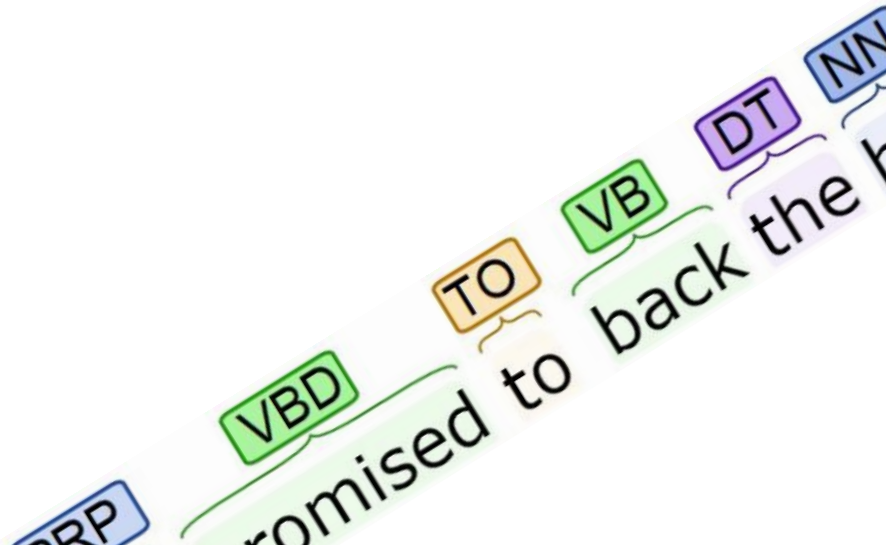
Tagging

(sequence labeling for NLP)

Anton Alekseev
Steklov Mathematical Institute in St Petersburg

ITMO University, St. Petersburg, 2019
anton.m.alexeyev+itmo@gmail.com



Plan

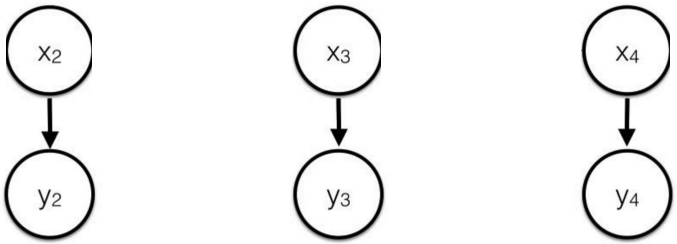
1. Motivation
2. Approaches we will discuss
3. Methods
    a. Classifiers
    b. Hidden Markov models
    c. Structured perceptron
    d. (a lil bit of) Conditional Random Fields
4. Tools and data

Motivation (very general)

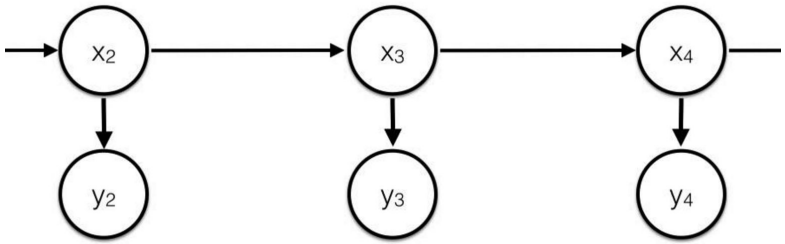
We don’t always have to work with **iid** data (independent identically distributed), which was the case with text classification, for example

Sometimes items’ order is important and the context should be taken into account

life without dependences



life without dependences



https://www.youtube.com/watch?v=9yl4XGp5OEg

Motivation: PoS tagging

Part-of-speech tagging
(aka POS tagging, word-category tagging, ...)
matching words and parts-of-speech in the text

We have already seen morphological analyzers that can offer **several suggestions of parts-of-speech per word**

Hence, most importantly -- PoS-tagging is the **disambiguation task**

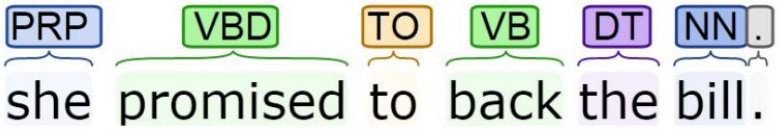
There/EX are/VBP 70/CD children/NNS **there/RB**

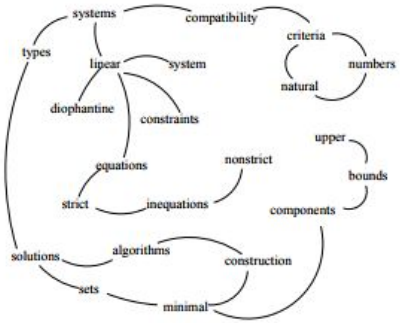
Motivation: PoS tagging

Why tag parts of speech?

Examples

- helps to extract keyphrases + allows using patterns in information extraction (very widely spread)
- PoS as a replacement for rare terms help to generalize
- useful as a first step in syntax analysis





"The vertices added to the graph can be restricted with syntactic filters, which select only lexical units of a certain part of speech..." [TextRank paper]

Table 1

Extracted phrases' patterns.

