Natural Language Understanding

Lecture 11: Unsupervised Part-of-Speech Tagging with Neural Networks

Frank Keller

School of Informatics
University of Edinburgh
keller@inf.ed.ac.uk

March 3, 2017

1. Introduction
   - Hidden Markov Models
   - Extending HMMs
2. Maximum Entropy Models as Emissions
   - Estimation
   - Features
   - Results
3. Embeddings as Emissions
   - Embeddings
   - Estimation
   - Results

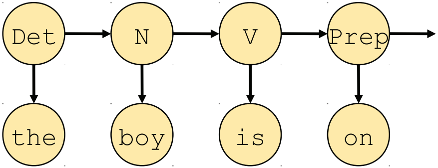
Reading: Berg-Kirkpatrick et al. (2010); Lin et al. (2015).
Background: Jurafsky and Martin (2009: Ch. 6.5).

Hidden Markov Models

Recall our notation for HMM from the last lecture:

tags $\mathbf{t} = t_1 \ldots t_n$

words $\mathbf{w} = w_1 \ldots w_n$



$$P(\mathbf{t}, \mathbf{w}) = \prod_{i=1}^{n} P(t_i | t_{i-1}) P(w_i | t_i)$$

The parameters of the HMM are $\theta = (\tau, \omega)$. They define:

- τ: the probability distribution over tag-tag transitions;
- ω: the probability distribution over word-tag outputs.

Hidden Markov Models

The model is based on a set of multinomial distributions. For tag types $t = 1 \ldots T$ and word types $w = 1 \ldots W$:

- $\omega = \omega^{(1)} \ldots \omega^{(T)}$: the output distributions for each tag;
- $\tau = \tau^{(1)} \ldots \tau^{(T)}$: the transition distributions for each tag;
- $\omega^{(t)} = \omega_1^{(t)} \ldots \omega_W^{(t)}$: the output distribution from tag t;
- $\tau^{(t)} = \tau_1^{(t)} \ldots \tau_T^{(t)}$: the transition distribution from tag t.

Goal of this lecture: *replace the output distributions ω with something cleverer than multinomials.*

Hidden Markov Models

Example: $\omega^{(\texttt{NN})}$ is the output distribution for tag `NN`:

w
John
Mary
running
jumping

[Source: Taylor Berg-Kirkpatrick et al: Painless Unsupervised Learning with Features, ACL slides 2010.]

Hidden Markov Models

Example: $\omega^{(\mathtt{NN})}$ is the output distribution for tag $\mathtt{NN}$:

$\omega_w^{(\mathtt{NN})}$	w
0.1	John
0.0	Mary
0.2	running
0.0	jumping

[Source: Taylor Berg-Kirkpatrick et al: Painless Unsupervised Learning with Features, ACL slides 2010.]

Hidden Markov Models

Example: $\omega^{(\mathtt{NN})}$ is the output distribution for tag $\mathtt{NN}$:

$\omega_w^{(\mathtt{NN})}$	w	$\mathbf{f}(\mathtt{NN}, w)$
0.1	John	+Cap
0.0	Mary	+Cap
0.2	running	+ing
0.0	jumping	+ing

[Source: Taylor Berg-Kirkpatrick et al: Painless Unsupervised Learning with Features, ACL slides 2010.]

Hidden Markov Models

Example: $\omega^{(\texttt{NN})}$ is the output distribution for tag NN:

$\omega_w^{(\texttt{NN})}$	w	$\mathbf{f}(\texttt{NN}, w)$	$e^{\boldsymbol{\lambda}\cdot\mathbf{f}(\texttt{NN},w)}$
0.1	John	+Cap	0.3
0.0	Mary	+Cap	0.3
0.2	running	+ing	0.1
0.0	jumping	+ing	0.1

First idea: use local features to define $\omega^{(t)}$ (Berg-Kirkpatrick et al. 2010):

$$\omega_w^{(t)} = \frac{\exp(\boldsymbol{\lambda} \cdot \mathbf{f}(t, w))}{\sum_{w'} \exp(\boldsymbol{\lambda} \cdot \mathbf{f}(t, w'))} \tag{1}$$

Multinomials become maximum entropy models.

