CMPT-413
Computational Linguistics

Anoop Sarkar
http://www.cs.sfu.ca/∼anoop

March 8, 2011

Human Supervision in Part of Speech Tagging

- In unseen data, we wish to find the part of speech tags:
  Input: *In 1994 , Hartnett said*
  Output: *In*_`IN` *1994*_`CD` ,_, *Hartnett*_`NNP` *said*_`VBD`
- The set of part of speech tags are decided by experts
- The experts also have to provide adequate amounts of data in which the part of speech tags have been listed for each word in context.
- This general approach is called **supervised learning** since the training data is provided by humans.

Trigram Models for Part of Speech Tagging

```
THE/DT BONEYARD/NNP Northrop/NNP  Grumman/NNP 's/POS modest/JJ
flight/NN museum/NN occupies/VBZ a/DT corner/NN of/IN one/CD of/IN
its/PRP$ power-seat/NN adjusters/NNS ,/, door/NN trim/JJ now/RB
made/VBN in/IN South/NNP Korea/NNP 's/POS antiquated/JJ coal-fired/JJ
power/NN plant/NN in/IN Canada/NNP ,/, to/TO a/DT 11.9/CD million/CD
mark/NN investment/NN in/IN Samsung/NNP 's/POS Sachon/NNP plant/NN
in/IN Taiwan/NNP as/IN part/NN of/IN a/DT steam/NN turbine/NN ,/,
a/DT new/JJ high-yielding/JJ rice/NN plant/NN was/VBD reorganized/VBN
into/IN a/DT big/JJ expansion/NN of/IN a/DT fuel-fabrication/NN
plant/NN near/IN Nagoya/NNP in/IN Aichi/NNP Prefecture/NNP
```

Borges on Tagsets

```
Borges gives a vague reference to some work by Franz Kuhn
allegedly commenting on the classification of animals by a
Chinese encyclopedia called the _Celestial Emporium of
Benevolent Knowledge_.

>> ... animals are divided into:
(a) those that belong to the Emperor,
(b) embalmed ones,
(c) those that are trained,
(d) suckling pigs,
(e) mermaids,
(f) fabulous ones,
(g) stray dogs,
(h) those that are included in this classification,
(i) those that tremble as if they were mad,
(j) innumerable ones,
(k) those drawn with a very fine camel brush,
(l) others,
(m) those that have just broken a flower vase,
(n) those that resemble flies from a distance. <<
-- Jorge Luis Borges, "Other Inquisitions"
```

Part of Speech Tagging using Trigram Models

- Let the input sentence (word sequence) be $w_0, w_1, \ldots, w_n$
- Let the most likely tag sequence be $T^* = t_0^*, t_1^*, \ldots, t_n^*$
- In order to compare all possible tag sequences we build a probability model:

$$P(t_0, t_1, \ldots, t_n \mid w_0, w_1, \ldots, w_n)$$

Part of Speech Tagging using Trigram Models

- The best (or most likely) tag sequence is:

$$T^* = \underset{t_0, \ldots, t_n}{\text{arg max}} \; P(t_0, \ldots, t_n \mid w_0, \ldots, w_n)$$

$$\begin{aligned} & P(t_0, \ldots, t_n \mid w_0, \ldots, w_n) \\ = \quad & \frac{P(w_0, \ldots, w_n \mid t_0, \ldots, t_n) \times P(t_0, \ldots, t_n)}{P(w_0, \ldots, w_n)} \text{(Bayes Rule)} \\ = \quad & \color{blue}{P(w_0, \ldots, w_n \mid t_0, \ldots, t_n)} \times \color{red}{P(t_0, \ldots, t_n)} \end{aligned}$$

Part of Speech Tagging using Trigram Models

$$
\begin{aligned}
& P(w_0, \ldots, w_n \mid t_0, \ldots, t_n) \\
& \quad = \quad P(w_0 \mid t_0) \times P(w_1 \mid t_1) \times \ldots \times P(w_n \mid t_n) \\
& \quad = \quad \prod_{i=0}^{n} P(w_i \mid t_i)
\end{aligned}
$$

$$
\begin{aligned}
& P(t_0, \ldots, t_n) \\
& \quad = \quad P(t_0) \times P(t_1 \mid t_0) \times P(t_2 \mid t_0, t_1) \times \ldots \times P(t_n \mid t_{n-2}, t_{n-1}) \\
& \quad = \quad P(t_0) \times P(t_1 \mid t_0) \times \prod_{i=2}^{n} P(t_i \mid t_{i-2}, t_{i-1})
\end{aligned}
$$

