7. Language as a Sequence

Mariana Romanyshyn, *Grammarly, Inc.*

NLP Viewpoints

* Bag-of-words
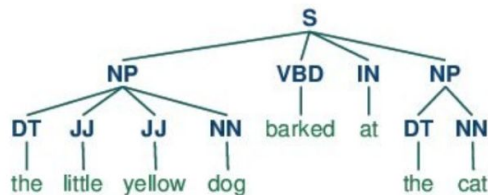
* Sequence

* Tree

* Graph

Contents

———

1. Sequence labeling
2. Hidden Markov model
3. Logistic regression
4. Feature encoding
5. Conditional random fields
6. More about ngrams

1. Sequence Labeling

Bag of words vs. sequence

_ _ _

Trump beat *Clinton* *in the election.*

= or **≠**

Clinton beat *Trump* *in the election.*

Sequence Labeling

———

Part-of-speech tagging:

DT	NN	VBD	NNS	IN	DT	DT	NN	CC	DT	NN	.
The	pound	extended	losses	against	both	the	dollar	and	the	euro	.

Sequence Labeling

———

Named-entity recognition:

PER	O	PER	PER	PER	O	O	O	O	ORG	O
Fred	showed	Sue	Mengzui	Huang	's	painting	in	the	Met	.

Sequence Labeling

———

Named-entity recognition:

B-PER	O	**B-PER**	**B-PER**	**I-PER**	O	O	O	O	**B-ORG**	O
Fred	showed	Sue	Mengzui	Huang	's	painting	in	the	Met	.

Sequence Labeling

———

Error detection:

+	+	+	x	+	+	+	+	+	x	+
I	like	to	playing	the	guitar	and	sing	very	louder	.

Sequence Labeling

_ _ _

Word segmentation:

S	S	B	E	B	M	E	S
我	有	一	台	计	算	机	。
(I)	(have)	(a)		(computer)			(.)

Sequence Labeling

– – –

Semantic role labeling:

The police officer detained the suspect at the scene of the crime

Agent   Predicate   Theme   Location

Sequence Labeling

– – –

Genome analysis:

intron   exon   intron   exon   intron

AGCTAACGTTCGATACGGATTACAGCCT

Sequence Labeling

———

There's more:
- dialogue act tagging
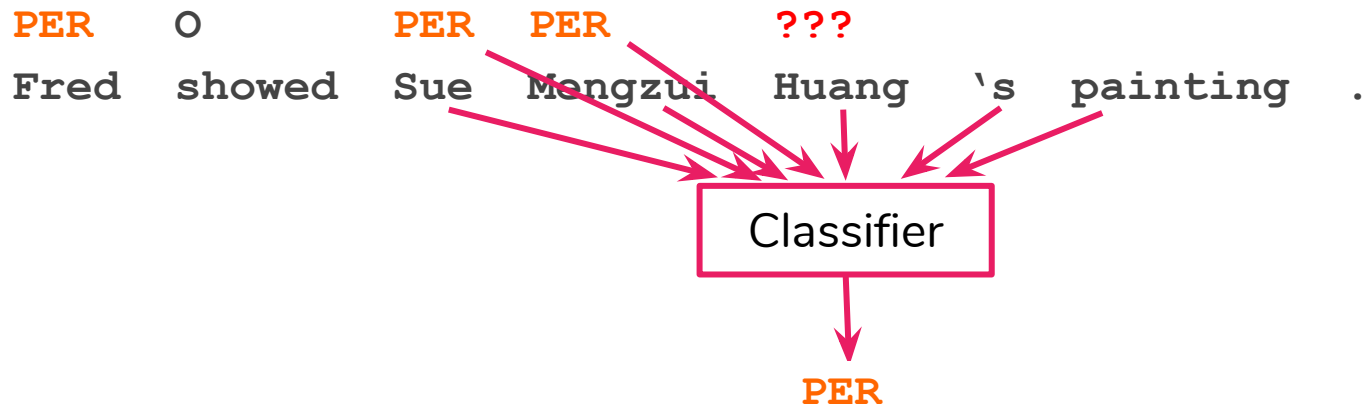- lexical stress and pitch accent detection
- sentence segmentation
- chunking
- etc.

Sequence Labeling

Sequence labeling is essentially a **classification** of each incoming element taking into account left and right context.

PER	O	PER	PER	???			
Fred	showed	Sue	Mengzui	Huang	's	painting	.

Classifier

PER

Models for sequence labelling

−−−

- HMM - generative, classifies the whole sequence at once
  - $p(x, y)$
- MaxEnt - discriminative, classifies elements one by one
  - $p(y_i=1|x_i)$
- CRFs - discriminative, classifies the whole sequence at once
  - $p(y|x)$

2. Hidden Markov Model

Hidden Markov Model

———

HMM - a **generative** probabilistic sequence model used for:
- speech recognition
- segmentation (words, sentences, genomes)
- NER
- POS tagging

HMM for POS tagging: notation

− − −

V - vocabulary

T - POS tags

x - sentence (observation)

y - tag sequences (state)

S - all sentence/tag-sequence pairs $\{x_1 \ldots x_n, y_1 \ldots y_n\}$

- $n > 0$
- $x_i \in V$
- $y_i \in T$

18

HMM for POS tagging

— — —

S - all sentence/tag-sequence pairs $\{x_1 \ldots x_n, y_1 \ldots y_n\}$

x:	Chewie	,	we	're	home	.
y:	NNP	,	PRP	VBP	RB	.
	NN	,	PRP	VBP	RB	.
	NNP	,	PRP	VBP	NN	.
	NN	,	PRP	VBP	NN	.
	...					