[Source: Taylor Berg-Kirkpatrick et al: Painless Unsupervised Learning with Features, ACL slides 2010.]

Hidden Markov Models

Example: $\omega^{(\mathtt{NN})}$ is the output distribution for tag `NN`:

w
John
Mary
running
jumping

Hidden Markov Models

Example: $\omega^{(\mathtt{NN})}$ is the output distribution for tag $\mathtt{NN}$:

$\omega_w^{(\mathtt{NN})}$	w
0.1	John
0.0	Mary
0.2	running
0.0	jumping

Hidden Markov Models

Example: $\omega^{(\texttt{NN})}$ is the output distribution for tag $\texttt{NN}$:

$\omega_w^{(\texttt{NN})}$	w	$\mathbf{v}_w$
0.1	John	[0.1 0.4 0.06 1.7]
0.0	Mary	[0.2 1.3 0.20 0.0]
0.2	running	[3.1 0.4 0.06 1.7]
0.0	jumping	[0.7 0.4 0.02 0.5]

Hidden Markov Models

Example: $\omega^{(\mathtt{NN})}$ is the output distribution for tag $\mathtt{NN}$:

$\omega_w^{(\mathtt{NN})}$	w	$\mathbf{v}_w$	$p(\mathbf{v}_w; \boldsymbol{\mu}_t, \boldsymbol{\Sigma}_t)$
0.1	John	[0.1 0.4 0.06 1.7]	0.3
0.0	Mary	[0.2 1.3 0.20 0.0]	0.3
0.2	running	[3.1 0.4 0.06 1.7]	0.1
0.0	jumping	[0.7 0.4 0.02 0.5]	0.1

Second idea: use word embeddings to define $\omega^{(t)}$ (Lin et al. 2015):

$$\omega_w^{(t)} = \frac{\exp(-\frac{1}{2}(\mathbf{v}_w - \boldsymbol{\mu}_t)^\top \boldsymbol{\Sigma}_t^{-1}(\mathbf{v}_w - \boldsymbol{\mu}_t))}{\sqrt{(2\pi)^d |\boldsymbol{\Sigma}_t|}}$$

Multinomials become multivariate Gaussians with d dimensions.

Standard Expectation Maximization

For both ideas, we can use the *Expectation Maximization Algorithm* to estimate model parameters.

Standard EM optimizes $L(\theta) = \log P_\theta(\mathbf{w})$. The E-step computes the expected counts for the emissions:

$$e_{(t,w)} \leftarrow \mathbb{E}_\omega \left[\sum_i \mathbb{I}(t, w_i) \middle| \mathbf{w} \right] \quad (2)$$

The expected counts are then normalized in the M-step to re-estimate θ:

$$\omega_w^{(t)} \leftarrow \frac{e_{(t,w)}}{\sum_{w'} e_{(t,w')}} \quad (3)$$

The expected counts can be computed efficiently using the Forward-Backward algorithm (aka Baum-Welch algorithm).

Expectation Maximization for HMMs with Features

Now the E-step first computes $\omega_w^{(t)}$ given $\boldsymbol{\lambda}$ as in (1), then it computes the expectations as in (2) using Forward-Backward.

The M-step now optimizes the regularized expected log likelihood over all word-tag pairs:

$$\ell(\boldsymbol{\lambda}, \mathbf{e}) = \sum_{(t,w)} e_{(t,w)} \log \omega_w^{(t)}(\boldsymbol{\lambda}) - \kappa ||\boldsymbol{\lambda}||_2^2$$

To compute $\ell(\boldsymbol{\lambda}, \mathbf{e})$, we use a general gradient-based search algorithm, e.g., the LBFGS (Limited-memory Broyden-Fletcher-Goldfarb-Shanno) algorithm.

