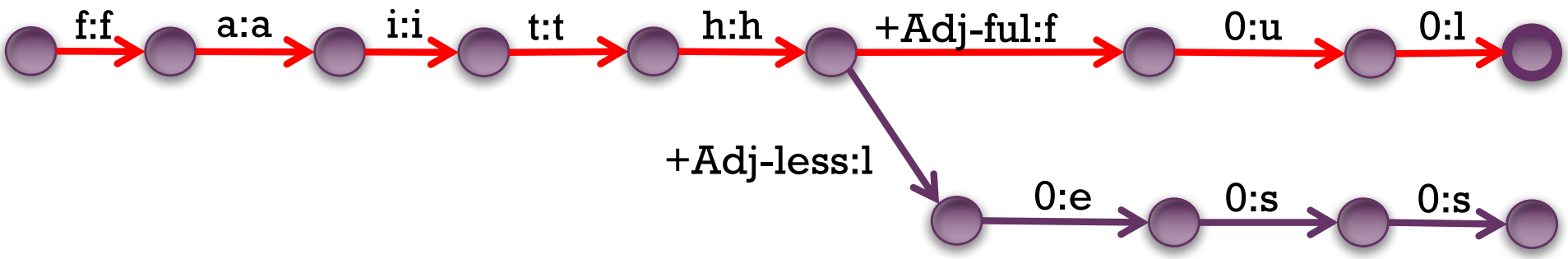


fly+N-Plural → flies

Example: Derivations

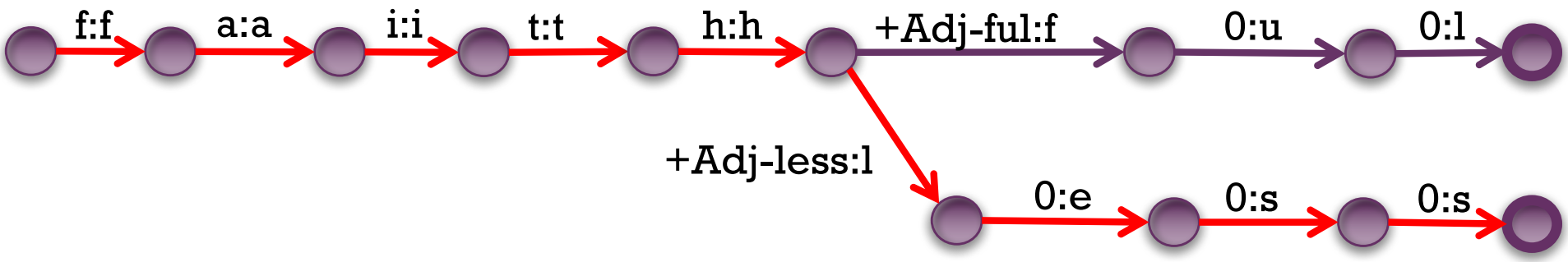


Example: Derivations



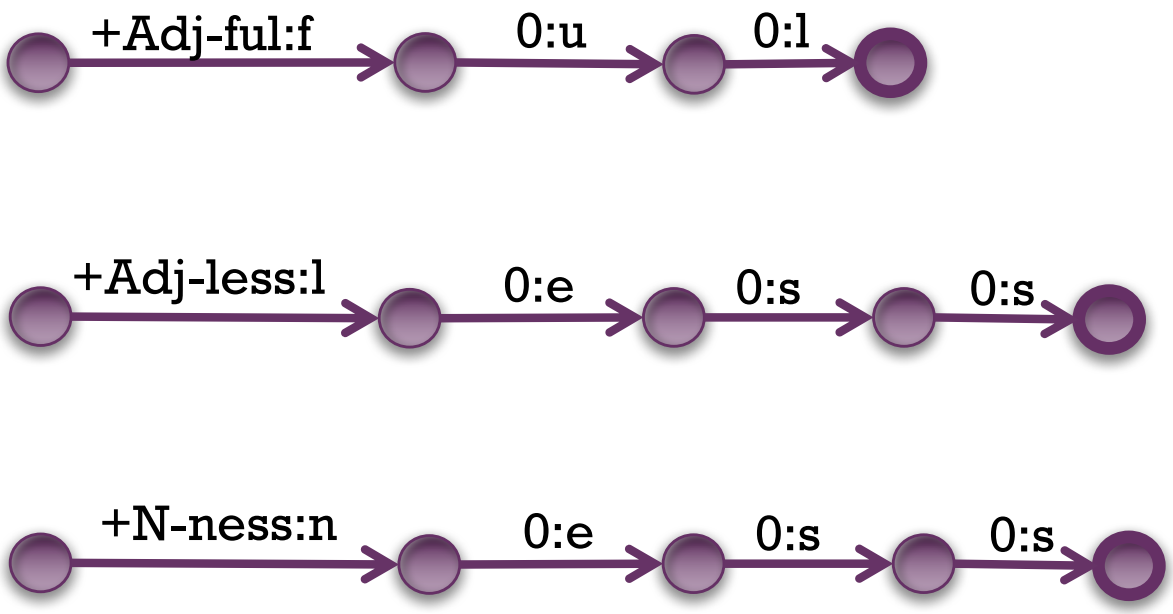
faith+Adj-ful → faithful

Example: Derivations

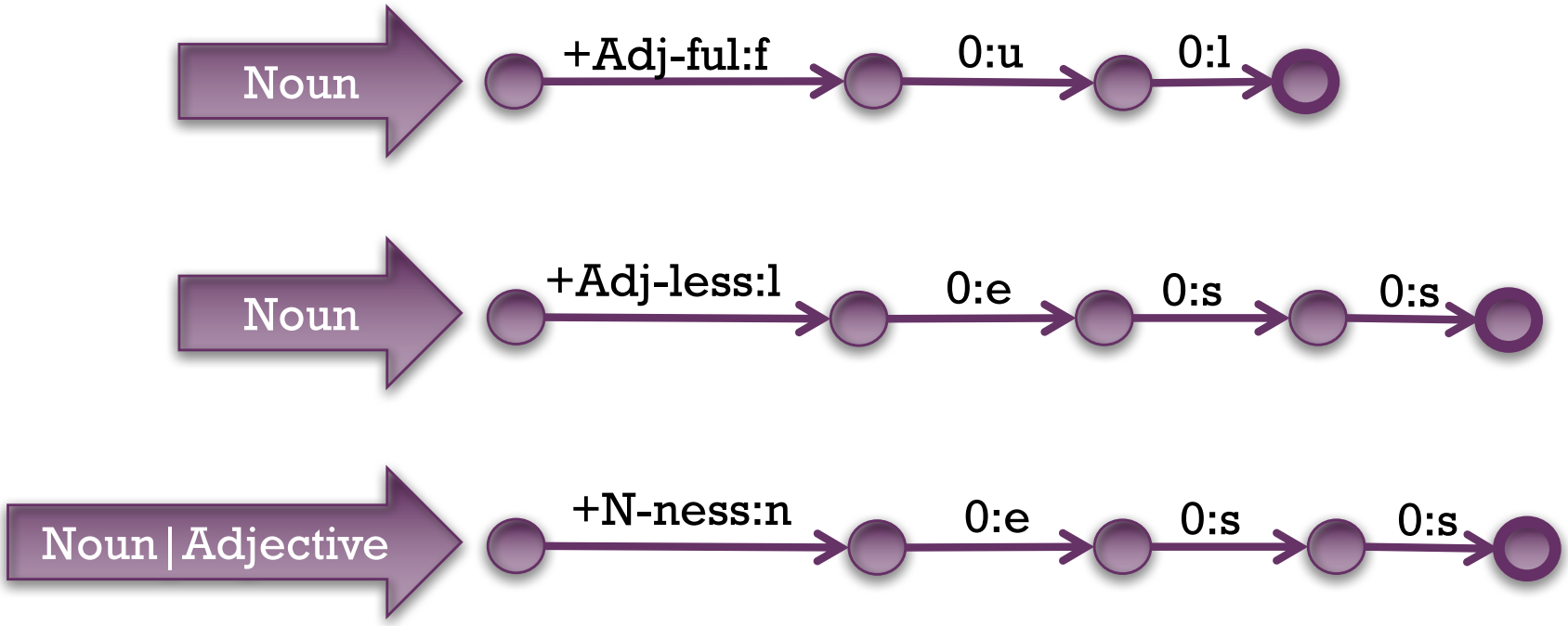


faith+Adj-less → faithless

Example: Morphotactics