Aim: find $\{x_1 \ldots x_n, y_1 \ldots y_n\}$ with the highest probability.

Hidden Markov Model: assumptions

— — —

- Markov Assumption: "*The **future** is independent of the **past** given the **present**.*"
  - Trigram HMM: each state depends only on the previous two states in the sequence

- Independence assumption:
  - the state of x_i depends only on the value of x_i, independent of the previous observations and states

Hidden Markov Model: assumptions

－－－

S - all sentence/tag-sequence pairs $\{x_1 \dots x_n, y_1 \dots y_n\}$

x:	Chewie	,	we	're	home	.
y:	NNP	,	PRP	VBP	?	.
	NN	,	PRP	VBP	?	.
	...					

Trigram Hidden Markov Model: parameters

$$p(x_1 \ldots x_n, y_1 \ldots y_{n+1}) = \prod_{i=1}^{n+1} q(y_i | y_{i-2}, y_{i-1}) \prod_{i=1}^{n} e(x_i | y_i)$$

- q(s|u, v) - the probability of tag s after the tags (u, v)
  - s, u, v ∈ T

- e(x|s) - the probability of observation x paired with state s
  - x ∈ V, s ∈ T

Trigram Hidden Markov Model: parameters

———

- q(s|u, v) - the probability of tag s after the tags (u, v)

$$q(s|u, v) = \frac{c(u, v, s)}{c(u, v)}$$

- e(x|s) - the probability of observation x paired with state s

$$e(x|s) = \frac{c(s \rightsquigarrow x)}{c(s)}$$

For example

———

x: Chewie , we 're home .

y: *NNP* , *PRP* *VBP* *RB* .

p(x, y) = ?

For example

‒ ‒ ‒

x: Chewie   ,   we   're   home   .
y: NNP   ,   PRP   VBP   RB   .

p(x, y) = c(NNP,|,|,PRP) / c(NNP,|,|) * c(|,|,PRP, VBP) / c(|,|,PRP) *
              c(PRP, VBP,RB) / c(PRP,VBP) * c(VBP,RB,|.|) / c(VBP,RB) * ...

For example

———

x: Chewie   ,   we   're   home   .
y: *NNP*   ,   *PRP* *VBP* *RB*   .

p(x, y) = c(NNP,|,|,PRP) / c(NNP,|,|) * c(|,|,PRP, VBP) / c(|,|,PRP) *
        c(PRP, VBP,RB) / c(PRP,VBP) * c(VBP,RB,|.|) / c(VBP,RB) *
        c(NNP->Chewie) / c(NNP) * c(|,|->,) / c(|,|) *
        c(PRP->we) / c(PRP) * c(VBP->'re) / c(VBP) *
        c(RB->home) / c(RB) * c(|.|->.) / c(|.|)

One thing missing

– – –

x: Chewie   ,   we   're   home   .

y: <S> <S> NNP   ,   PRP VBP   RB   . </S>

p(x, y) = c(NNP,|,|,PRP) / c(NNP,|,|) * c(|,|,PRP, VBP) / c(|,|,PRP) *
c(PRP, VBP,RB) / c(PRP,VBP) * c(VBP,RB,|.|) / c(VBP,RB) *
c(<S>,<S>,NNP) / c(<S>,<S>) * c(<S>,NNP,|,|) / c(<S>,NNP) *
c(RB,|.|,</S>) / c(RB,|.|)* c(NNP->Chewie) / c(NNP) * c(|,|->,) / c(|,|) *
c(PRP->we) / c(PRP) * c(VBP->'re) / c(VBP) *
c(RB->home) / c(RB) * c(|.|->.) / c(|.|)

HMM: problem 1

– – –

Enumerating all possible tag sequences is not feasible — T^n.

44 tags ** 6-token sentence = 7,256,313,856 tag sequences

Solutions:
- use dynamic programming (the Viterbi algorithm) - $n*T^3$
- limit the number of candidates with a dictionary - $n*8^3$
- use beam search

HMM: problem 2

_ _ _

Zero probabilities can occur because of OOV or rare words.

Solution: use *smoothing*
- **add-1:** pretend you saw each word (or each new word) one more time (also: **add-k**)
- **Good-Turing:** when the trigram count is near 0, rely on bigram
- **Kneser-Ney:** reallocate the probability of ngrams that occur *r+1* times to the ngrams that occur *r* times

HMM: problem 3

———

Limited features taken into account:

- **p(tag|word)** could be informative

- incorporating lemmas, grammatical properties, spelling properties, etc. is hard

3. Logistic Regression

Logistic Regression

———

Logistic regression - a **discriminative** linear model used for binary classification.

- like Perceptron, it's linear
- like NB, it extracts a set of weighted features, takes logs, and combines them linearly
- unlike NB, it's discriminative

$$z = \left(\sum_{i=1}^{n} w_i x_i \right)$$

Logistic Regression

A [0; 1] function would be handy: y = 1 if p(y=1|x) > 0.5.