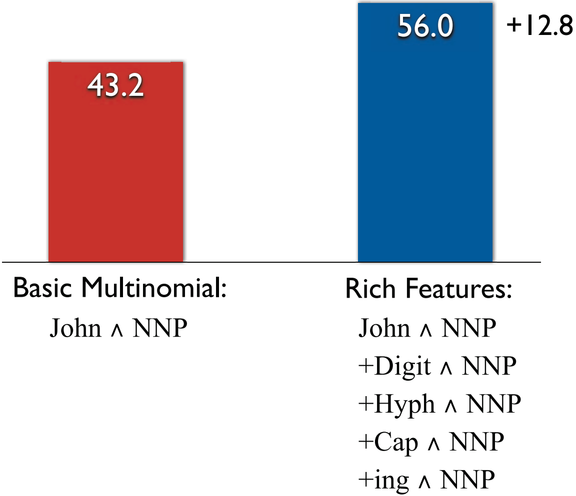
HMMs with Features

The key advantage of Berg-Kirkpatrick et al.'s (2010) approach is that we can now add arbitrary features to the HMM:

BASIC:	$\mathbb{I}(w = \cdot, t = \cdot)$
CONTAINS-DIGIT:	Check if w contains digit and conjoin with t: $\mathbb{I}(containsDigit(w) = \cdot, t = \cdot)$
CONTAINS-HYPHEN:	$\mathbb{I}(containsHyphen(w) = \cdot, t = \cdot)$
INITIAL-CAP:	Check if the first letter of w is capitalized: $\mathbb{I}(isCap(w) = \cdot, t = \cdot)$
N-GRAM:	Indicator functions for character n-grams of up to length 3 present in w.

A standard HMM only has the BASIC features. ($\mathbb{I}$ is the indicator function; returns 1 if the features is present, 0 otherwise.)

Results



[Source: Taylor Berg-Kirkpatrick et al: Painless Unsupervised Learning with Features, ACL slides 2010.]

Embeddings as Multivariate Gaussians

Given a tag t, instead of a word w, we generate a pretrained embedding $\mathbf{v}_w \in \mathbb{R}^d$ (d dimensionality of the embedding).

We assume that $\mathbf{v}_w$ is distributed according to a multivariate Gaussian with the mean vector $\boldsymbol{\mu}_t$ and covariance matrix $\boldsymbol{\Sigma}_t$:

$$\omega_w^{(t)} = p(\mathbf{v}_w; \boldsymbol{\mu}_t, \boldsymbol{\Sigma}_t) = \frac{\exp(-\frac{1}{2}(\mathbf{v}_w - \boldsymbol{\mu}_t)^\top \boldsymbol{\Sigma}_t^{-1}(\mathbf{v}_w - \boldsymbol{\mu}_t))}{\sqrt{(2\pi)^d |\boldsymbol{\Sigma}_t|}}$$

This means we assume that the embeddings of words which are often tagged as t are concentrated around the point $\boldsymbol{\mu}_t$, where the concentration decays according to $\boldsymbol{\Sigma}_t$.

Embeddings as Multivariate Gaussians

Now, the joint distribution over a sequence of words $\mathbf{w} = w_1 \ldots w_n$ is represented as a sequence of vectors $\mathbf{v} = v_{w_1} \ldots v_{w_n}$. The joint probability of a word and tag sequence is:

$$P(\mathbf{t}, \mathbf{w}) = \prod_{i=1}^{n} P(t_i | t_{i-1}) p(\mathbf{v}_w; \boldsymbol{\mu}_t, \boldsymbol{\Sigma}_t)$$

We again estimate the parameters $\boldsymbol{\mu}_t$ and $\boldsymbol{\Sigma}_t$ using Forward-Backward.