Pattern	The first word	The second word	The third word
Pattern 1	JJ	NN/NNS	—
Pattern 2	JJ	NN/NNS	NN/NNS
Pattern 3	RB/RBR/RBS	JJ	—
Pattern 4	RB/RBR/RBS	JJ/RB/RBR/RBS	NN/NNS
Pattern 5	RB/RBR/RBS	VBN/VBD	—
Pattern 6	RB/RBR/RBS	RB/RBR/RBS	JJ

Extracting Product Features and Opinion Words Using Pattern Knowledge in Customer Reviews
Su Su Htay and Khin Thidar Lynn, 2013

Parts-of-speech

PoS

1. **Closed classes**:
   they would not change

prepositions: on, under, over, near, by, at, from, to, with
determiners: a, an, the
pronouns: she, who, I, others
conjunctions: and, but, or, as, if, when
auxiliary verbs: can, may, should, are
particles: up, down, on, off, in, out, at, by
numerals: one, two, three, first, second, third

2. **Open classes**: PoS classes, where new words appear once in a while

Nouns, verbs, adjectives, adverbs, ...

There are languages with great difficulties in determining parts-of-speech

Parts-of-speech

There may be several tagsets for PoS, e.g.
Penn Treebank has 45 tags

1	Согласно	СОГЛАСНО	ADP
2	указанному	УКАЗАННЫЙ	ADJ
3	закону	ЗАКОН	NOUN
4	первые	ПЕРВЫЙ	ADJ
5	выборы	ВЫБОРЫ	NOUN
6	высших	ВЫСШИЙ	ADJ
7	должностных	ДОЛЖНОСТНОЙ	ADJ
8	лиц	ЛИЦО	NOUN
9	должны	ДОЛЖЕН	ADJ
10	были	БЫТЬ	AUX
11	состояться	СОСТОЯТЬСЯ	VERB
12	14	14	ADJ
13	октября	ОКТЯБРЬ	NOUN

Tag	Description	Example	Tag	Description	Example
CC	coordin. conjunction	*and, but, or*	SYM	symbol	+,%, &
CD	cardinal number	*one, two*	TO	"to"	*to*
DT	determiner	*a, the*	UH	interjection	*ah, oops*
EX	existential 'there'	*there*	VB	verb base form	*eat*
FW	foreign word	*mea culpa*	VBD	verb past tense	*ate*
IN	preposition/sub-conj	*of, in, by*	VBG	verb gerund	*eating*
JJ	adjective	*yellow*	VBN	verb past participle	*eaten*
JJR	adj., comparative	*bigger*	VBP	verb non-3sg pres	*eat*
JJS	adj., superlative	*wildest*	VBZ	verb 3sg pres	*eats*
LS	list item marker	*1, 2, One*	WDT	wh-determiner	*which, that*
MD	modal	*can, should*	WP	wh-pronoun	*what, who*
NN	noun, sing. or mass	*llama*	WP$	possessive wh-	*whose*
NNS	noun, plural	*llamas*	WRB	wh-adverb	*how, where*
NNP	proper noun, sing.	*IBM*	$	dollar sign	*$*
NNPS	proper noun, plural	*Carolinas*	#	pound sign	*#*
PDT	predeterminer	*all, both*	"	left quote	' or "
POS	possessive ending	*'s*	"	right quote	' or "
PRP	personal pronoun	*I, you, he*	(	left parenthesis	[, (, {, <
PRP$	possessive pronoun	*your, one's*	)	right parenthesis	],), }, >
RB	adverb	*quickly, never*	,	comma	,
RBR	adverb, comparative	*faster*	.	sentence-final punc	. ! ?
RBS	adverb, superlative	*fastest*	:	mid-sentence punc	: ; ... – -
RP	particle	*up, off*			

Figure 10.1 Penn Treebank part-of-speech tags (including punctuation).

Motivation: NER

Named entity recognition
(aka NER, entity identification, entity chunking and entity extraction) -- detection and classification of real-world named objects mentions in the texts

Examples:

- search and aggregation of **companies' and persons' names** etc. for further analysis and easy access to information: 'who is the most popular', 'who today's news are about'

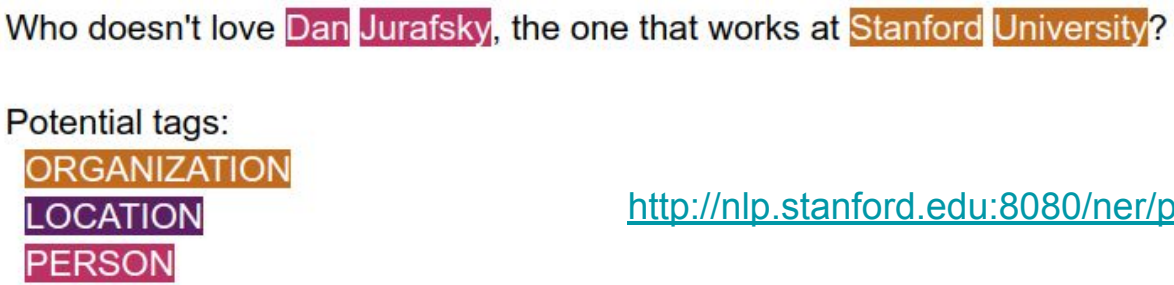
- as a preprocessing stage for more complex tasks, e.g. relation extraction