Sigmoid function:

$$P(y=1) = \sigma(w \cdot x + b)$$
$$= \frac{1}{1+e^{-(w \cdot x + b)}}$$
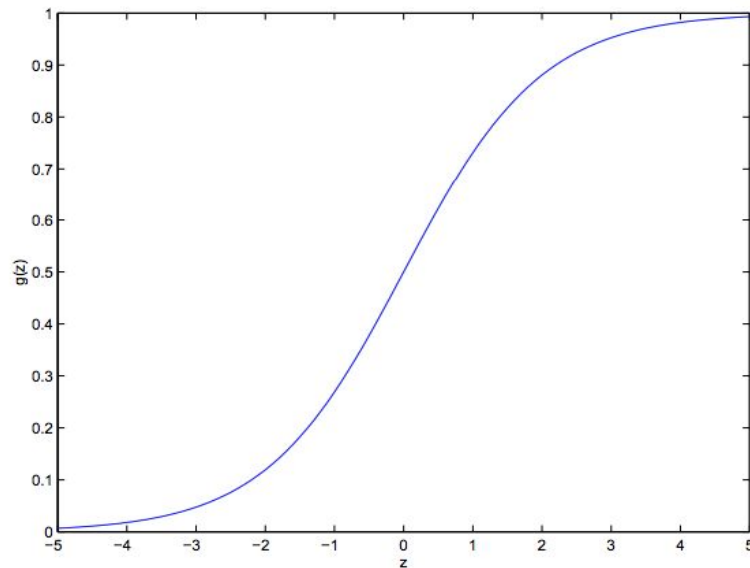
Logistic Regression: multiclass

— — —

one vs. rest   multinomial (MaxEnt)

http://scikit-learn.org/stable/auto_examples/linear_model/plot_logistic_multinomial.htm
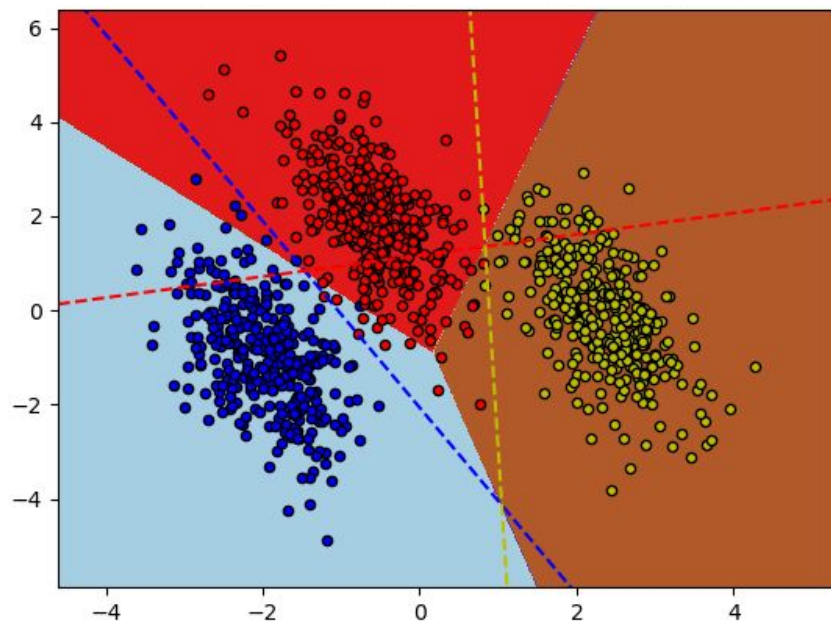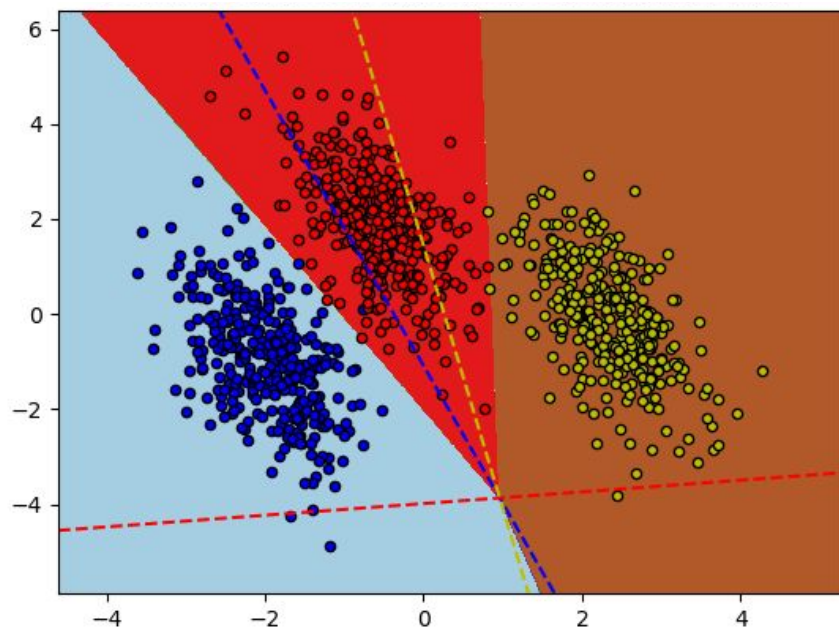
Logistic Regression

– – –

For multinomial logistic regression, use softmax:

$$p(c|x) \; = \; \frac{\exp\left(\sum\limits_{i=1}^{N} w_i f_i(c,x)\right)}{\sum\limits_{c' \in C} \exp\left(\sum\limits_{i=1}^{N} w_i f_i(c',x)\right)}$$

Logistic Regression: example

— — —

Welcome to St . Paul 's Cathedral !