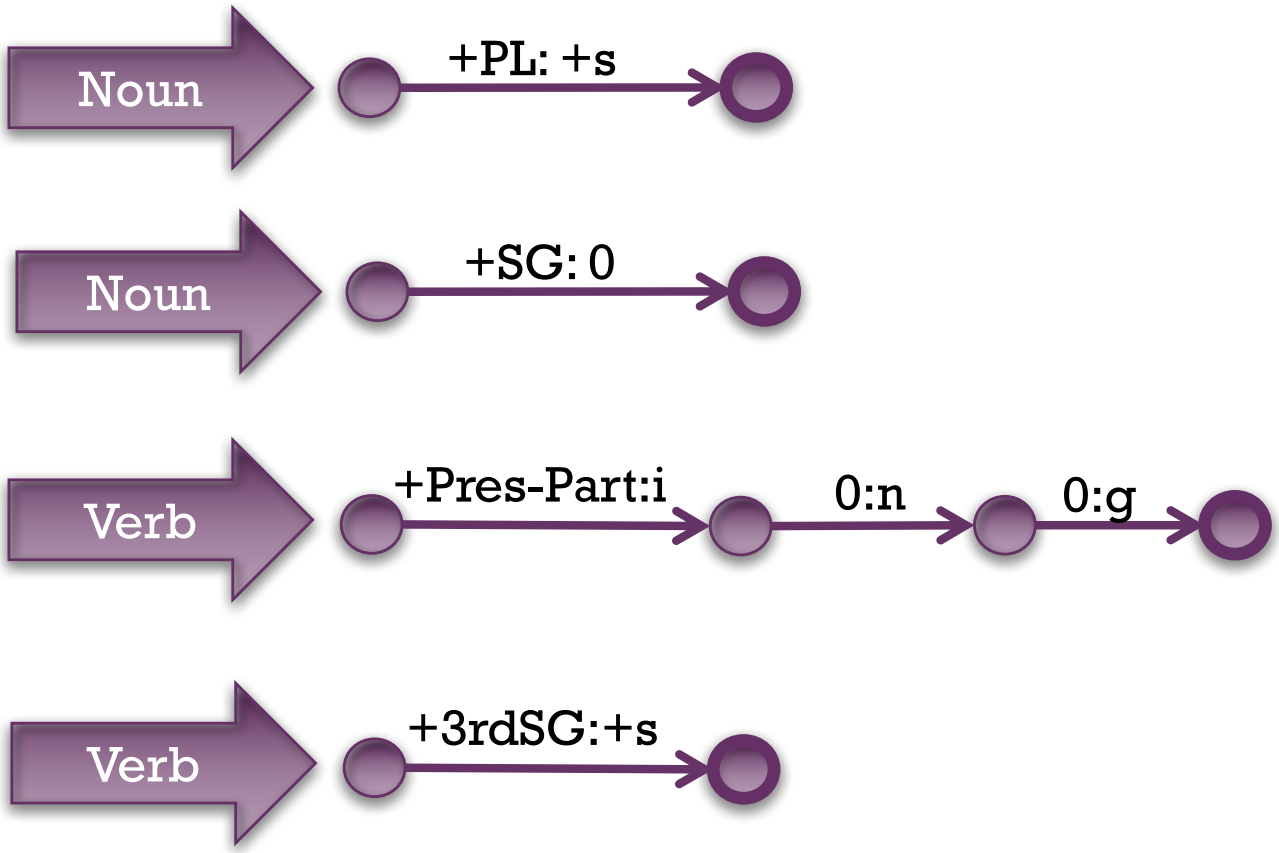


Example: Morphotactics



Defining valid words

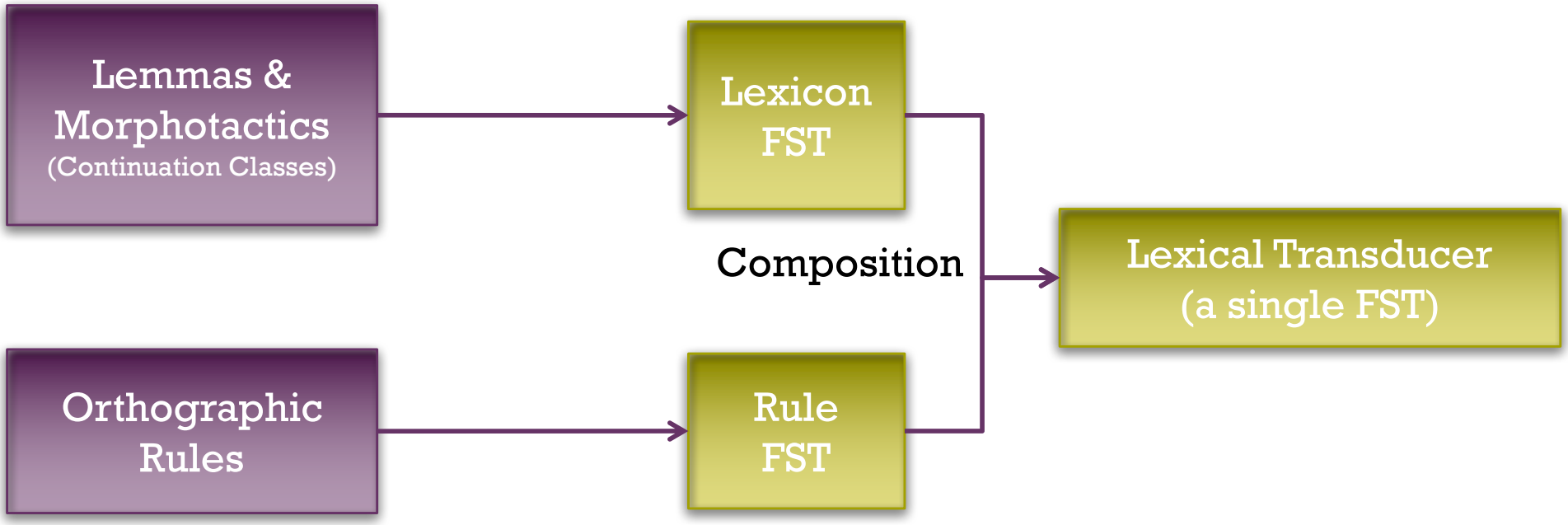
Example: Morphotactics



Defining valid words

Finite State Morphology

HLT10



Two-Level Morphology

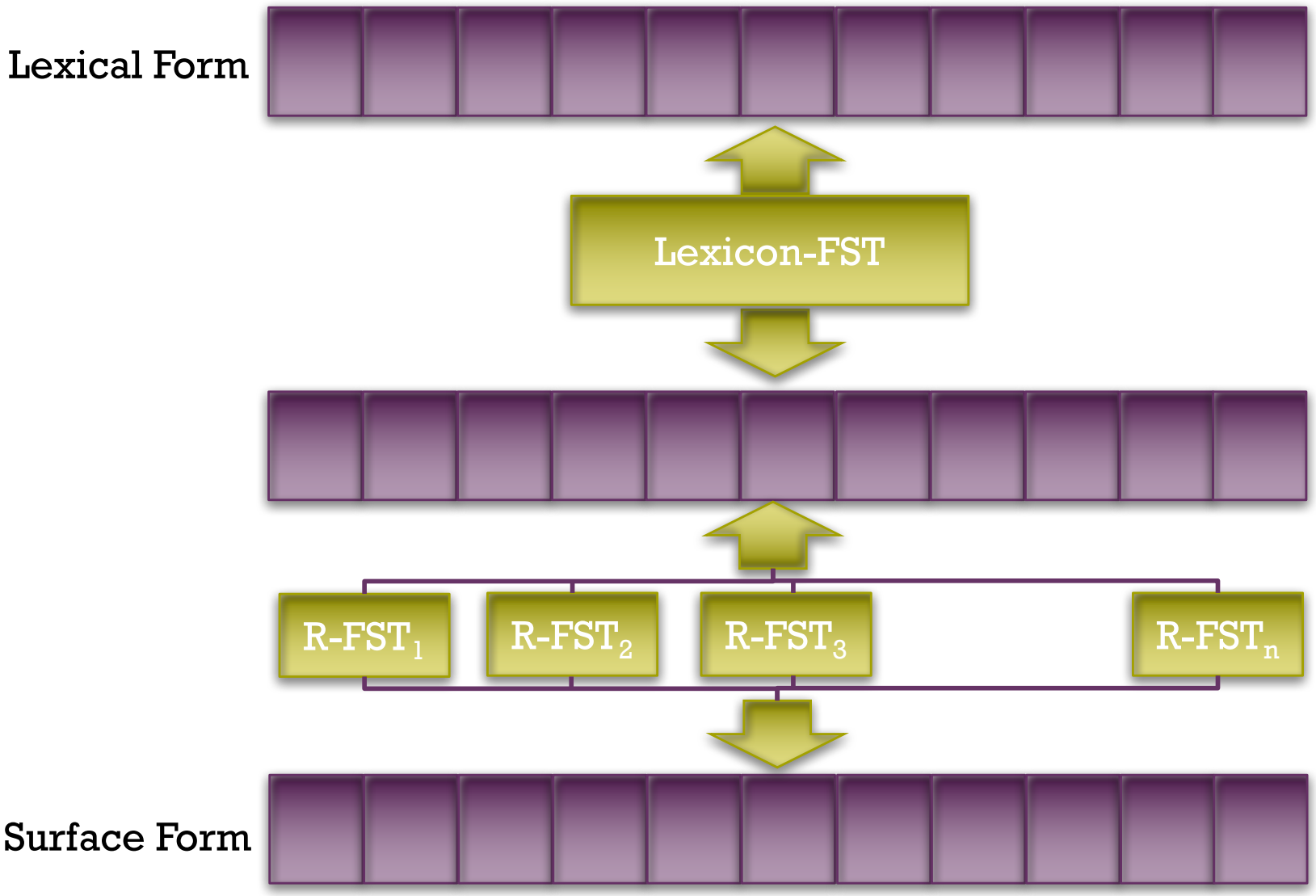


Figure based on Jurafsky & Martin (2008)