Part of Speech Tagging using Trigram Models

$$
\begin{aligned}
& P(t_0, \ldots, t_n \mid w_0, \ldots, w_n) \\
&= P(w_0, \ldots, w_n \mid t_0, \ldots, t_n) \times P(t_0, \ldots, t_n) \\
&= \left(\prod_{i=0}^{n} P(w_i \mid t_i) \right) \times \left(P(t_0) \times P(t_1 \mid t_0) \times \prod_{i=2}^{n} P(t_i \mid t_{i-2}, t_{i-1}) \right) \\
&= \prod_{i=0}^{n} P(w_i \mid t_i) \times P(t_i \mid t_{i-2}, t_{i-1})
\end{aligned}
$$

Part of Speech Tagging using Bigram Models

$$P(t_0, \ldots, t_n \mid w_0, \ldots, w_n) \;=\; \prod_{i=0}^{n} P(w_i \mid t_i) \times P(t_i \mid t_{i-1})$$

- This allows us to represent tagging as a Hidden Markov Model (*hmm*).
- Each state in the *hmm* is a tag t_i
- The advantage is that we can reuse efficient *hmm* algorithms like Viterbi to find the most likely tag sequence for a given word sequence.
- However, instead of using Forward-Backward to find the values of $P(w_i \mid t_i)$ and $P(t_i \mid t_{i-1})$ we directly use frequencies from human labelled training data

Part of Speech Tagging using Trigram Models

$$P(t_0, \ldots, t_n \mid w_0, \ldots, w_n) = \prod_{i=0}^{n} P(w_i \mid t_i) \times P(t_i \mid t_{i-2}, t_{i-1})$$

- We can construct a *hmm* that is equivalent to the above model. Exactly the same construction as equivalence of Markov chains with *n*-gram models.
  - Except instead of pairs of words we have pairs of tags as states in the Markov chain.
  - And we add the emission probability to each state to extend the Markov chain to a *hmm*.

Part of Speech Tagging using Trigram Models

$$P(t_0, \ldots, t_n \mid w_0, \ldots, w_n) \;=\; \prod_{i=0}^{n} P(w_i \mid t_i) \times P(t_i \mid t_{i-2}, t_{i-1})$$

- Each state in the *hmm* is of the form $\langle t_j, t_k \rangle$ where i, j vary over all tags. Number of states is $|T|^2$ for a tag set T.
- Each transition from $\langle t_{i-2}, t_{i-1} \rangle$ to $\langle t_{i-1}, t_i \rangle$ occurs with transition probability $P(t_i \mid t_{i-2}, t_{i-1})$
- Each state $\langle t_{i-1}, t_i \rangle$ emits word w_i with emission probability $P(w_i \mid t_i)$

Part of Speech Tagging using Trigram Models

- So, all we need to do to find the most likely tag sequence is to *train* the following two probability models:

$$P(w_i \mid t_i) \text{ and } P(t_i \mid t_{i-2}, t_{i-1})$$

- Easy to do if we have **training data** with word and tag sequences.
- All we need after we have the probability models is an algorithm to find the most likely tag sequence
- Use the algorithm used to find the best tag sequence in Hidden Markov Models: the *Viterbi* algorithm

Part of Speech Tagging using Trigram Models

- **Evaluation**: *train* your model on the training data, *test* on unseen test data to obtain best tag sequence for each word sequence.
- **Accuracy** is measured as the percentage of correct tags for words in the test data.

Brief History of Part of Speech Tagging

- Corpus building: English
  - Brown Corpus: 1979 (87 tags)
  - Penn Treebank Corpus: 1993 (45 tags)
  - British National Corpus (BNC): 1997
  - LOB corpus
- Other languages: Chinese, Czech, German, Korean, Turkish, ...

Brief History of Part of Speech Tagging

- Models and Algorithms:
  - ngram models for tagging: Church 1988
  - extension of ngram model using HMMs: Xerox (Cutting et al) 1992
  - Transformation-Based Learning: Brill 1995
  - Maximum Entropy Models: Ratnaparkhi 1997
  - Reranking with Voted Perceptron: Collins 2002
  - Conditional Random Fields: Sha and Pereira, 2003
  - Improved MaxEnt Models: Toutanova et. al. 2003
  - Remove strict left to right sequence for tagging: Shen et. al. 2007
    - Current best error rate for English is 2.67% per tag

Applications of Part of Speech Tagging

- Other applications in NLP can be represented as POS tagging:
  - Chunking
  - Named-entity recognition (name-finding)
  - Cascaded Chunking
  - Word segmentation

Standard Part of Speech Tagging

- Part of speech tagging: finding the best sequence of POS tags for an input sentence (word sequence)
  - Representation: what does each POS tag represent?
  - Tagset: standard POS tags (NN=noun, VB=verb, etc.)
  - Training: word sequences with corresponding tag sequences
  - Input: word sequences (sentence)
  - Output: tag sequence