[Is this period a sentence end?]

Logistic Regression: example

_ _ _

Welcome to St . Paul 's Cathedral !

[Is this period a sentence end?]

y: *{is-end, is-not-end}*
x: {"word+1_is_cap", "word-1=kittens", "word-1=St", "tag-1=PRP", "tag+1=JJ"}

Logistic Regression: example

— — —

Welcome to St . Paul 's Cathedral !

[Is this period a sentence end?]

y: *{is-end, is-not-end}*
x: *{"word+1_is_cap", "word-1=kittens", "word-1=St", "tag-1=PRP", "tag+1=JJ"}*

x$_j$*: [1, 0, 1, 0, 0]*

Logistic Regression: example

– – –

Welcome to St . Paul 's Cathedral !

[Is this period a sentence end?]

y: *{is-end, is-not-end}*
x: *{"word+1_is_cap", "word-1=kittens", "word-1=St", "tag-1=PRP", "tag+1=JJ"}*

x$_j$: *[1, 0, 1, 0, 0]*

w$_{is\text{-}end}$: *[2.9, 2.5, -0.9, 0, 0]*
w$_{is\text{-}not\text{-}end}$: *[0.5, -0.7, 2.9, 0, 0]*

Logistic Regression: example

———

Welcome to St . Paul 's Cathedral !

[Is this period a sentence end?]

y: *{is-end, is-not-end}*
x: *{"word+1_is_cap", "word-1=kittens", "word-1=St", "tag-1=PRP", "tag+1=JJ"}*

x$_j$*: [1, 0, 1, 0, 0]*

w$_{is\text{-}end}$*: [2.9, 2.5, -0.9, 0, 0]* $P(is\text{-}end|x_j) = e^{2.9\text{-}0.9} / (e^{2.9\text{-}0.9} + e^{0.5+2.9}) = 0.2$
w$_{is\text{-}not\text{-}end}$*: [0.5, -0.7, 2.9, 0, 0]* $P(is\text{-}not\text{-}end|x_j) = e^{0.5+2.9} / (e^{2.9\text{-}0.9} + e^{0.5+2.9}) = 0.8$

Logistic Regression: weights

\- \- \-

Learn weights:
- start with a vector of zeros
- move towards the gradient
- to maximize the probability / minimize the loss function

$$\hat{w} = \underset{w}{\mathrm{argmax}} \sum_{j} \log P(y^{(j)}|x^{(j)})$$

4. Feature Encoding

Encode neighbors: feature template

— — —

DT	NN	VBD	NNS	IN	DT	DT	NN	CC	DT	NN	.
The	pound	extended	losses	against	both	the	dollar	and	the	euro	.

"losses"/NNS:

{"word-2": "pound",   "tag-2": "NN",
 "word-1":   "tag-1": "VBD",
"extended",   "tag+1": "IN",
 "word+1": "against",   "tag+2": "DT"}
 "word+2": "both",

43

Encode neighbors: feature template

———

DT	NN	VBD	NNS	IN	DT	DT	NN	CC	DT	NN	.
The	pound	extended	losses	against	both	the	dollar	and	the	euro	.

[1,   1,   0,   ...]
  word-2=pound   *word-1=extended*   *word+1=hello*

Encode neighbors: feature template

– – –

DT	NN	VBD	NNS	IN	DT	DT	NN	CC	DT	NN	.
The	pound	extended	losses	against	both	the	dollar	and	the	euro	.

"losses"/NNS:
{"wt-2": "pound_NN",
 "wt-1": "extended_VBD",
 "wt+1": "against_IN",
 "wt+2": "both_DT"}

Encode context (ngrams)

– – –

DT	NN	VBD	NNS	IN	DT	DT	NN	CC	DT	NN	
The	pound	extended	losses	against	both	the	dollar	and	the	euro	.

"losses"/NNS:
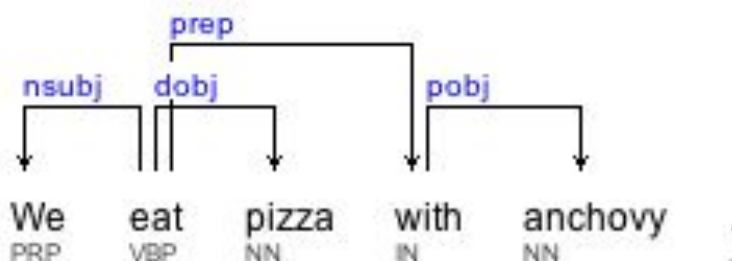*{"left-bigram":   "pound extended",*
 "right-bigram": "against both",
 "context":   "extended losses against both"}

Encode dependencies

"eat"/VBP:
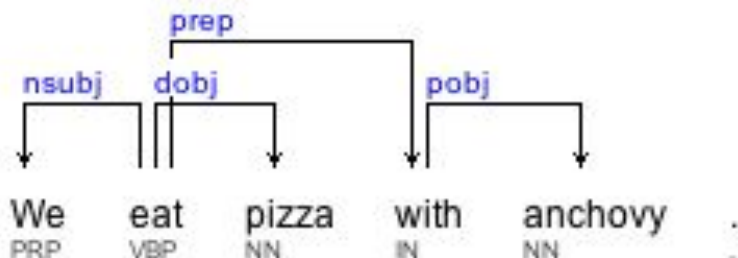
[1,   0,   0,   1,   0,   1,   0,   0,   ...]
nsubj   acl   relcl   dobj   pobj   prep   punct   xcomp