EM for HMMs with Embeddings

In each EM iteration, we update $\boldsymbol{\mu}_{t^*}$:

$$\boldsymbol{\mu}_{t^*}^{new} = \frac{\sum_{\mathbf{v} \in \mathcal{T}} \sum_{i=1 \dots n} p(t_i = t^* | \mathbf{v}) \cdot \mathbf{v}_{w_i}}{\sum_{\mathbf{v} \in \mathcal{T}} \sum_{i=1 \dots n} p(t_i = t^* | \mathbf{v})}$$

where $\mathcal{T}$ is a data set of word embedding sequences $\mathbf{v}$, each of length $|\mathbf{v}| = n$, and $p(t_i = t^* | \mathbf{v})$ is the posterior probability of label t^* at position i in the sequence $\mathbf{v}$.

EM for HMMs with Embeddings

In each EM iteration, we update $\mathbf{\Sigma}_{t^*}$:

$$\mathbf{\Sigma}_{t^*}^{new} = \frac{\sum_{\mathbf{v} \in \mathcal{T}} \sum_{i=1\ldots n} p(t_i = t^* | \mathbf{v}) \cdot \boldsymbol{\delta}\boldsymbol{\delta}^\top}{\sum_{\mathbf{v} \in \mathcal{T}} \sum_{i=1\ldots n} p(t_i = t^* | \mathbf{v})}$$

where $\boldsymbol{\delta} = \mathbf{v}_{w_i} - \boldsymbol{\mu}_{t^*}^{new}$.

Model Comparison

Compare related models for unsupervised PoS tagging:

- HMM with multinomial emissions;
- HMM with MaxEnt emissions (Berg-Kirkpatrick et al. 2010);
- conditional random field (CRF) autoencoder with multinomial reconstructions;
- HMM with Gaussian emissions;
- CRF autoencoder with Gaussian reconstructions.

Note: CRF models will not be covered in this course.

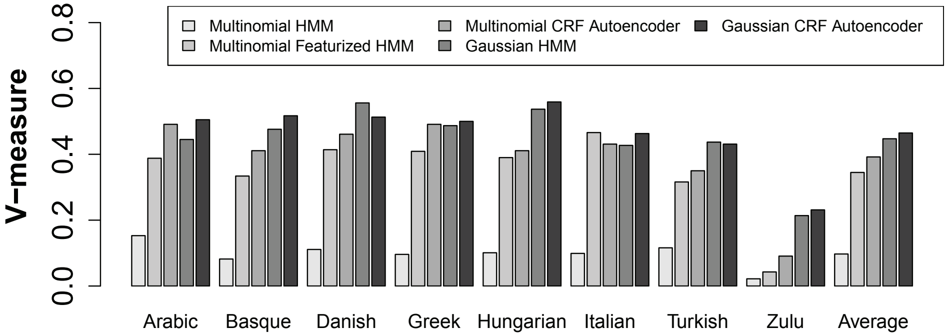
Setup

- Train models on CoNLL shared task data for eight languages;
- for evaluation, map language-specific gold-standard tag sets onto universal PoS tags;
- use skip-gram embeddings with window size 1 and $d = 100$;
- train embeddings on largest corpus available for each language;
- estimate $\boldsymbol{\mu}_t$ as above;
- estimating $\boldsymbol{\Sigma}_t$ did not lead to improvement; assume fixed, diagonal co-variance matrix;
- HMM parameters initialized randomly;
- tune hyperparameters on English PTB and then keep fixed;
- evaluate using V-measure.