NER, tagsets types

- Oldies can rock: e.g. manually created grammars can achieve amazing precision, however, they usually have low recall and require a lot of labour of linguists

- A few tagset types, e.g. we can start the name annotation with the tag **B_xxx**, and give the rest **I_xxx**. Those that don't make up the name have **O** (BIO-markup)

- Replace **xxx** with your class name; this way one can extract entities of different types



http://nlp.stanford.edu:8080/ner/process

Plan

1. ~~Motivation~~
2. Approaches we will discuss
3. Methods
    a. Classifiers
    b. Hidden Markov models
    c. Structured perceptron
    d. (a lil bit of) Conditional Random Fields
4. Tools and data

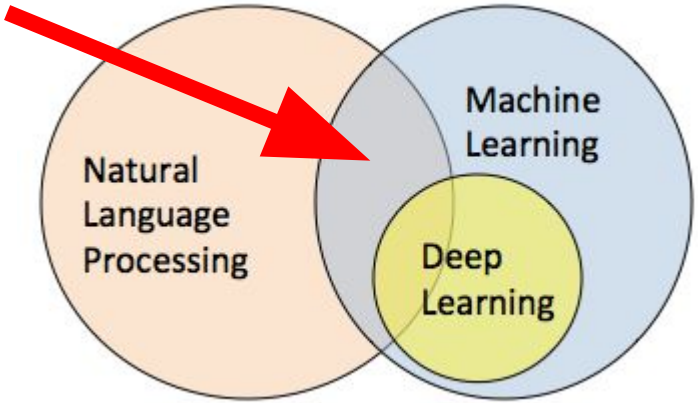
“Denial of responsibility”

PoS-tagging and NER can be solved with many different approaches
(*rules, incl. grammars, vocabularies, gazetteers, etc.*)

We will only look at these tasks as machine learning problems

sequence learning < structured learning

To justify this, I promise that you’ll find these approaches useful in many domains other than NLP



Generalization of the tasks

We have the training set

1. A sequence of observations (e.g. words)
   o_1, o_2, o_3, …
   Всё смешалось в доме Облонских
   Everything was in confusion in the Oblonskys' house
2. A sequence of states (parts of speech, borders of words)
   q'_1, q'_2, q'_3, …
   Всё/**O** смешалось/**O** в/**O** доме/**O** Облонских/**B**
   Everything/O was/O in/O confusion/O in/O the/O Oblonskys'**/B** house/O

Build a model, using which we can

- decode **the most probable sequence** of states given a sequence of observations
- *estimate the probability of tagging

Plan

1. ~~Motivation~~
2. ~~Approaches we will discuss~~
3. Methods
    a. Classifiers
    b. Hidden Markov models
    c. Structured perceptron
    d. (a lil bit of) Conditional Random Fields
4. Tools and data

Simple idea: many classifiers

Train a classifier on features built based on the features of nearby words and predict labels one by one?

1. Yeah, can be done, try it out in your homework :)

2. Problem the method doesn’t take neighbours **labels** into account

Plan

1. ~~Motivation~~
2. ~~Approaches we will discuss~~
3. ~~Methods~~
    a. ~~Classifiers~~
    b. Hidden Markov models
    c. Structured perceptron
    d. (a lil bit of) Conditional Random Fields
4. Tools and data

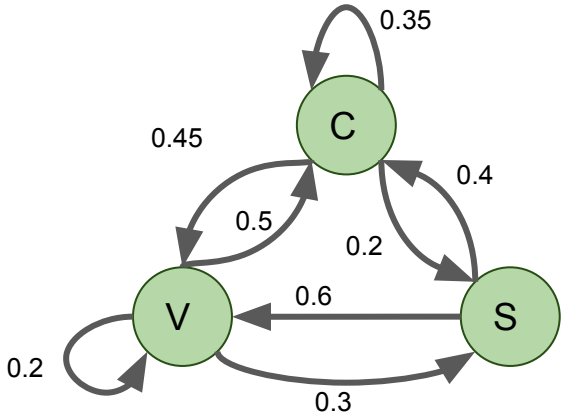
REMINDER: Markov chain

Is set by a stochastic transition matrix

Example. Events: vowel (v), consonant (c), white**s**pace/punctuation (s)
(probabilities in the example are fake)

$$P_{trans} =$$

	v	**c**	**s**
v	0.2	0.5	0.3
c	0.45	0.35	0.2
s	0.6	0.4	0.0



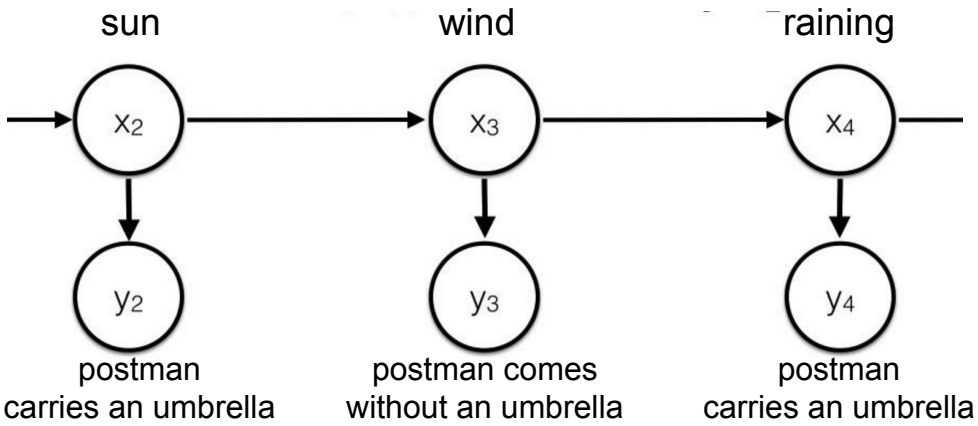
DEMO: ugly self-promotion: http://antonalexeev.hop.ru/markov/index.html