Encode dependencies

_ _ _

"eat"/VBP:
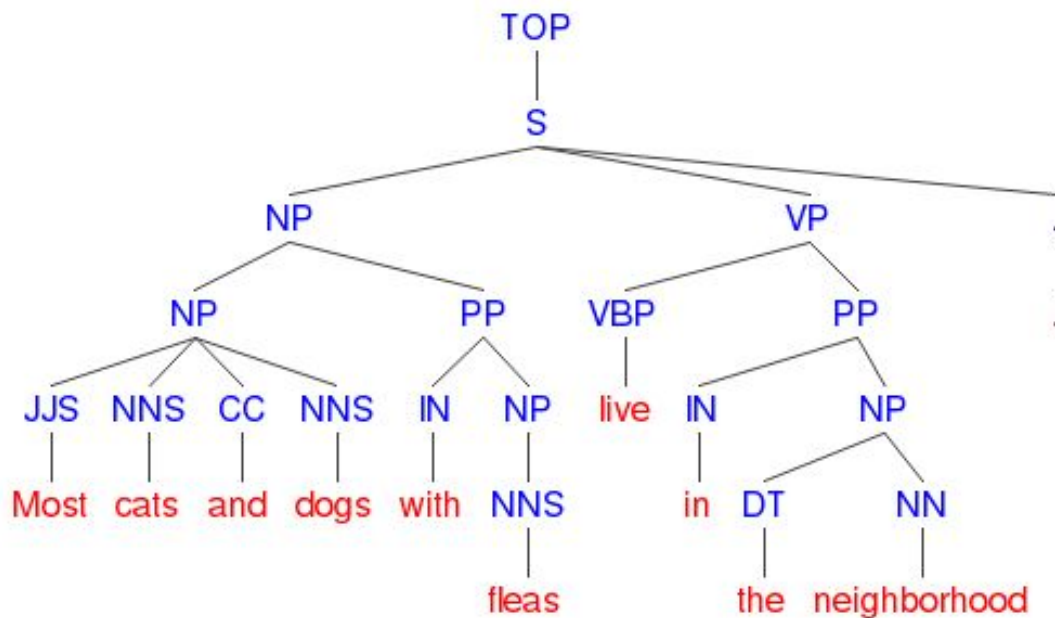
- *nsubj_We, dobj_pizza, prep_with*
- *nsubj_PRP, dobj_NN, prep_IN*
- *nsubj_We, dobj_pizza, prep_with_pobj_anchovy*

Encode constituents

———

"fleas"/NNS:

{*"label": "NP",*
 "anc-left": "PP",
 "anc-right": "S",
 "span-start": 5,
 "span-end": 6}

More features

———

- affixes
- coreference
- sentiment
- grammatical characteristics of various parts of speech:
    - countability of nouns
    - tense of verbs
    - degree of comparison of adjectives
    - pronoun type
    - conjunction type

More features

– – –

- *capitalized?*
- *hyphenated?*
- *compound?*
- *lemma*
- *sense id*
- *number of senses in WordNet*
- *is in X dictionary*
- *has X as a synonym*
- *...*

What features to use?

———

DT	NN	VBD	NNS	IN	DT	DT	NN	CC	DT	NN	.
The	pound	extended	losses	against	both	the	dollar	and	the	euro	.

- *gold or predicted?*

5. Conditional Random Fields

Conditional Random Fields

———

CRFs = MaxEnt + HMM

- HMM - generative, **classifies the whole sequence at once**
  - *p(x, y)*
- MaxEnt - **discriminative**, classifies elements one by one
  - *$p(y_i=1|x_i)$*
- CRFs - **discriminative, classify the whole sequence at once**
  - *p(y|x)*

Conditional Random Fields

- MaxEnt

$$p(c|x) = \frac{\exp\left(\sum_{i=1}^{N} w_i f_i(c,x)\right)}{\sum_{c' \in C} \exp\left(\sum_{i=1}^{N} w_i f_i(c',x)\right)}$$

- CRF also learns transitions

$$p(l|s) = \frac{exp(\sum_{j=1}^{m} \sum_{i=1}^{n} w_j f_j(s,i,l_i,l_{i-1}))}{\sum_{l'} exp(\sum_{j=1}^{m} \sum_{i=1}^{n} w_j f_j(s,i,l'_i,l'_{i-1}))}$$

Conditional Random Fields

‒ ‒ ‒

$$p(l|s) = \frac{exp(\sum_{j=1}^{m} \sum_{i=1}^{n} w_j f_j(s, i, l_i, l_{i-1}))}{\sum_{l'} exp(\sum_{j=1}^{m} \sum_{i=1}^{n} w_j f_j(s, i, l'_i, l'_{i-1}))}$$

s - sentence (list of words)
l - list of labels

i - index of a word in s
l_i - label of the word *i*
l_{i-1} - label of the word before *i*

6. More about Ngrams

What are ngrams

———

Ngram - a contiguous sequence of *n* items from a given text.

What are ngrams

– – –

Ngram - a contiguous sequence of **n** items from a given text.