Two-Level Morphology

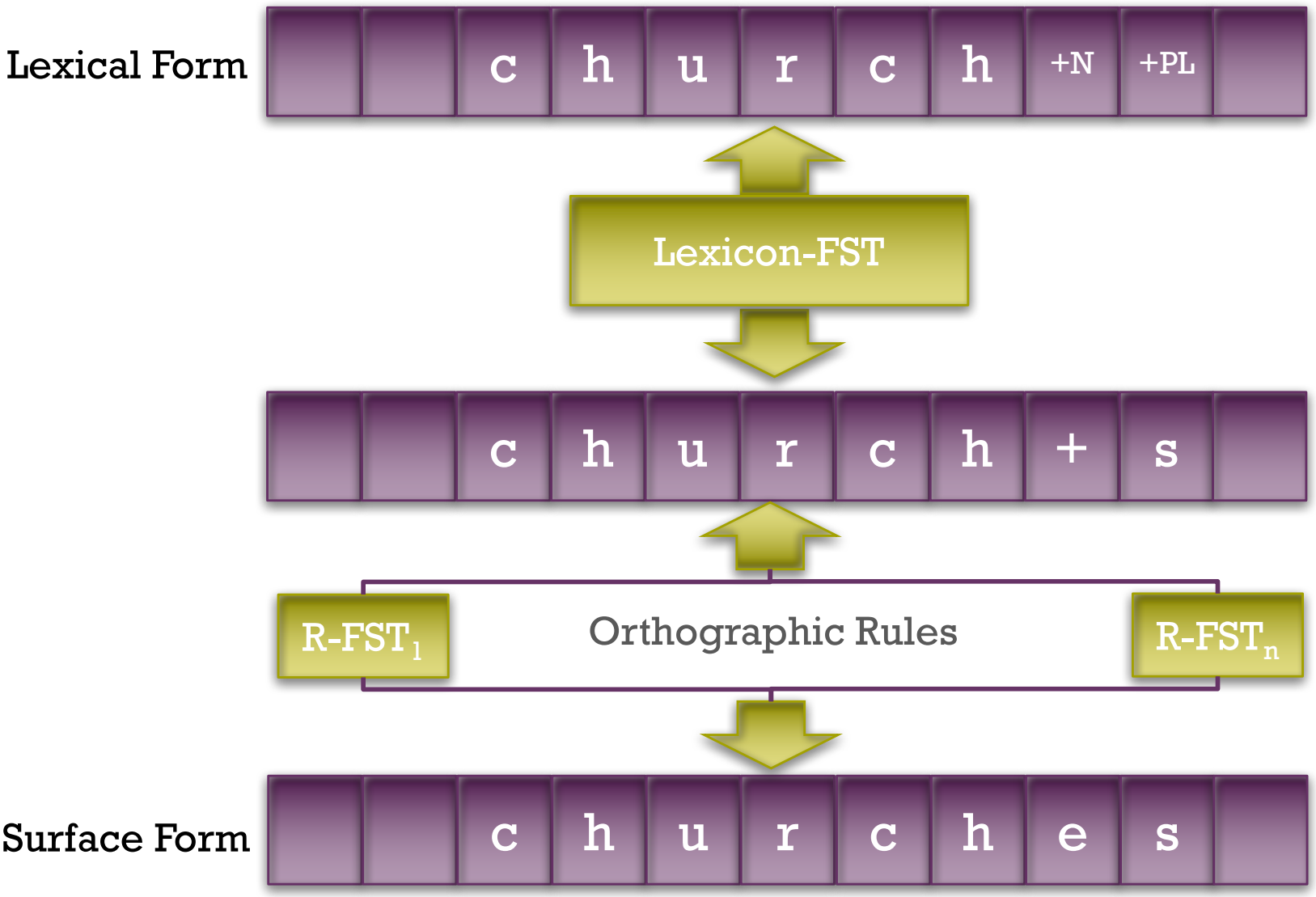


Figure based on Jurafsky & Martin (2008)

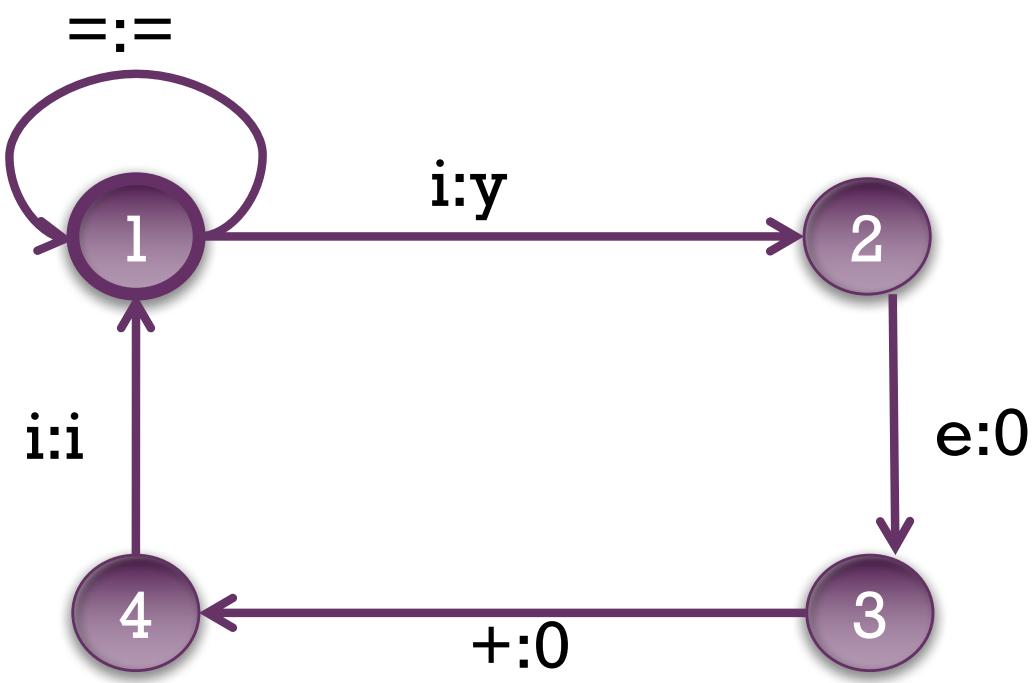
Rule-FST: Alphabets (Σ) and Sets

	Lexical	Surface
Σ	a b c d e f g h i j k l m n o p q r s t u v w x y z – ‘ “ “ +	a b c d e f g h i j k l m n o p q r s t u v w x y z – ‘ “ “
C	b c d f g h j k l m n p q r s t v w x z	b c d f g h j k l m n p q r s t v w x z
V	a e i o u y	a e i o u y
S	s x z	s x z
=	a b c d e f g h i j k l m n o p q r s t u v w x y z – ‘ “ “ 0	a b c d e f g h i j k l m n o p q r s t u v w x y z – ‘ “ “ 0

Note that the “=“ in the above table represents any sound that is not explicit on a transition i.e. “=“ means anything else – this just simplifies the presentation of the rules.

Classification based on Ritchie et al (1992)

Example: i-to-y Rule-FST

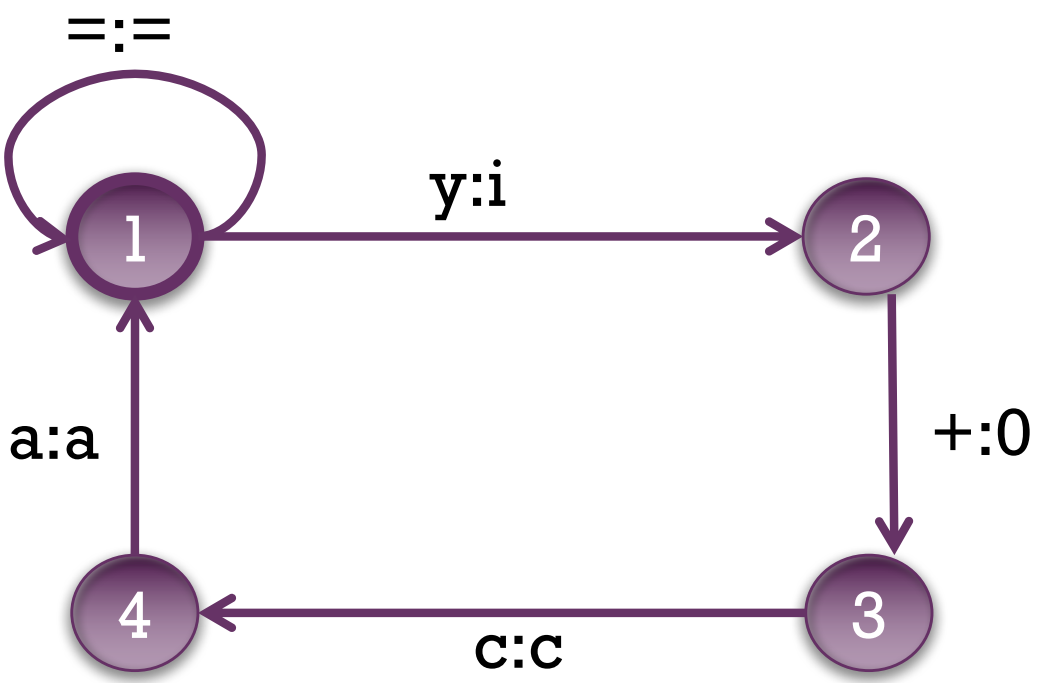


e.g.