Markov chains

Markov chain actually sets a weighted finite automaton, with **conditional probabilities as transition weights**

Stochastic process set by it generates the trajectory of states, allowing to estimate those transition weights.
We have already done this! (e.g. language modeling)

Now let us imagine **we don’t see the true states of the process**.
We can only see the observations that depend on them.



Hidden Markov model

$Q = q_1 q_2 \ldots q_N$

A set of all **N** possible states

$A = a_{11} a_{12} \ldots a_{n1} \ldots a_{nn}$

Transition probabilities matrix (rows sum to 1)

$\pi = \pi_1, \pi_2, \ldots, \pi_N$

Probabilities to be in states 1..N before the first step of the process

$O = o_1 o_2 \ldots o_T$

A sequence of **T** observations from the set $\mathbf{V = v_1,...v_v}$

$B = b_i(o_t)$

A matrix of emission probabilities: observation $\mathbf{o_t}$ from state $\mathbf{q_i}$

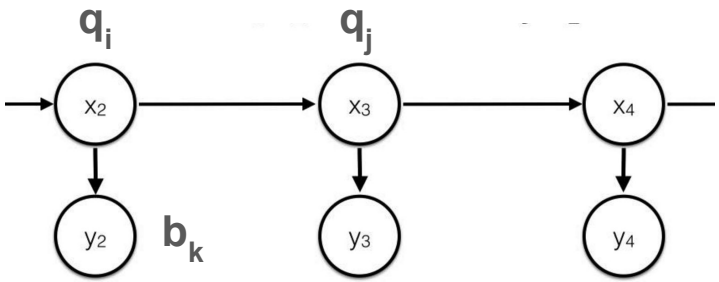
q_0, q_F

Initial and terminal states

Hidden Markov model

We assume that the object we are modeling can be approximated with a generative process:

1) throw the dice for the first state, let's say we get state **i**
2) for state **i** using **A** we generate observation $\mathbf{b_k}$
3) for state **i** we generate the next state **j**
4) i := j and goto (2) until we reach terminal state $\mathbf{q_F}$



Task 1: estimate the probability of the sequence

Let's say we have HMM(A, B).
What is the probability of O: **p(O|HMM(A,B))**?

Imagine we know the state sequence, then

$$P(O|Q) = \prod_{i=1}^{T} P(o_i|q_i)$$

but we don't know it, but we know the definition of the conditional probability

$$P(O,Q) = P(O|Q) \times P(Q) = \prod_{i=1}^{T} P(o_i|q_i) \times \prod_{i=1}^{T} P(q_i|q_{i-1})$$

and then

$$P(O) = \sum_{Q} P(O,Q) = \sum_{Q} P(O|Q)P(Q)$$

Task 1: estimate the probability of the sequence

Sad: traversing all possible state sequences ~ $\mathbf{O(N^T)}$

Good news: can be computed with dynamic programming ~ $\mathbf{O(N^2T)}$:
For every moment of time **t** we can estimate the probability of observation sequence $\mathbf{o_1...o_t}$, if we know the probability of the sequence $\mathbf{o_1...o_{t-1}}$