So, if **n = 3**:

<S> *Why* *did* *n't* *you* *listen* *to* *me* *?* </S>

What are ngrams

– – –

Ngram - a contiguous sequence of **n** items from a given text.

So, if **n = 3**:

<S> *Why   did   n't*   *you   listen   to   me   ?   </S>*

What are ngrams

———

Ngram - a contiguous sequence of **n** items from a given text.

So, if **n = 3**:

<S> Why did n't you listen to me ? </S>

What are ngrams

–––

Ngram - a contiguous sequence of **n** items from a given text.

So, if **n = 3**:

<S>   Why   did   n't   you   listen   to   me   ?   </S>

What are ngrams

– – –

Ngram - a contiguous sequence of **n** items from a given text.

So, if **n = 3**:

<S> *Why* *did* *n't* *you* *listen* *to* *me* *?* *</S>*

What are ngrams

———

Ngram - a contiguous sequence of *n* items from a given text.

So, if *n = 3*:

<S> *Why* *did* *n't* *you* *listen* *to* *me* ? </S>

What are ngrams

– – –

Ngram - a contiguous sequence of **n** items from a given text.

So, if **n = 3**:

<S>   Why   did   n't   you   listen   to   me   ?   </S>

What are ngrams

– – –

Ngram - a contiguous sequence of **n** items from a given text.

So, if **n = 3**:

<S> Why did n't you listen to me ? </S>

Token ngrams

–––

Usually 1 ≥ n ≥ 5.

<S>   Why   did   n't   you   listen   to   me   ?   </S>

n = 1: (<S>), (Why), (did), (n't), (you), (listen), (to), (me), (?)…

n = 2: (<S> Why), (Why did), (did n't), (n't you), (you listen), (listen to)…

n = 3: (<S> Why did), (Why did n't), (did n't you), (you listen to)…

…

Character Ngrams

– – –

<S> Why did n't you listen to me ? </S>

For words:

n = 3: (<w> W h), (W h y), (h y </w>), (<w> d i), (d i d), (i d n), (d n ')...

For sentences:

n = 3: (W h y), (h y _), (y _ d), (_ d i), (d i d), (i d n), (d n '), (n ' t)...

POS Ngrams

– – –

<S> Why did n't you listen to me ? </S>
<S> WDT VDB RB PRP VB TO PRP . </S>

POS:

n = 3: (<S>, WDT, VBD), (WDT, VBD, RB), (VBD, RB, PRP), (RB, PRP, VB)…

Token+POS:

n = 2: (<S>_<S>, Why_WDT), (Why_WDT, did_VBD), (did_VBD, n't_RB)…

Token or POS:

n = 3: (<S>, WDT, did), (WDT, did, RB), (did, RB, PRP), (RB, PRP, listen)…
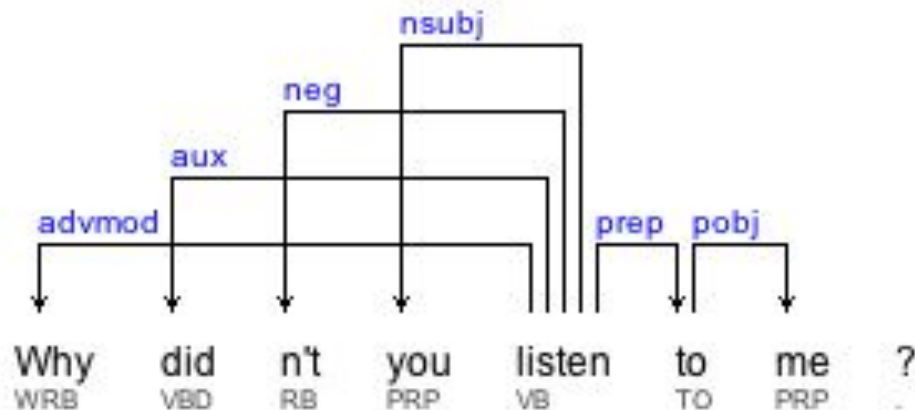
Tree Ngrams