tie + ing → tying

lie + ing → lying

Example: y-to-i-before-c Rule-FST

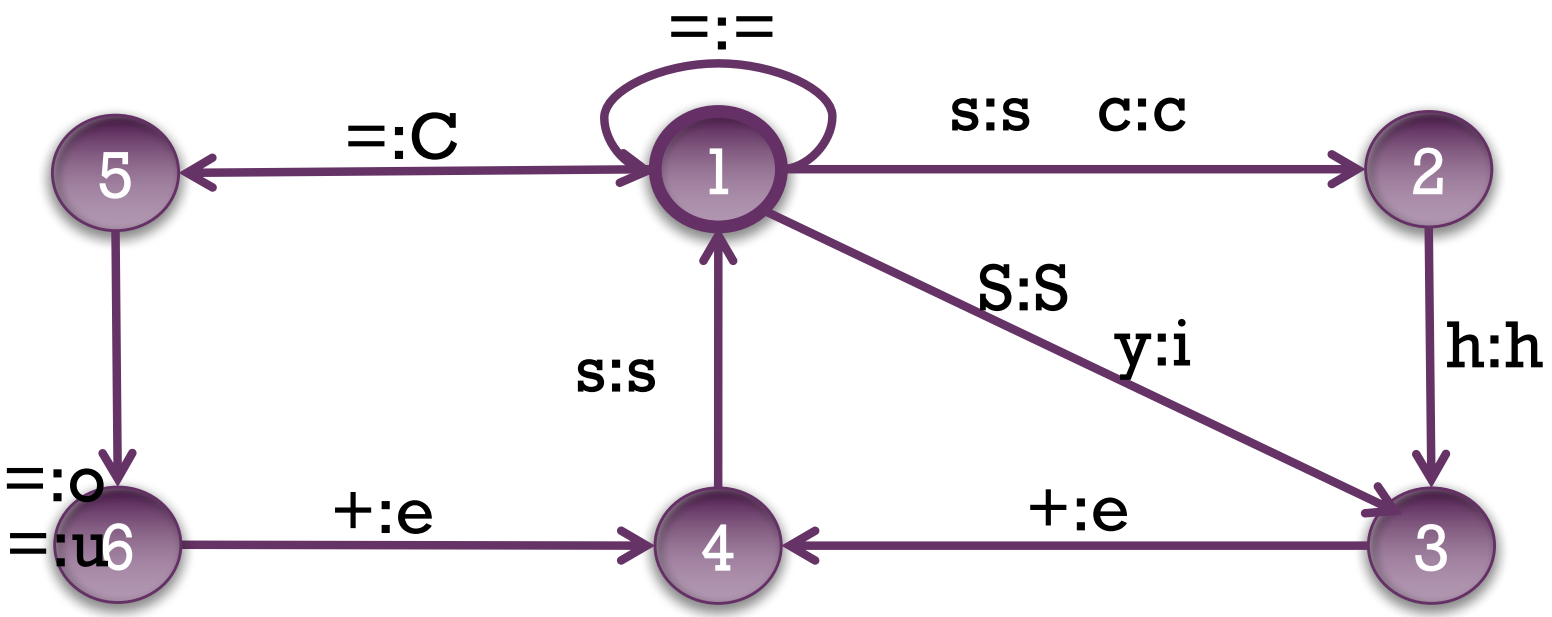


e.g.

apply + cation → application

morphology + cal → morphological

Example: e-Insertion Rule-FST



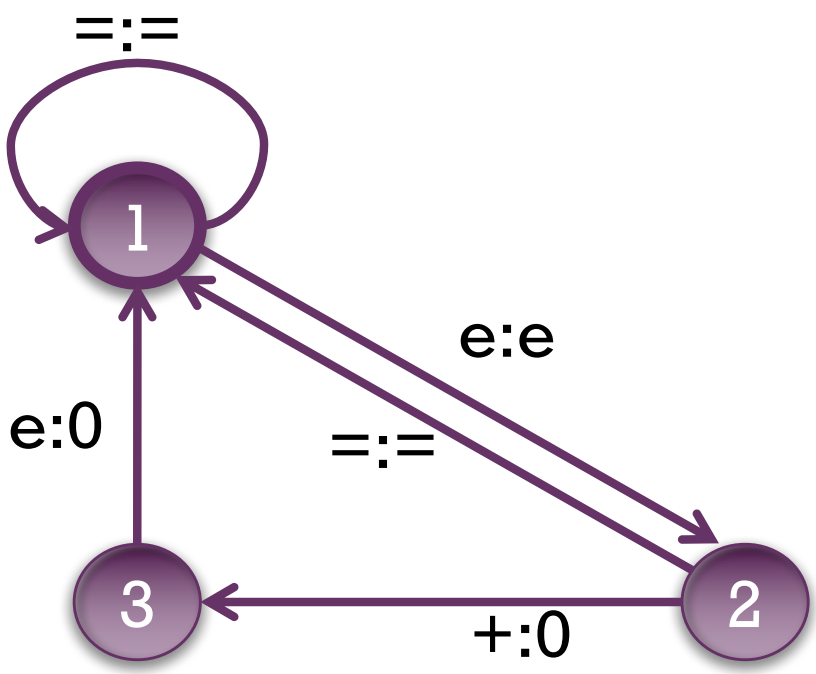
e.g.

tomato +s → tomatoes

church +s → churches

box +s → boxes

Example: e-Deletion Rule-FST



e.g.

move +ed → moved

smoke +ed → smoked

Two-Level Morphology

HLT10

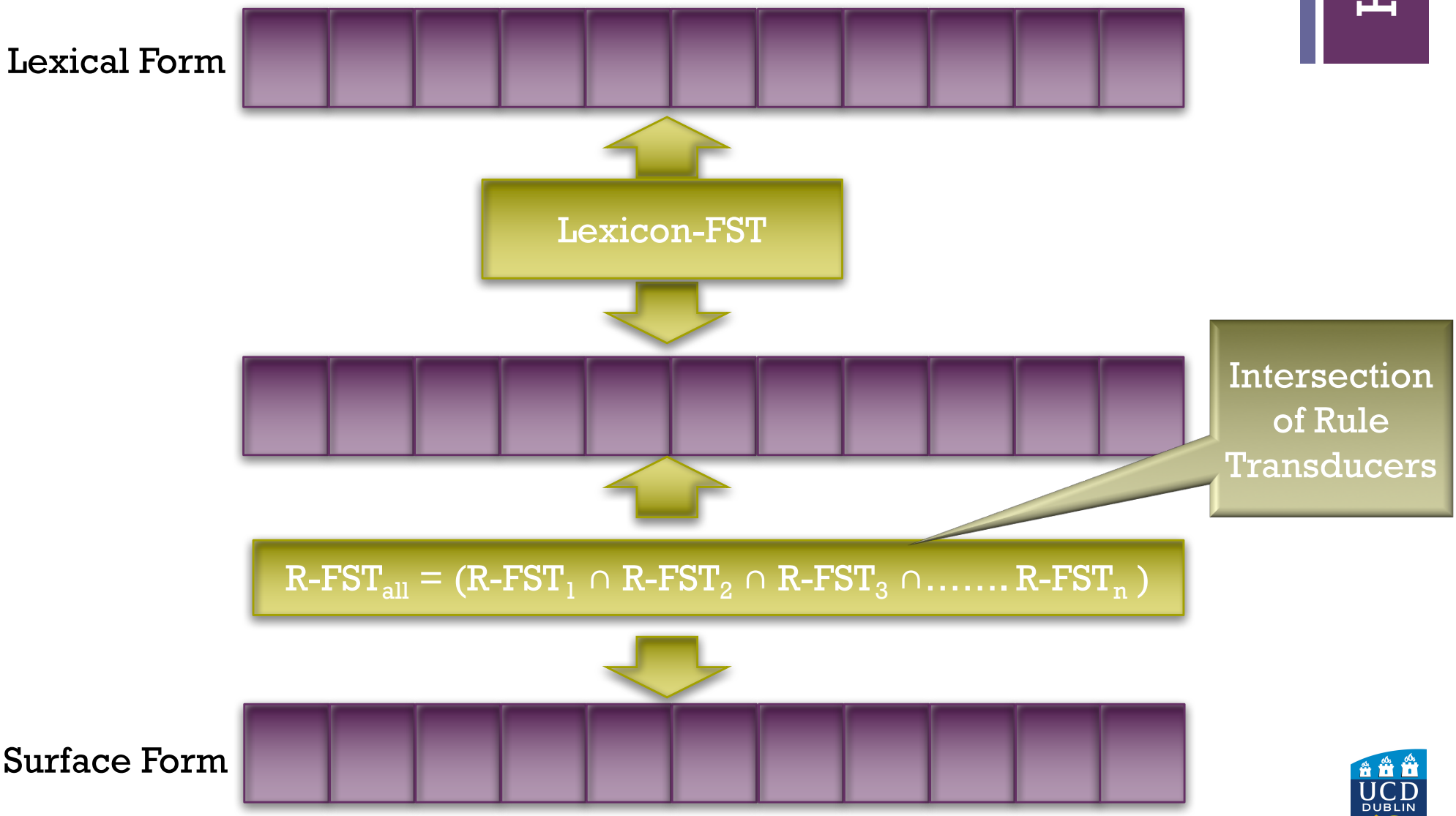


Figure based on Jurafsky & Martin (2008)