Noun Phrase Chunking

- Noun phrase chunking: e.g. input: *Brunetti gave the widow the news .*
  output: [Brunetti] gave [the widow] [the news] .
  - Representation: is each word inside an NP or not?
  - Tagset: 3 tags: **I** (inside NP), **O** (outside NP), **B** (boundary of 2 NPs) e.g. *Brunetti*/**I** *gave*/**O** *the*/**I** *widow*/**I** *the*/**B** *news*/**I** *.*
  - Training: word sequences with chunk tag sequences
  - Input: word sequences (sentence)
  - Output: chunk sequence
- A major limitation: chunking only permits non-recursive phrases.
- Finite set of tags do not permit recursive noun phrases, e.g.
  [[the police officer] from Venice] gave [the widow] [the news]
- There is no finite tagset that can capture arbitrary opening or closing brackets.
  e.g. $[_1[_2\ldots[_n$ assigned to tag B_n – not a finite tagset

Noun Phrase Chunking

- Noun phrase chunking: *Brunetti/***I** *gave/***O** *the/***I** *widow/***I** *the/***B** *news/***I** *.*
  - Tagset: Different options for the tags, as long as they correspond to the bracketing: [Brunetti] gave [the widow] [the news] .
  - For example, another representation could be: **I** (inside NP), **O** (outside NP), **E** (end of NP)
    e.g. *Brunetti/***E** *gave/***O** *the/***I** *widow/***E** *the/***I** *news/***E** *.*
  - If training data is in one representation, then we can transform from one tagset to another
- What about other kinds of phrases?

General Chunking

- Intuition for Noun Phrase chunking: In the sentence

  The company with the highest gain yesterday collapsed in today's market

  The relationship between the verb *collapsed* is to the entire phrase *The company with the highest gain yesterday*
- Similar intuition about other phrases, like prepositional phrases: *in today's market*

General Chunking

- General chunking is non-overlapping:
  e.g. input: *The company with the highest gain yesterday collapsed in today's market*,
  output: [**B-NP** The company] [**B-PP** with] [**B-NP** the highest gain] [**B-NP** yesterday] [**B-VP** collapsed] [**B-PP** in] [**B-NP** today's market]
  - Representation: is each word inside a chunk or not?
  - Tagset: **O** tag for outside chunk, **B-** or **E-** prefix to the types of chunks we want, for instance **NP, VP, PP**
    e.g. *The*/**B-NP** *company*/**E-NP** *with*/**B-PP** *the*/**B-NP** *highest*/**B-NP** *gain*/**E-NP** *yesterday*/**B-NP** *collapsed*/**B-VP** *in*/**B-PP** *today's*/**B-NP** *market*/**B-NP**

General Chunking

- General chunking is non-overlapping
  - Representation: is each word inside a chunk or not?
  - Tagset: **O** tag for outside chunk, **B**- or **E**- prefix to the types of chunks we want, **NP, VP, PP**
  - Training: word sequences with corresponding chunk tag sequences
  - Input: word sequences (sentence)
  - Output: chunk sequence

Named Entity Recognition

- In the sentence

  Mr. Vinken is chairman of Elsevier N. V. , a publishing group based in the Netherlands .

- We want to find names, such as person names, corporation names of locations:

  [PER Mr. Vinken] is chairman of [ORG Elsevier N. V.] , a publishing group based in the [LOC Netherlands] .

Named Entity Recognition

- A *named entity* is a chunk that contains only names of persons, organizations or locations
  - Representation: a word or group of words as a named entity
  - Tagset: **O** tag for outside any named entity, **B**- or **E**- prefix to the types of named entities we want: **PER** $=$ **person, LOC** $=$ **location, ORG** $=$ **organization**
  - Training: word sequences with corresponding named-entity tag sequences
  - Input: word sequences (sentence)
    Output: named-entity tag sequence

Cascaded Chunking

Input:	Mr.	Vinken	is	chairman	of	Elsevier	N.	V.
POS:	NNP	NNP	VBZ	NN	IN	NNP	NNP	NNP
NP:	I-NP	E-NP		I-NP		I-NP	I-NP	I-NP
PP:					I-PP	I-PP	I-PP	I-PP
VP:			I-VP	I-VP	I-VP	I-VP	I-VP	I-VP
S:	I-S	I-S	I-S	I-S	I-S	I-S	I-S	I-S

Cascaded Chunking

- A sequence of tagging steps
- Each step adds some more information
- Chunking had the disadvantage of not having overlapping chunks, cascaded chunking does not have this problem
  However, later steps cannot fix errors in earlier steps. For instance, a part of speech tagging error can cause errors in every successive step of cascaded chunking
- Later we will look at trees which generalize cascaded chunking in a principled way.

Summary: Part of Speech (POS) Tagging

- POS tagging is very similar to Hidden Markov Models
- POS tagging models are different from HMMs in the following ways:
  - The state sequences correspond to a particular representation (e.g. for trigram tagging each state in the *hmm* is a pair of tags)
  - The training data always has to contain the right tag for each word in the word (or observation) sequence (for supervised learning)
- Viterbi algorithm provides the best sequence of tags for a given input

  Part of speech tagging can be applied to many applications like chunking, name finding, among others