— — —

Head+dependency:
listen_nsubj
listen_nsubj_you
listen_prep_to_pobj_me

Head+POS+dependency:
listen/VB_nsubj
listen/VB_nsubj_you/PRP

Ngrams usage

– – –

1. Speech recognition
2. Autocompletion

Ngrams usage

— — —

1. Speech recognition
2. Autocompletion
3. Machine Translation

Ngrams usage

– – –
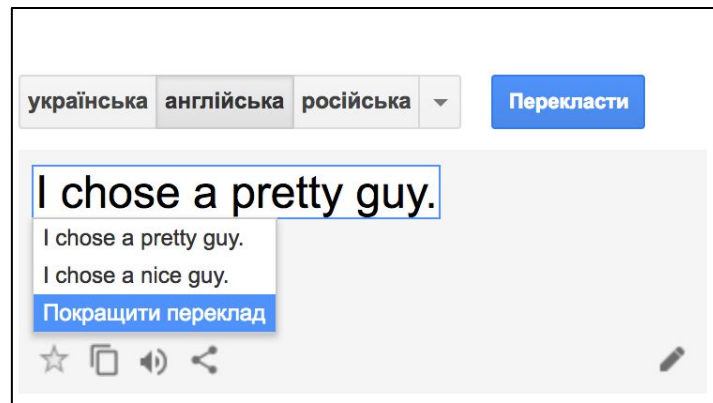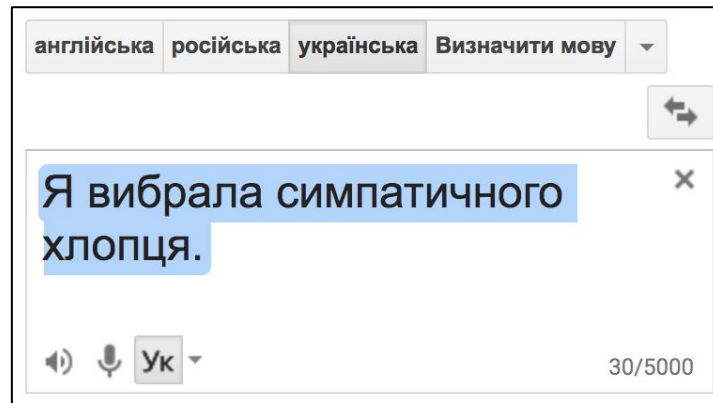
1. Speech recognition
2. Autocompletion
3. Machine Translation
4. Handwriting recognition
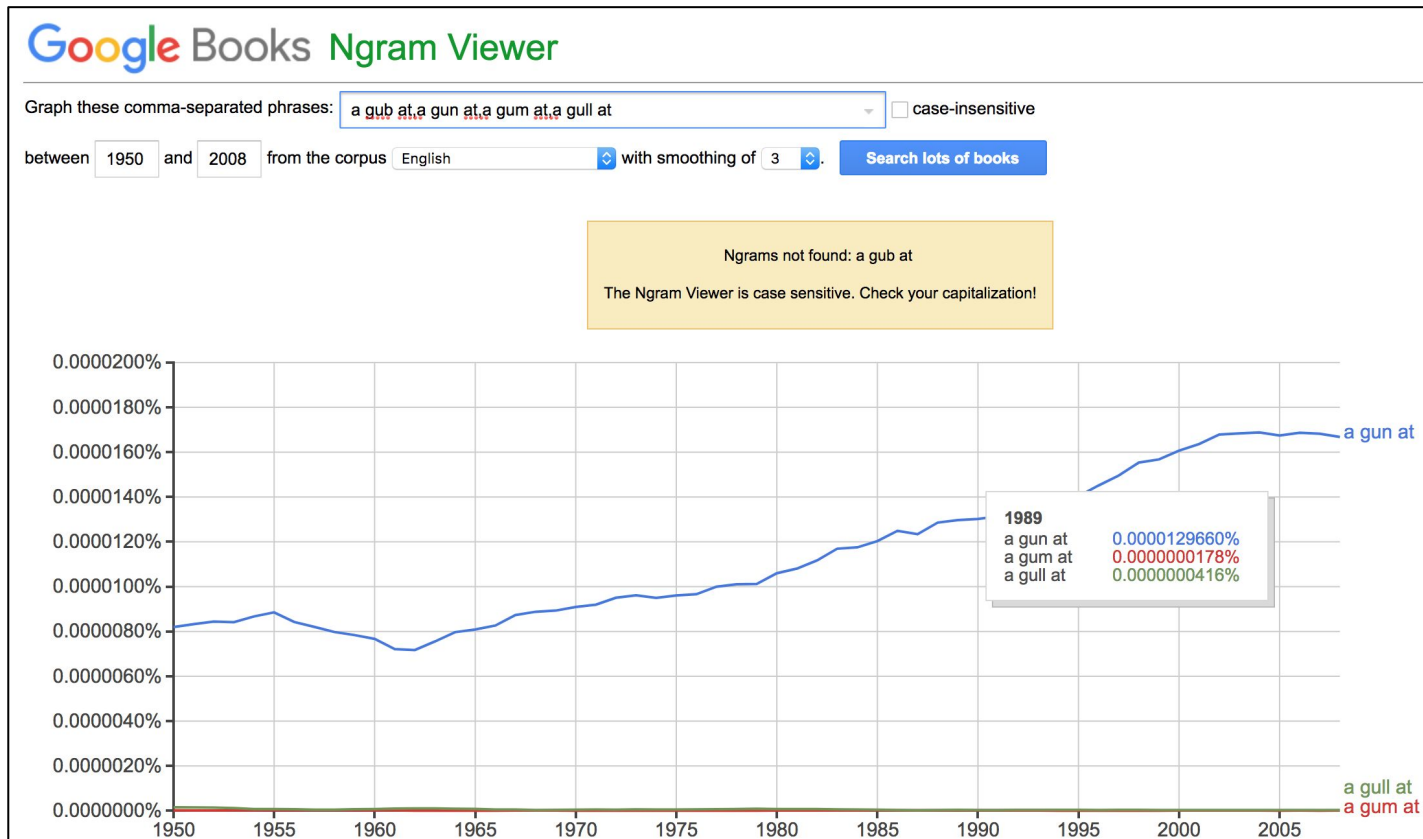5. Spelling correction
6. (and GEC in general)

"I am pointing a gub at you." What's gub?