UCD DUBLIN

Two-Level Morphology

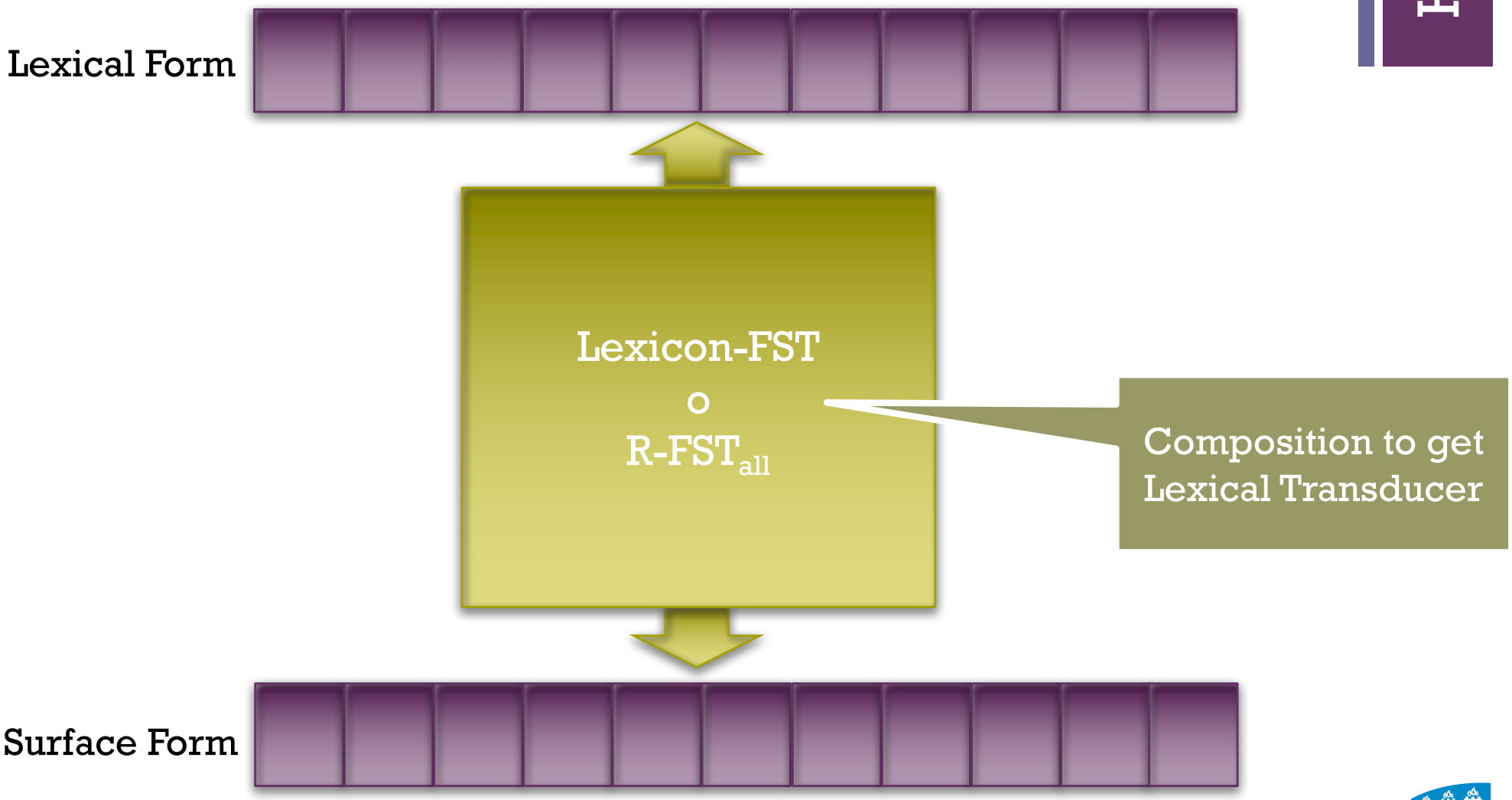
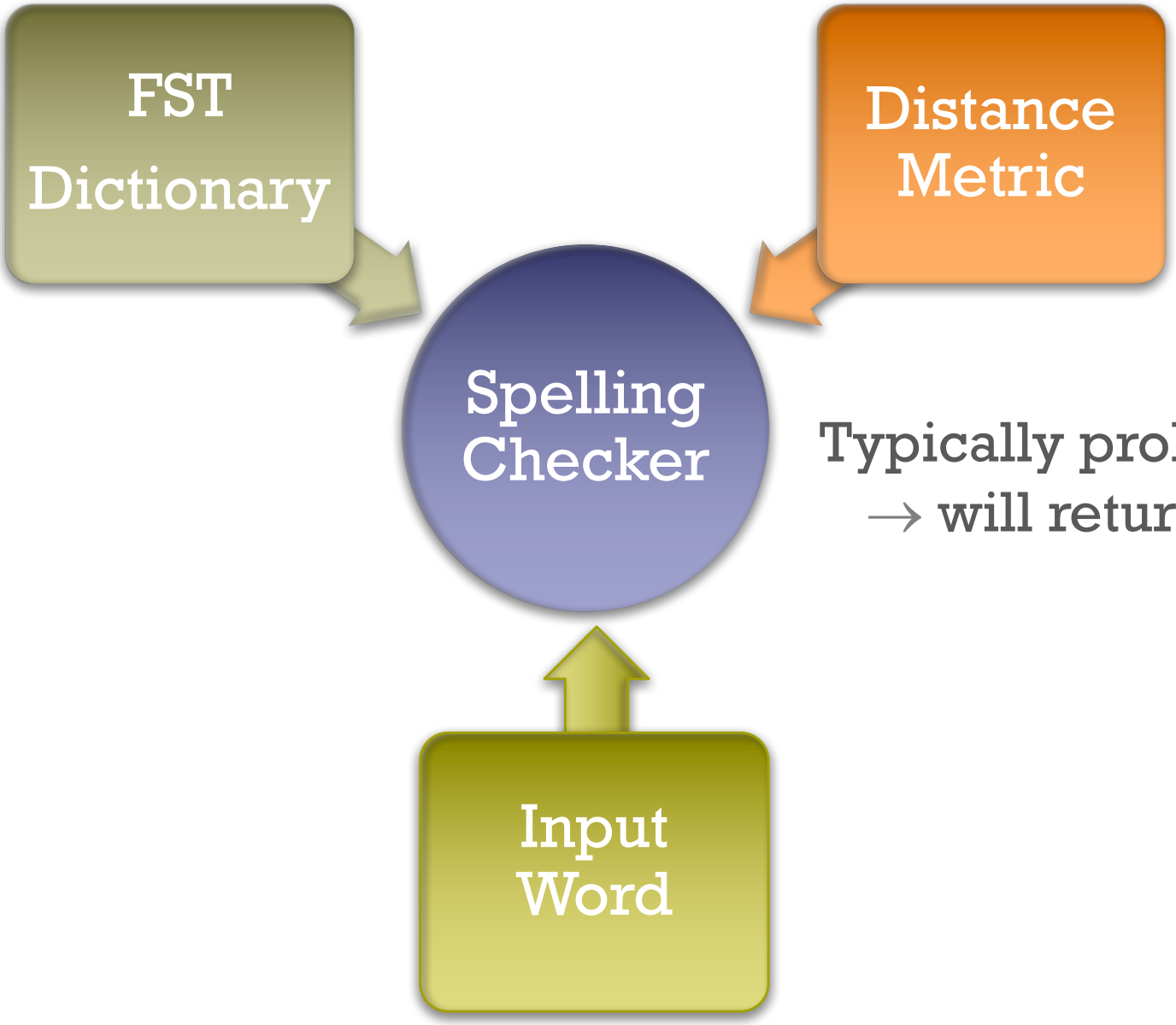


Figure based on Jurafsky & Martin (2008)

Dictionaries and Spelling



Typically probabilistic
→ will return to this

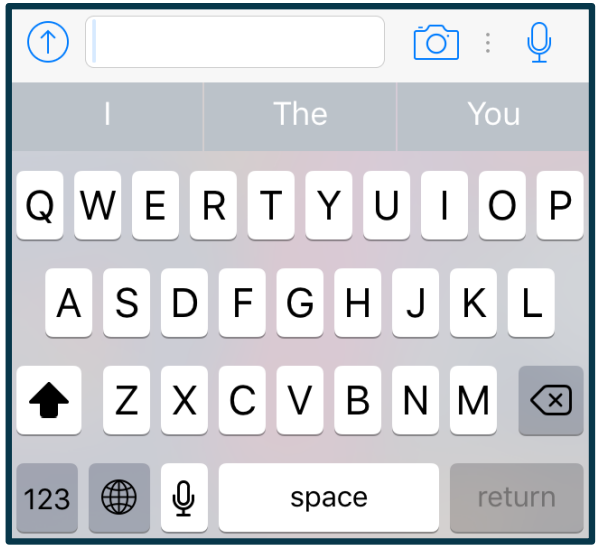
Spelling Errors and Distance

HLT10

Using Edit-Distance to find the minimum distance between two strings based on the number of operations required to transform one to the other

l	a	n	g	u	a	g	e

l	a	n	g	a	u	g	e



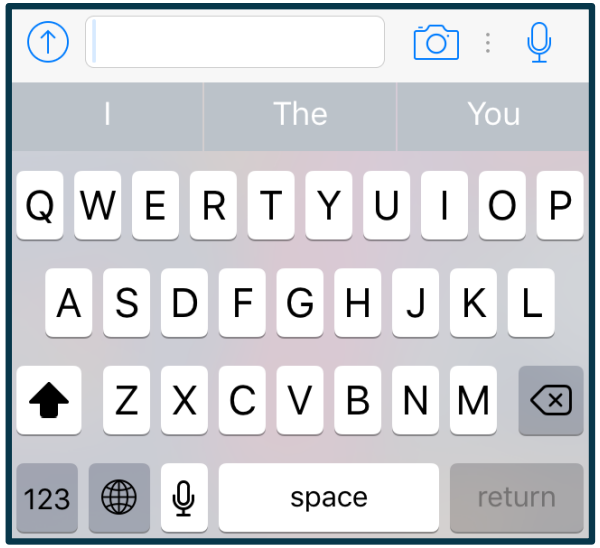
UCD DUBLIN

Spelling Errors and Distance

HLT10

Using Edit-Distance to find the minimum distance between two strings based on the number of operations required to transform one to the other