We can compute this

$$\alpha_t(j) = P(o_1, o_2 \ldots o_t, q_t = j|\lambda)$$

Based on all possible previous states

$$\alpha_t(j) = \sum_{i=1}^{N} \alpha_{t-1}(i) a_{ij} b_j(o_t)$$

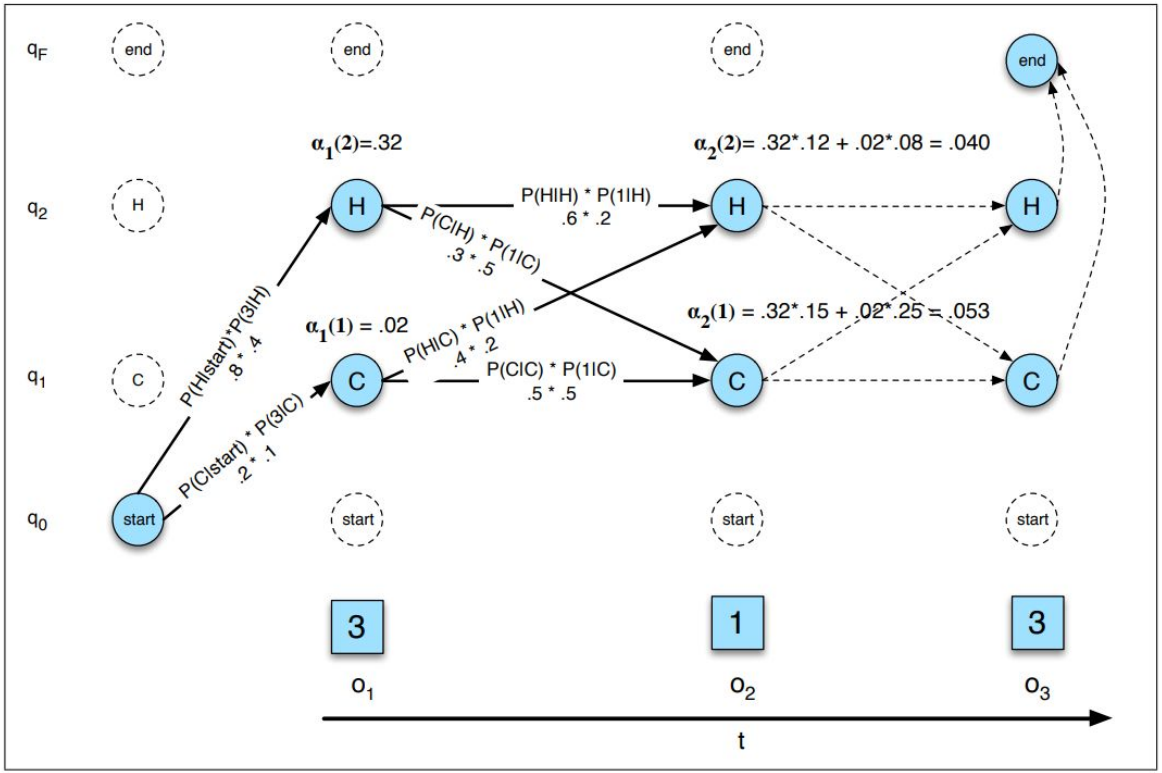
Task 1: estimate the probability of the sequence

Filling the matrix, the results are computed using the last row

$$\alpha_t(j) = \sum_{i=1}^{N} \alpha_{t-1}(i) a_{ij} b_j(o_t)$$

aka

the forward algorithm



Task 2: decoding

...which is the recovery of the most probable sequence of states given the sequence of observations and the model **HMM(A,B)**

Naive algorithm: iterate through all possible states sequences, estimate their probabilities given data $\mathbf{o_1,...,o_T}$ using the forward-algorithm

Why naive? We have all the methods, let's just compute it!

Task 2: decoding

...which is the recovery of the most probable sequence of states given the sequence of observations and the model **HMM(A,B)**

Naive algorithm: iterate through all possible states sequences, estimate their probabilities given data $\mathbf{o_1,...,o_T}$ using the forward-algorithm

...Which means combinatorial complexity!

As with forward algorithm, dynamic programming comes for help: let's build a matrix, where at step **t** the cell **j** will be filled with the probability of the process to be in the state **j after going through the most probable state sequence** $\mathbf{q_0, q_2, \ldots, q_{t-1}}$

Task 2: decoding, Viterbi algorithm

Andrew Viterby (b. 1935) - American engineer of Italian origin, co-founder of Qualcomm

Idea: for every step **t** and for every state **j** we recursively compute the probability of the process to be in the state **j** assuming we have come to it using the most probable 'states path' $\mathbf{q_0}$, $\mathbf{q_2}$, …, $\mathbf{q_{t-1}}$ given observations $\mathbf{o_1}$,...,$\mathbf{o_t}$

$$v_t(j) = \max_{q_0, q_1, \ldots, q_{t-1}} P(q_0, q_1 \ldots q_{t-1}, o_1, o_2 \ldots o_t, q_t = j | \lambda)$$