Borrowed from https://www.youtube.com/watch?v=l3pdGfKycz8

Ngrams as a feature

———

Frequency or probability:

a *gub* at
a *gun* at
a *gum* at
a *gull* at

Ngrams as a feature

———

Frequency / probability

a cute guy
a handsome guy

a beautiful guy

a pretty guy

1995   2000   2005

Ngrams as a feature

\- \- \-
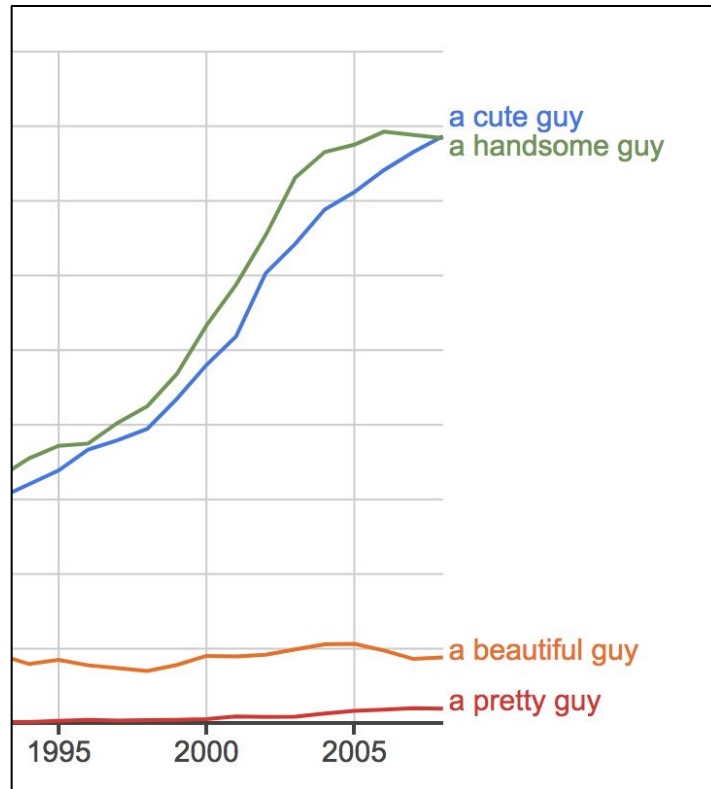
Frequency or probability

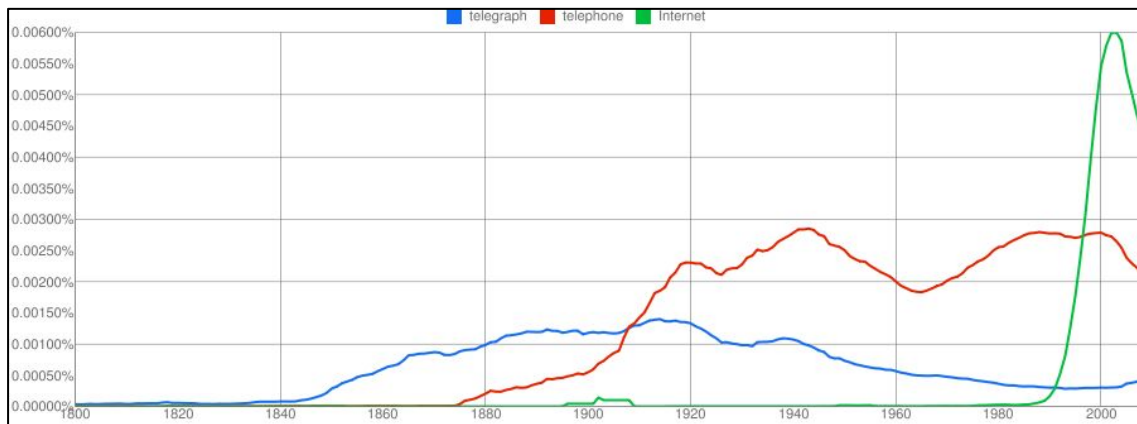|	
at	0.1
by	0.2
for	0.1
He will take our place **in** the line. ⟶	**0.3**
from	0.0
to	0.1
with	0.1

Ngrams as a feature

–––

Conditional probability

$$P(w_n | w_{n-1}) = \frac{C(w_{n-1} w_n)}{C(w_{n-1})}$$

To be continued on the lecture about language modeling...

Where to get ngrams

— — —

- [1 mln of 2/3/4/5-ngrams from COCA](#) for free
- [Google ngrams](#) (and [how to download](#))
- [Google syntactic ngrams](#)
- collect on your own

How to encode ngram frequencies

— — —

Ngrams:

"met a cute": 3250, "a cute guy": 25289, "met a cute guy": 600, ...
"met a pretty": 2925, "a pretty guy": 1159, "met a pretty guy": 0, ...

- As additional vector to concatenate:
  - *[3250, 25289, 600, 2925, 1159, 0, ...]*

- As part of the feature dictionary:
  - {"left-3gr": 3250, "right-3gr": 25289, "middle-4gr": 600,
    "left-3gr-2": 2925, "right-3gr-2": 1159, "middle-4gr-2": 0, ...}

References

———

1. *Logistic Regression* in *Speech and Language Processing* by D. Jurafsky and J. H. Martin (2017)
2. *Hidden Markov Models* in *Speech and Language Processing* by D. Jurafsky and J. H. Martin (2017)
3. *Introduction to Conditional Random Fields* by Edwin Chen (2012)
4. *Conditional Random Fields: An Introduction* by Hanna M. Wallach (2004)
5. *Sequence labeling* by Raymond J. Mooney (2016)
6. *Multiclass and Multi-label Classification* by Prof. Michael Paul (2017)