l	a	n	g	u	a	g	e
M	M	M	M	S	S	M	M
l	a	n	g	a	u	g	e

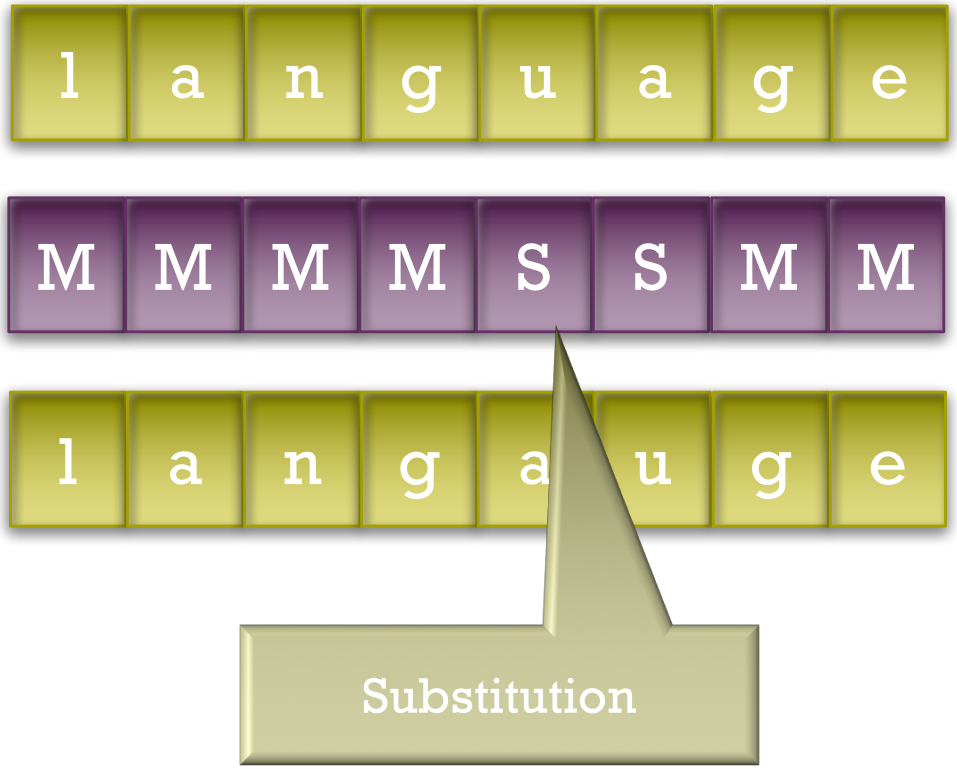


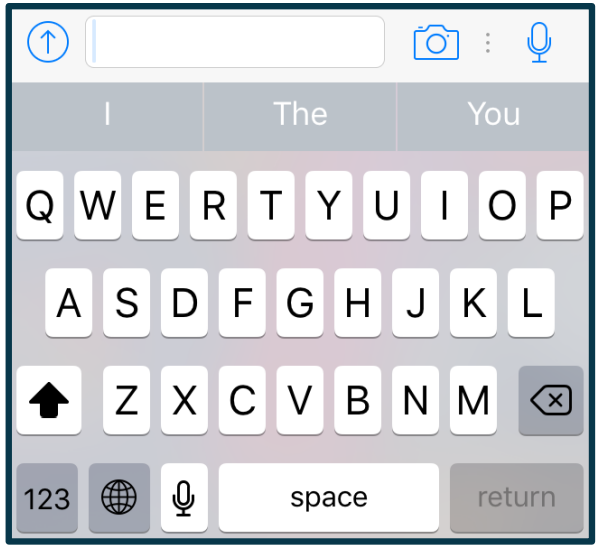
UCD DUBLIN

Spelling Errors and Distance

HLT10

Using Edit-Distance to find the minimum distance between two strings based on the number of operations required to transform one to the other





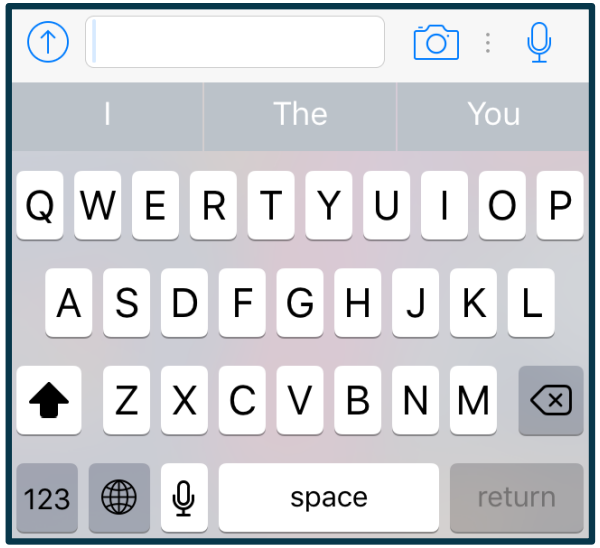
UCD DUBLIN

Spelling Errors and Distance

HLT10

Using Edit-Distance to find the minimum distance between two strings based on the number of operations required to transform one to the other

l	a	n	g	u	a	g	e
S	M	M	M	M	M	M	M
k	a	n	g	u	a	g	e

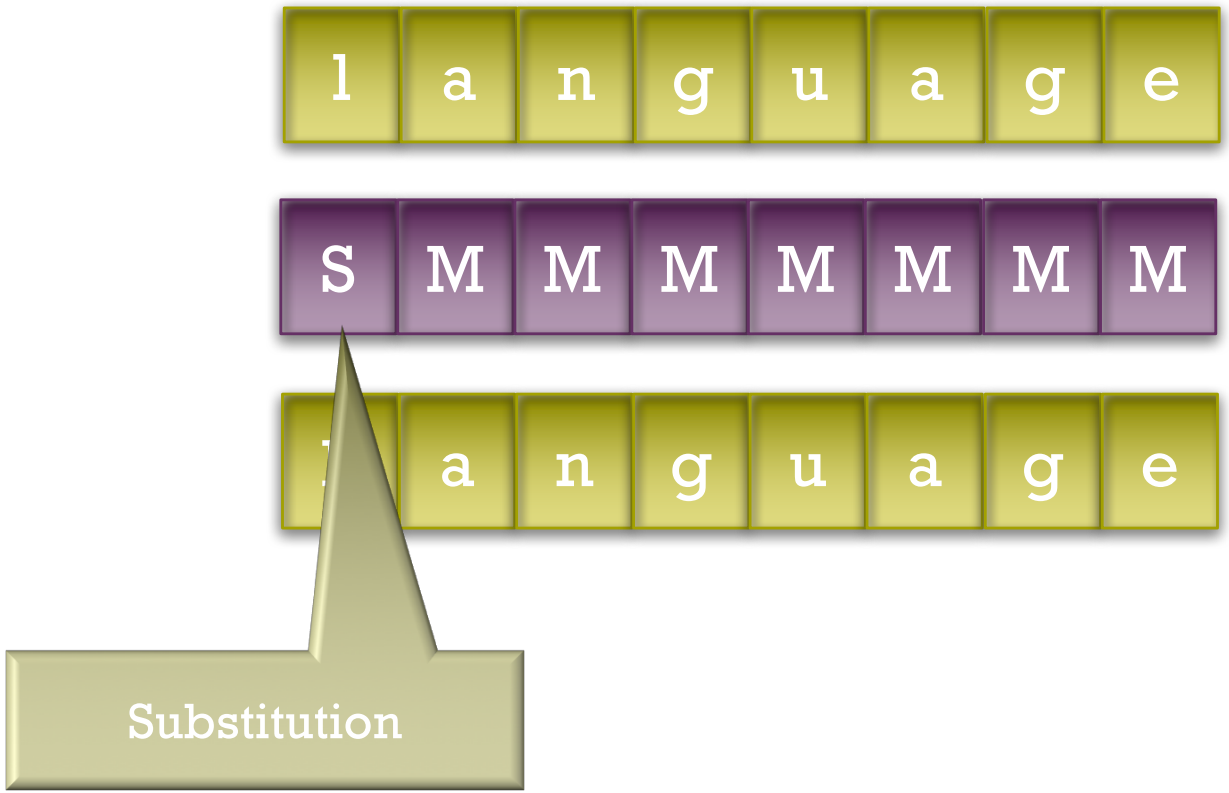


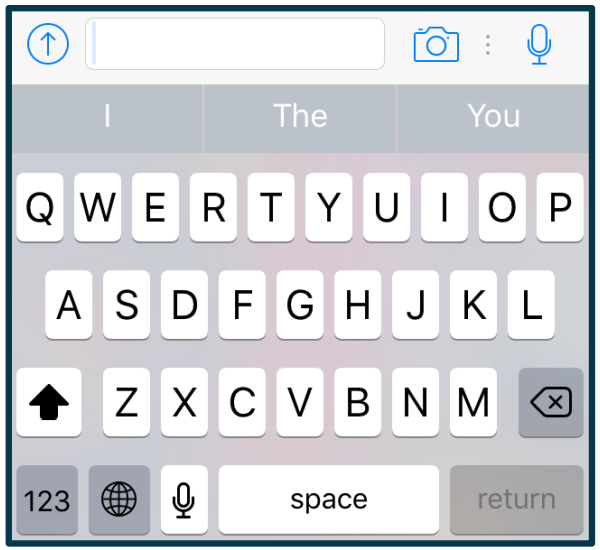
UCD DUBLIN

Spelling Errors and Distance

HLT10

Using Edit-Distance to find the minimum distance between two strings based on the number of operations required to transform one to the other