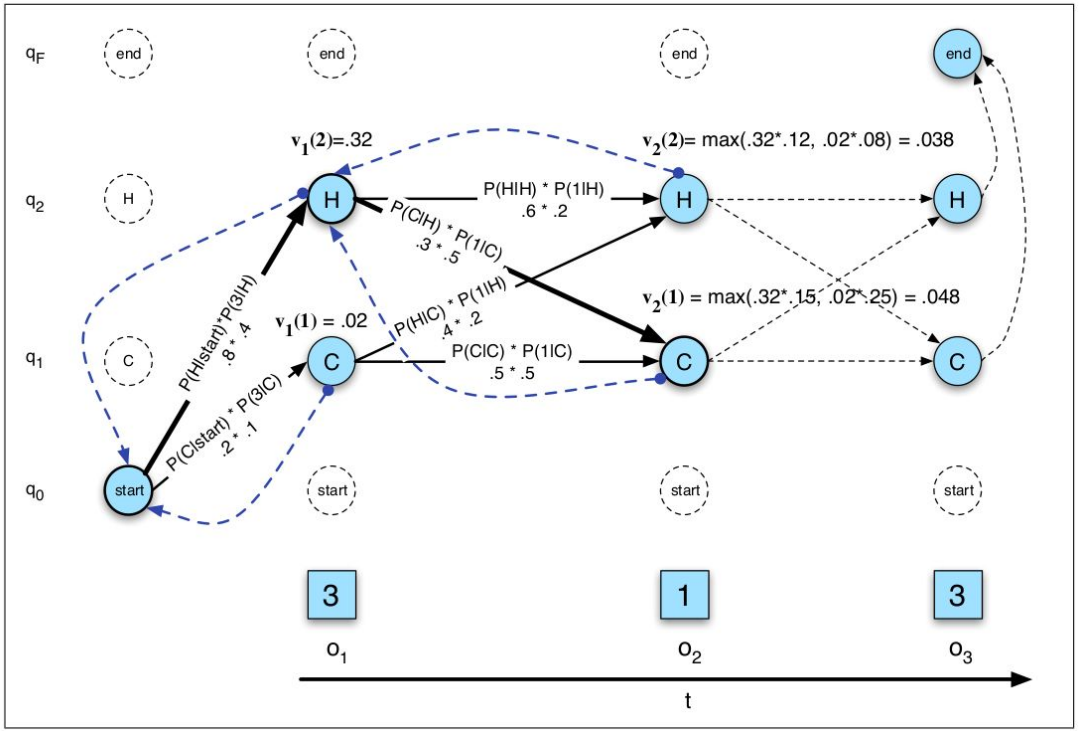
Which can be rewritten as

$$v_t(j) = \max_{i=1}^{N} v_{t-1}(i)\, a_{ij}\, b_j(o_t)$$

we take the probabilities for the previous step, multiply them by the transition-to-the-current-state probability and the current observation emission probability.

Can easily prove this approach is valid using mathematical induction: just write down $\mathbf{v_t(j)}$ and prove that this is the largest possible probability (chain rule + cond.independence + moving constant values)

Task 2: Viterbi decoding



Saving both maximums and argmaxes: to determine which states we've come from at each step **t**

Having estimated the probabilities on the last step, we take the cell with max probability, and go in the reverse direction using the backpointers (argmaxes)

Task 3: training

When we don’t have data annotated with tags completely, we can use **algorithm** (aka Forward-Backward aka **Baum-Welch algorithm**)

But POS, BIO-NER are given and set in our case

Hence we can simply estimate these probabilities as **count**s

$$P(t_i|t_{i-1}) = \frac{C(t_{i-1},t_i)}{C(t_{i-1})}$$

$$P(w_i|t_i) = \frac{C(t_i,w_i)}{C(t_i)}$$

$$P(VB|MD) = \frac{C(MD,VB)}{C(MD)} = \frac{10471}{13124} = .80$$

$$P(will|MD) = \frac{C(MD,will)}{C(MD)} = \frac{4046}{13124} = .31$$

Computed on Wall Street Journal corpus

Hidden Markov models

Once again, how it works:

1. Annotated corpus:
   “words” — observations
   tags — states
2. Estimation of conditional probabilities of transition and generation
3. [probabilities smoothing]
4. Running Viterbi algorithm on the incoming previously unseen sequence and getting tags

Hidden Markov models: discussion

Local Classifiers

- Form:

$$\mathbf{w} \cdot \mathbf{f}(\mathbf{x}, i, l)$$

- Learning: standard classifiers
- Prediction: independent for each x_i
- Advantage: feature-rich
- Drawback: no label interactions

HMM

- Form:

$$\pi_{y_1} O_{y_1,x_1} \prod_{i>1} T_{y_{i-1},y_i} O_{y_i,x_i}$$

- Learning: relative counts
- Prediction: Viterbi
- Advantage: label interactions
- Drawback: no fine-grained features

http://lxmls.it.pt/2015/strlearn.pdf

Hidden Markov models: discussion

1. A simple sequence modeling technique

2. Can easily be generalized if we want to take larger context into account (for NLP tasks such models are used, the **big-O(n^x)** gradually grows, **x** ~ the size of context)

3. For practical use, many more complex modifications were developed

Hidden Markov models are not useful when we want to take arbitrary contextual features, there are other richer models for that

Plan

1. ~~Motivation~~
2. ~~Approaches we will discuss~~
3. ~~Methods~~
    a. ~~Classifiers~~
    b. ~~Hidden Markov models~~
    c. Structured perceptron
    d. (a lil bit of) Conditional Random Fields
4. Tools and data

Linear factorized models

$$\hat{\mathbf{y}} = \underset{\mathbf{y} \in \mathcal{Y}^n}{\operatorname{argmax}} \sum_{i=1}^{n} \mathbf{w} \cdot \mathbf{f}(\mathbf{x}, i, y_{i-1}, y_i)$$

the goal is to find tags so that the sum of scalar products of weights and features would be max (when summing over all elements of the sequence)

Viterbi helps again: let's set

$$\delta(i, a) = \max_{\mathbf{y} \in \mathcal{Y}^i : y_i = a} \sum_{j=1}^{i} \mathbf{w} \cdot \mathbf{f}(\mathbf{x}, j, y_{j-1}, y_j)$$

then we shall simply do this

$$\begin{aligned} \delta(1, a) &= \mathbf{w} \cdot \mathbf{f}(\mathbf{x}, 1, y_0 = \text{NULL}, a) \\ \delta(i, a) &= \max_{b \in \mathcal{Y}} \delta(i-1, b) + \mathbf{w} \cdot \mathbf{f}(\mathbf{x}, i, b, a) \end{aligned}$$

http://lxmls.it.pt/2015/strlearn.pdf

Structured perceptron

A linear model; training: updating weights based on errors when predicting in online fashion -- as in one-layer perceptron