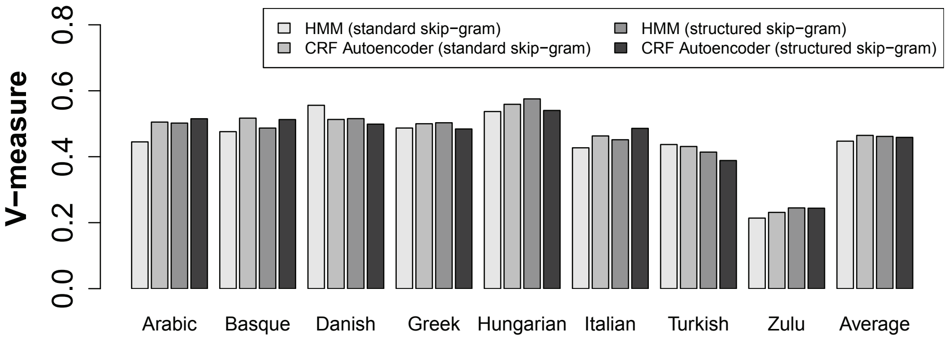
Universal PoS Tagset

ADJ	adjective
ADP	adposition
ADV	adverb
AUX	auxiliary verb
CONJ	coordinating conjunction
DET	determiner
INTJ	interjection
NOUN	noun
NUM	numeral
PART	particle
PRON	pronoun
PROPN	proper noun
PUNCT	punctuation
SCONJ	subordinating conjunction
SYM	symbol
VERB	verb
X	other

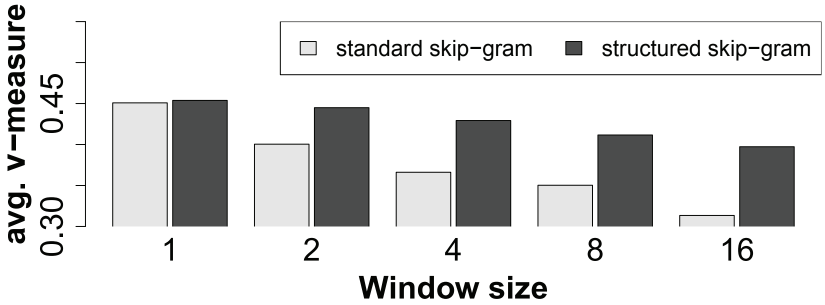
Results: Effect of Model



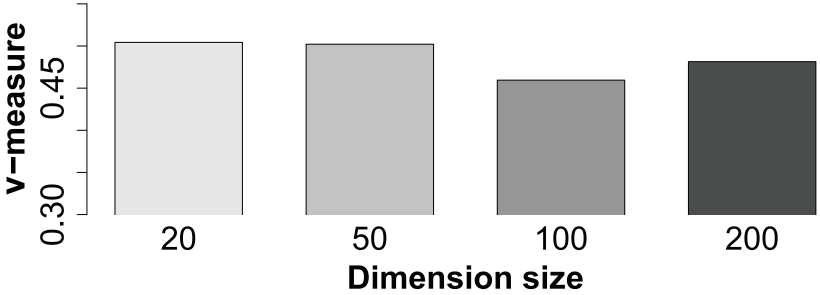
Results: Standard Skip-gram vs. Structured Skip-gram



Results: Window Size



Results: Dimensionality of Embeddings



Summary

- Word embeddings improve unsupervised PoS tagging;
- Gaussian HMM outperforms MaxEnt HMM and CRF autoencoder;
- Gaussian CRF autoencoder similar to Gaussian HMM;
- but: models with embeddings use a lot more training data;
- structured skip-gram slightly outperform skip-gram;
- embeddings with $d = 20$ outperform embeddings with high dimensionality;
- window size of 1 is optimal.

References

Berg-Kirkpatrick, Taylor, Alexandre Bouchard-Côté, John DeNero, and Dan Klein. 2010. Painless unsupervised learning with features. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*. Uppsala, pages 582–590.

Jurafsky, Daniel and James H. Martin. 2009. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics and Speech Recognition*. Pearson Education, Upper Saddle River, NJ, 2 edition.

Lin, Chu-Cheng, Waleed Ammar, Chris Dyer, and Lori Levin. 2015. Unsupervised POS induction with word embeddings. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*. Denver, CO, pages 1311–1316.