UCD DUBLIN

Spelling Errors and Distance

HLT10

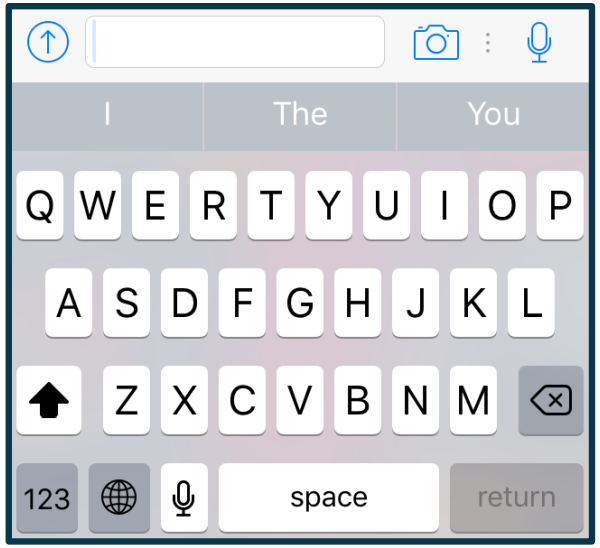
Using Edit-Distance to find the minimum distance between two strings based on the number of operations required to transform one to the other

l	a	n	g	u

a	g	e

M	M	M	M	M	I	M	M	M

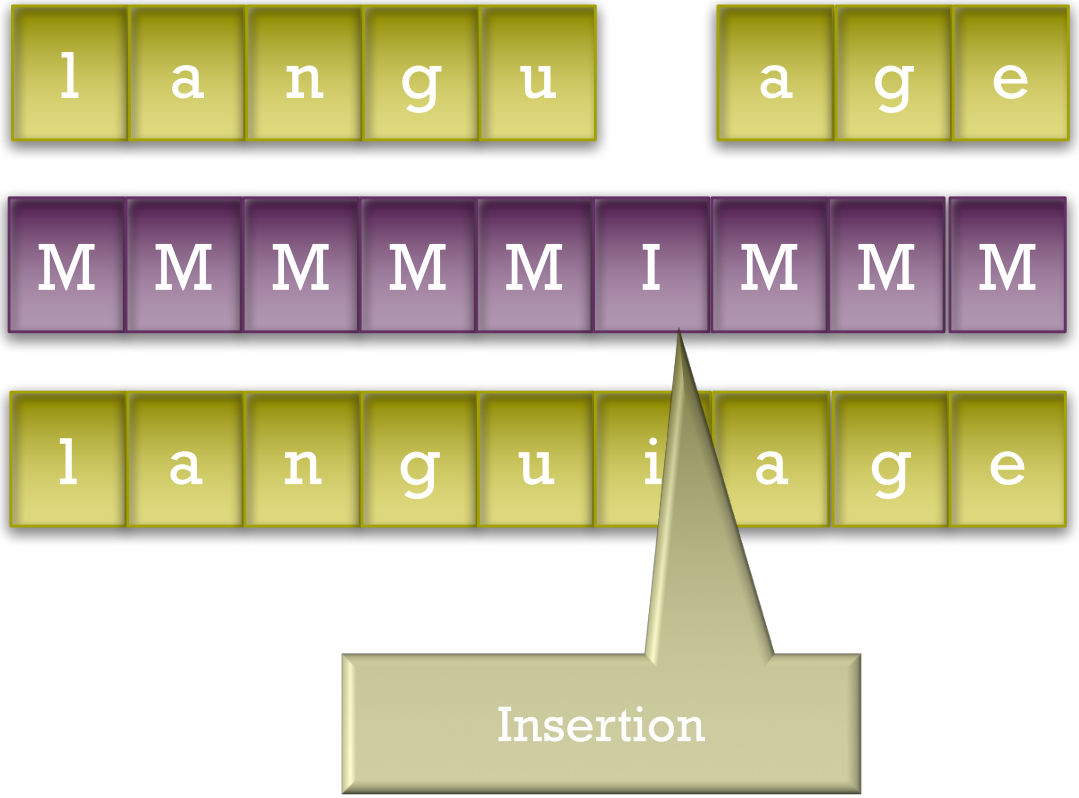
l	a	n	g	u	i	a	g	e

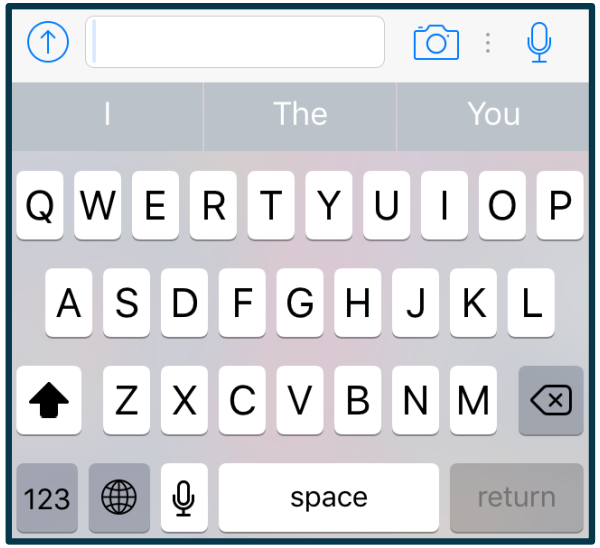


Spelling Errors and Distance

HLT10

Using Edit-Distance to find the minimum distance between two strings based on the number of operations required to transform one to the other





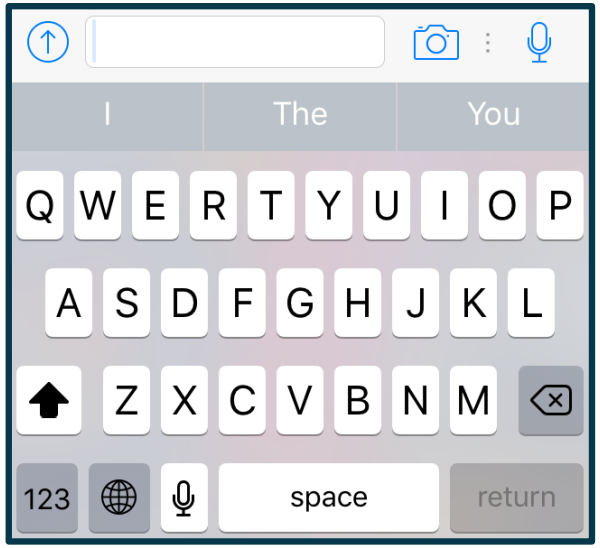
UCD DUBLIN

Spelling Errors and Distance

HLT10

Using Edit-Distance to find the minimum distance between two strings based on the number of operations required to transform one to the other

l	a	n	g	u	a	g	e
M	M	M	M	D	M	M	M
l	a	n	g		a	g	e

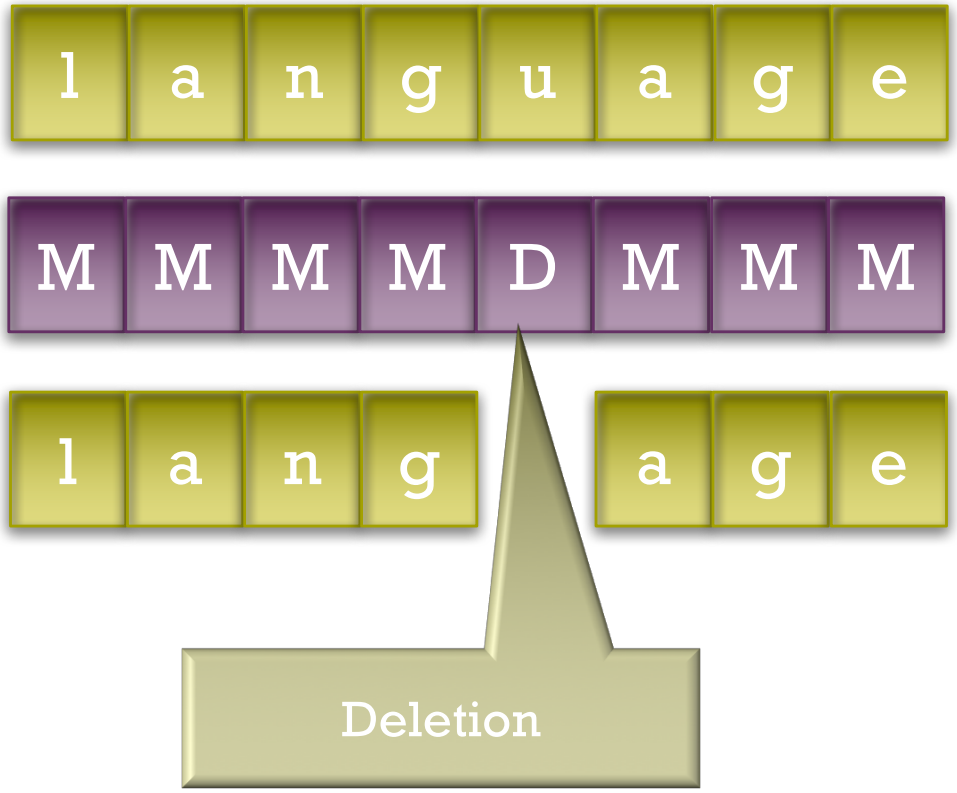


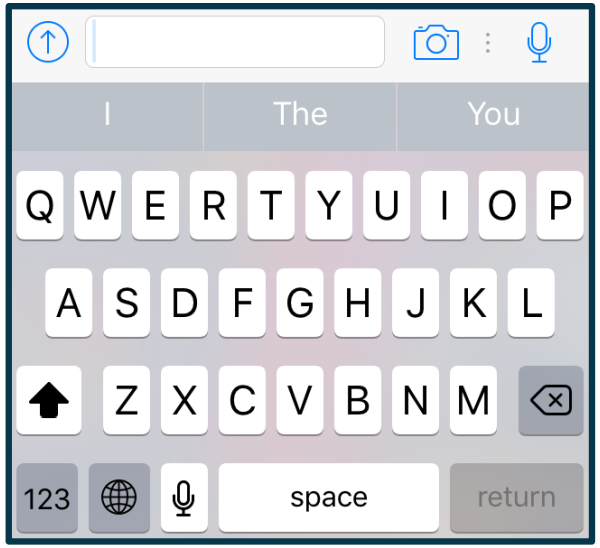
UCD DUBLIN

Spelling Errors and Distance

HLT10

Using Edit-Distance to find the minimum distance between two strings based on the number of operations required to transform one to the other