Set $\mathbf{w} = \mathbf{0}$

For $t = 1 \ldots T$

- For each training example $(\mathbf{x}, \mathbf{y})$
  1. Compute $\mathbf{z} = \text{argmax}_{\mathbf{z}} \, \mathbf{w} \cdot \mathbf{f}(\mathbf{x}, \mathbf{z})$
  2. If $\mathbf{z} \neq \mathbf{y}$

$$\mathbf{w} \leftarrow \mathbf{w} + \mathbf{f}(\mathbf{x}, \mathbf{y}) - \mathbf{f}(\mathbf{x}, \mathbf{z})$$

Return $\mathbf{w}$

Viterbi is hiding here

Updating the weights if predictions are wrong

Usually the updates are multiplied by the learning rate < 1

Michael Collins. 2002. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 1–8.

http://lxmls.it.pt/2015/strlearn.pdf

Averaged structured perceptron

When updates are averaged, the results are way better

Set $\mathbf{w} = \mathbf{0}$, $\mathbf{w}^a = \mathbf{0}$

For $t = 1 \dots T$

- For each training example $(\mathbf{x}, \mathbf{y})$
  1. Compute $\mathbf{z} = \text{argmax}_{\mathbf{z}}\, \mathbf{w} \cdot \mathbf{f}(\mathbf{x}, \mathbf{z})$
  2. If $\mathbf{z} \neq \mathbf{y}$

$$\mathbf{w} \leftarrow \mathbf{w} + \mathbf{f}(\mathbf{x}, \mathbf{y}) - \mathbf{f}(\mathbf{x}, \mathbf{z})$$

  3. $\mathbf{w^a} = \mathbf{w^a} + \mathbf{w}$

Идея усреднения параметров отсюда: Yoav Freund and Robert Schapire. 1999. Large margin classification using the perceptron algorithm. Machine Learning, 3(37):277–296.
Применение здесь же: Michael Collins. 2002. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 1–8.

Structured perceptron: discussion

- Arbitrary features that have no constraints that HMM features do have
- Is trained in online fashion, and convergence is fast (sometimes < 10 iterations)
- Averaged version works WAY BETTER than the standard one
- Shows results compatible to CRF and structured SVM

Plan

1. ~~Motivation~~
2. ~~Approaches we will discuss~~
3. ~~Methods~~
    a. ~~Classifiers~~
    b. ~~Hidden Markov models~~
    c. ~~Structured perceptron~~
    d. (a lil bit of) Conditional Random Fields
4. Tools and data

Conditional Random Fields (CRF)

1) can be called a ‘logistic regression for sequences’
2) ‘discriminative sibling’ of HMM

Let’s rewrite joint distribution of words and tags describing HMM

$$p(\mathbf{y}, \mathbf{x}) = \frac{1}{Z} \prod_{t=1}^{T} \exp \left\{ \sum_{i,j \in S} \theta_{ij} \mathbf{1}_{\{y_t = i\}} \mathbf{1}_{\{y_{t-1} = j\}} + \sum_{i \in S} \sum_{o \in O} \mu_{oi} \mathbf{1}_{\{y_t = i\}} \mathbf{1}_{\{x_t = o\}} \right\}$$

mu and **theta** are log-probabilities of transition and generation

Lafferty, J., McCallum, A., Pereira, F. (2001).
"Conditional random fields: Probabilistic models for segmenting and labeling sequence data". Proc. 18th International Conf. on Machine Learning. Morgan Kaufmann. pp. 282–289.

CRF

Generalization and rewriting:

$$p(\mathbf{y}, \mathbf{x}) = \frac{1}{Z} \prod_{t=1}^{T} \exp \left\{ \sum_{k=1}^{K} \theta_k f_k(y_t, y_{t-1}, x_t) \right\}$$

then

$$p(\mathbf{y}|\mathbf{x}) = \frac{p(\mathbf{y}, \mathbf{x})}{\sum_{\mathbf{y}'} p(\mathbf{y}', \mathbf{x})} = \frac{\prod_{t=1}^{T} \exp \left\{ \sum_{k=1}^{K} \theta_k f_k(y_t, y_{t-1}, x_t) \right\}}{\sum_{\mathbf{y}'} \prod_{t=1}^{T} \exp \left\{ \sum_{k=1}^{K} \theta_k f_k(y'_t, y'_{t-1}, x_t) \right\}}$$

CRF: definition

a **linear-chain** conditional random field is a distribution set as

$$p(\mathbf{y}|\mathbf{x}) = \frac{1}{Z(\mathbf{x})} \prod_{t=1}^{T} \exp\left\{ \sum_{k=1}^{K} \theta_k f_k(y_t, y_{t-1}, \mathbf{x}_t) \right\},$$

where $Z(\mathbf{x})$ is an instance-specific normalization function

$$Z(\mathbf{x}) = \sum_{\mathbf{y}} \prod_{t=1}^{T} \exp\left\{ \sum_{k=1}^{K} \theta_k f_k(y_t, y_{t-1}, \mathbf{x}_t) \right\}.$$

where **theta** are real-valued k-dimensional vectors of parameters
and **f** are feature functions

CRF: discussion

- popular in language processing as well as in bioinformatics, image analysis, etc.