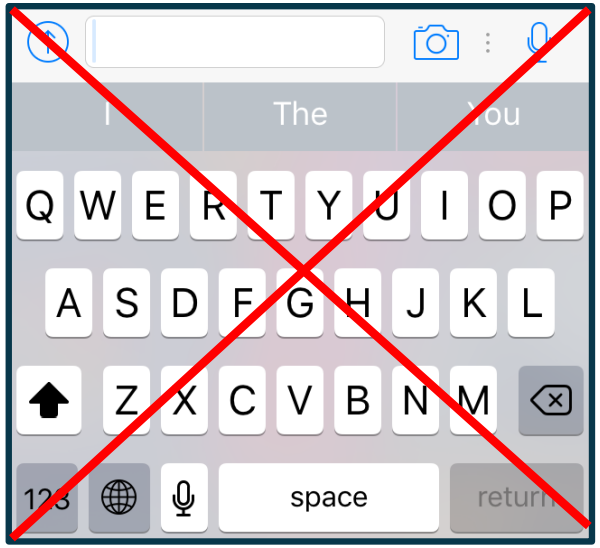
UCD DUBLIN

Spelling Errors and Distance

HLT10

Using Edit-Distance to find the minimum distance between two strings based on the number of operations required to transform one to the other

l	a	n	g	u	a	g	e
M	M	M	M	S	M	M	M
l	a	n	g	w	a	g	e

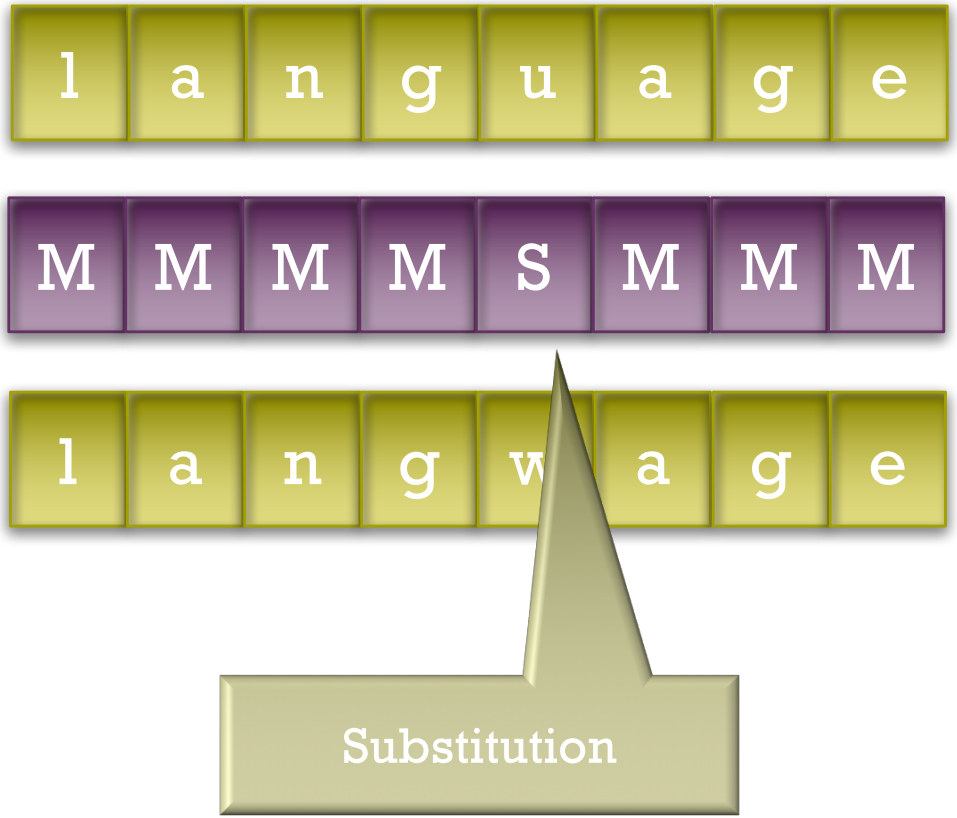


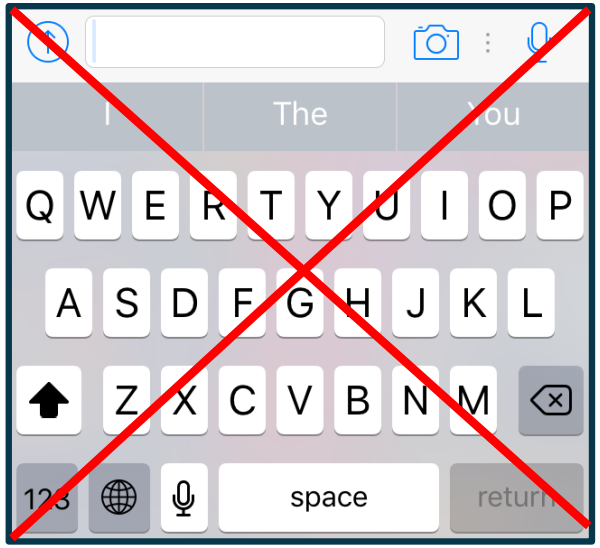
UCD DUBLIN

Spelling Errors and Distance

HLT10

Using Edit-Distance to find the minimum distance between two strings based on the number of operations required to transform one to the other





UCD DUBLIN

Spelling Error Data

Confusion matrix for spelling errors

sub[X, Y] = Substitution of X (incorrect) for Y (correct)

X	Y (correct)																								
	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
a	0	0	7	1	342	0	0	2	118	0	1	0	0	3	76	0	0	1	35	9	9	0	1	0	5	0
b	0	0	9	9	2	2	3	1	0	0	0	5	11	5	0	10	0	0	2	1	0	0	8	0	0	0
c	6	5	0	16	0	9	5	0	0	0	1	0	7	9	1	10	2	5	39	40	1	3	7	1	1	0
d	1	10	13	0	12	0	5	5	0	0	2	3	7	3	0	1	0	43	30	22	0	0	4	0	2	0
e	388	0	3	11	0	2	2	0	89	0	0	3	0	5	93	0	0	14	12	6	15	0	1	0	18	0
f	0	15	0	3	1	0	5	2	0	0	0	3	4	1	0	0	0	6	4	12	0	0	2	0	0	0
g	4	1	11	11	9	2	0	0	0	1	1	3	0	0	2	1	3	5	13	21	0	0	1	0	3	0
h	1	8	0	3	0	0	0	0	0	0	2	0	12	14	2	3	0	3	1	11	0	0	2	0	0	0
i	103	0	0	0	146	0	1	0	0	0	0	6	0	0	49	0	0	0	2	1	47	0	2	1	15	0
j	0	1	1	9	0	0	1	0	0	0	0	2	1	0	0	0	0	0	5	0	0	0	0	0	0	0
k	1	2	8	4	1	1	2	5	0	0	0	0	5	0	2	0	0	0	6	0	0	0	4	0	0	3
l	2	10	1	4	0	4	5	6	13	0	1	0	0	14	2	5	0	11	10	2	0	0	0	0	0	0
m	1	3	7	8	0	2	0	6	0	0	4	4	0	180	0	6	0	0	9	15	13	3	2	2	3	0
n	2	7	6	5	3	0	1	19	1	0	4	35	78	0	0	7	0	28	5	7	0	0	1	2	0	2
o	91	1	1	3	116	0	0	0	25	0	2	0	0	0	0	14	0	2	4	14	39	0	0	0	18	0
p	0	11	1	2	0	6	5	0	2	9	0	2	7	6	15	0	0	1	3	6	0	4	1	0	0	0
q	0	0	1	0	0	0	27	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
r	0	14	0	30	12	2	2	8	2	0	5	8	4	20	1	14	0	0	12	22	4	0	0	1	0	0
s	11	8	27	33	35	4	0	1	0	1	0	27	0	6	1	7	0	14	0	15	0	0	5	3	20	1
t	3	4	9	42	7	5	19	5	0	1	0	14	9	5	5	6	0	11	37	0	0	2	19	0	7	6
u	20	0	0	0	44	0	0	0	64	0	0	0	0	2	43	0	0	4	0	0	0	0	2	0	8	0
v	0	0	7	0	0	3	0	0	0	0	0	1	0	0	1	0	0	0	8	3	0	0	0	0	0	0
w	2	2	1	0	1	0	0	2	0	0	1	0	0	0	0	7	0	6	3	3	1	0	0	0	0	0
x	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	9	0	0	0	0	0	0	0
y	0	0	2	0	15	0	1	7	15	0	0	0	2	0	6	1	0	7	36	8	5	0	0	1	0	0
z	0	0	0	7	0	0	0	0	0	0	0	7	5	0	0	0	0	2	21	3	0	0	0	0	3	0

Example from Jurafsky: https://web.stanford.edu/class/cs124/lec/med.pdf

HLT10

UCD DUBLIN