- was designed as a (and actually is) a probabilistic graphical model

- as well as structured perceptron, CRF can ‘see’ the whole sequence for prediction, so one can easily set arbitrary features (using parts of the words, etc.)

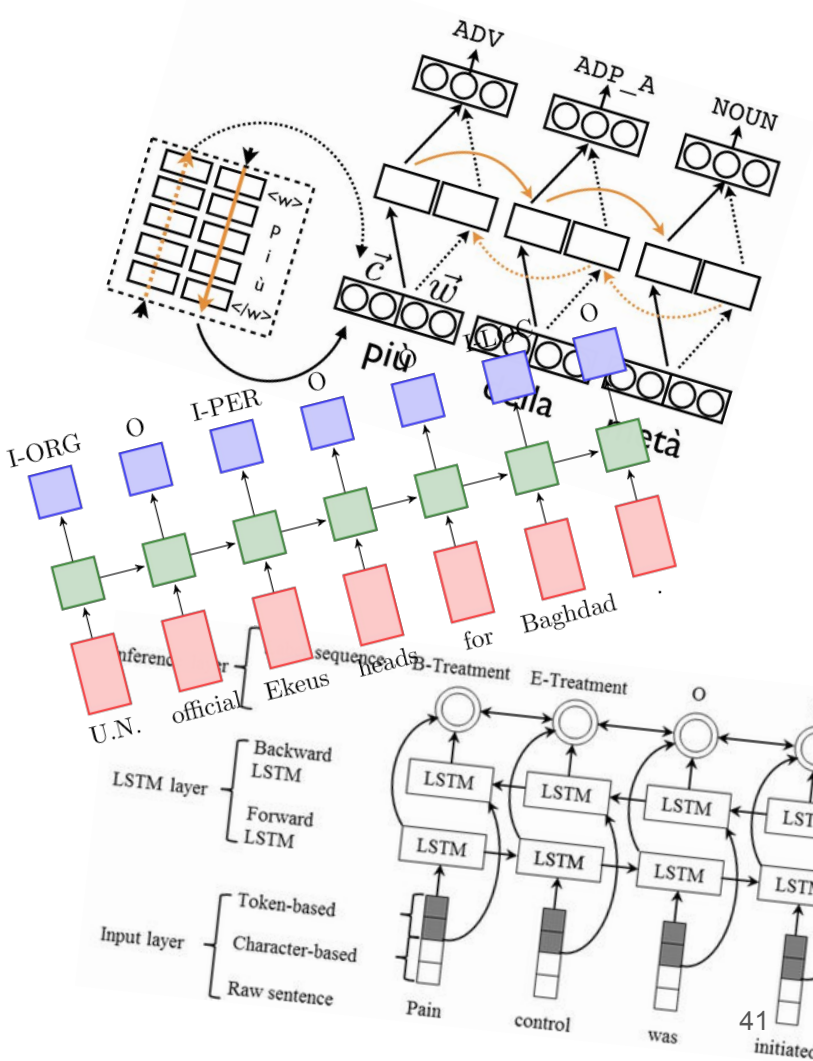
- has effective implementations and lots of extensions

What else?

- sequence learning — is certainly a task for neural architectures for sequential data!

  (current SOTA pretty much everywhere is close to the results achieved with **bi-LSTM-CRF**)

Stay tuned:
https://aclweb.org/aclwiki/State_of_the_art



Plan

1. ~~Motivation~~
2. ~~Approaches we will discuss~~
3. ~~Methods~~
    a. ~~Classifiers~~
    b. ~~Hidden Markov models~~
    c. ~~Structured perceptron~~
    d. ~~(a lil bit of) Conditional Random Fields~~
4. Tools and data

Data

- **POS-tagging**
    - Conference tasks tracks,
      e.g. **CoNLL-2000 Shared Task**
    - Annotated datasets for 40+ languages, the project called
      Universal Dependencies: http://universaldependencies.org/
- **NER**
    - NER data on Technion site:
      http://www.cs.technion.ac.il/~gabr/resources/data/ne_datasets.html

For different tasks for Russian language there are datasets on the conference Dialog site:
http://www.dialog-21.ru/en/evaluation/

Tools

- hmmlearn
  (unsupervised HMM, sklearn-like API)
- PyStruct
  (by one of sklearn major maintainers)
- CRF++ and CRFSuite
  (are said to be blazing fast; check if still maintained)
- seqlearn
  (seems to be abandoned by maintainers, though API is cool)
- MALLET
  (+GRMM)
- Alchemy is also an option

Used / recommended literature

1. Martin/Jurafsky, chapters 9-10, ed. 3
2. Rabiner's tutorial on HMM
3. Noah Smith's lecture slides on HMM etc. from LxMLS
4. Xavier Carreras lecture slides on structured prediction from LxMLS (hot! short description of the models, many links)
5. Introduction into CRF by Statton and McCallum (MALLET author)
6. Wikipedia

Tagging

(sequence labeling for NLP)

Anton Alekseev
Steklov Mathematical Institute in St Petersburg

ITMO University, St. Petersburg, 2019
anton.m.alexeyev+itmo@gmail.com

Thanks for helping me with the slides go to Denis Kiryanov