Lecture 5: Viterbi Walkthrough, HW1 Walkthrough, MEMMs.

Cornell Bowers C·IS
Computer Science

Claire Cardie, Tanya Goyal

CS 4740 (and crosslists): Introduction to Natural Language Processing

Announcements

- HW1 released.
  - HW1 milestone due on 12 September, 11.59 p.m.
  - HW1 due on 21 September, 11.59 p.m.

Today

- Viterbi walkthrough
- HW1 walkthrough
- HMMs vs MEMMs

HW1-programming walkthrough

- Task: NER using HMMs
- Dataset:
  - Train/val/test splits
    - text_i = ['Alice', 'and', 'Bob', 'walk', 'in', 'Paris']
    - NER_i = ['B-PER', 'O', 'B-PER', 'O', 'O', ‘B-LOC’]

Learn an HMM on the train data

Implement Viterbi

Use learnt HMM + Viterbi to predict NER tags on val and test

The ipynb walks you through these, simply follow that!

HW1-programming walkthrough

Learn an HMM on the train data

- text_i = ['alice', 'and', 'bob', 'walk', 'in', 'Paris']
- NER_i = ['B-PER', 'O', 'B-PER', 'O', 'O', 'B-LOC']

Implementation: use logprobs

- You will be computing probabilities (e.g. transition probabilities, emission probabilities)
- These numbers can be very small.
- Instead of multiplying probabilities $p_1 \times p_2 \times p_3 \ldots \times p_n$, ***work in the log space!***
  - $\log(p_1 \times p_2 \ldots \times p_n) = \log p_1 + \log p_2 + \ldots \log p_n$
- Avoids numerical overflow
- Can convert it back to a probability at the end if needed by taking the exp of the logprob.

HW1-programming walkthrough

Learn an HMM on the train data

- text_i = ['alice', 'and', 'bob', 'walk', 'in', 'Paris']
- NER_i = ['B-PER', 'O', 'B-PER', 'O', 'O', 'B-LOC']

Implementation choice: new/unknown words

- Your vocabulary is predetermined by the words you see in the training data (|V| = 6 when considering the above as the corpus)
  - Emission matrix will be of size (#tags, |V|)
- Suppose one document in the validation/test data has text = ["Bobby", "in", "Paris"]
  - What do we do? There are no cells corresponding to P("Bobby" | tag) !
- Typical technique: Pre-process your training data to replace low frequency words with <UNK>
  - e.g. suppose "Bob" and "Paris" are low frequency words (frequency computed over the whole dataset!)
  - ['Alice', 'and', '<UNK>', 'walk', 'in', '<UNK>'] – replace with <UNK> and add <UNK> to the vocabulary.
  - During prediction, any unseen work (e.g. "Bobby") can be similarly mapped to <UNK>.

HW1-programming walkthrough

Learn an HMM on the train data

- text_i = ['alice', 'and', 'bob', 'walk', 'in', 'Paris']
- NER_i = ['B-PER', 'O', 'B-PER', 'O', 'O', 'B-LOC']

Implementation choice: smoothing

- An unseen event isn't necessarily impossible! Safer to have all probs be non-zero.
- E.g. P('bob' | tag=O) = 0
- Consider hypothetical text sentence: ['I', 'bob', 'my', 'head', ….]. P('bob' | tag=O) needs to be non-zero for us to have any hope of assigning it the 'O' tag.

- Smoothing technique we will implement: Add-k smoothing.

HW1-programming walkthrough

Learn an HMM on the train data

- text_i = ['alice', 'and', 'bob', 'walk', 'in', 'Paris']
- NER_i = ['B-PER', 'O', 'B-PER', 'O', 'O', 'B-LOC']

- **Implementation choice: storing transition / emission probabilities.**

	few	mid	lot
H	0.1	0.3	0.6
C	0.5	0.4	0.1

- **Option 1: store as a matrix E**
  - Assign and store an index for each tag (e.g. $H \rightarrow 0, C \rightarrow 1$) and word (few -> 0, mid -> 1, lot -> 2)
  - Then E[0,1] corresponds to (H, mid), etc.
- **Option 2: Dictionary E**
  - Store values in a dictionary with keys (tag, word)
  - Get corresponding values via E[(tag, word)]

HW1-programming walkthrough

Use learnt HMM + Viterbi to predict NER tags on val and test

- Suppose the text of your val/test data is [['bobby', 'went', 'for', 'a', 'walk'], ['I', 'went', 'to', 'Paris']].
- Iterate through these documents:
  - For each, call ***viterbi(hmm, obs, tags)***

Set of possible tags (9 for the assignment)

Trained HMM model

current document tokens

HMM vs MEMMS

- HMMs: Find the tag sequence:

$$\arg\max_{t_1 \ldots t_N} P(t_1 \ldots t_N \mid o_{1:N}) = \arg\max_{t_1 \ldots t_N} \prod_i P(o_i \mid t_i) \times P(t_i \mid t_{i-1})$$

- MEMMs (Max Entropy Markov Models) assumptions:
  - Tag is independent of all other tags *except* the previous one.
  - But it can depend on the entire observation!

$$\arg\max_{t_1 \ldots t_N} P(t_1 \ldots t_N \mid o_{1:N}) = \arg\max_{t_1 \ldots t_N} \prod_i P_{\text{MEMM}}(t_i \mid t_{i-1}, o_{1:N})$$

Why condition on the whole input?

$$\arg\max_{t_1 \dots t_N} P(t_1 \dots t_N | o_{1:N}) = \arg\max_{t_1 \dots t_N} \prod_i P_{\text{MEMM}}(t_i | t_{i-1}, o_{1:N})$$

<s>/**START** I/**PP** am/**VBP** sitting/**VBG** in/**IN** Mindy/**NNP** 's/**POS** restaurant/**NN** eating/**VBG** the/**DT** gefilte/**NN** fish/**NN** ./**PERIOD** </s>/**END**

- Human analysts condition on the whole observation or sequence!
  - "Token ends in \`ing' –> Likely a verb. I can make this guess even if I have never seen this token before"
  - "Starts with a capital letter and not at the sentence start –> Could be a proper noun (but not if it is \`I')"
  - "Token is really long (lots of letters)? –> Probably not a preposition."

Features

- "Token ends in `ing' –> Likely a verb."
- "Token is really long (lots of letters)? –> Probably <u>not</u> a preposition."

Length of the token?

Does the token end in a "ing"? 0/1

- For a possible tag, some "*features*" of a token raise the chance of that tag and some <u>lower</u> it.
  - We should combine information components of the form:

Function that produces counts of occurrence of the "feature"

... multiplied by ...

a weight indicating how much positive/<u>negative</u> evidence the presence of that feature gives to the tag.

Formalizing features and evidence weights

- "Token ends in `ing' –> Likely a verb." Does the token end in a "ing"? 0/1
- "Token is really long (lots of letters)? –> Probably <u>not</u> a preposition." Length of the token?

$$P_{\text{MEMM}}(t_i \mid \boxed{t_{i-1}, o_{1:N}})$$

Extract features from these.

$f_1(t_i, t_{i-1}, o_{1:N}, i)$ = 1 if o_i ends in "ing".

The weight of this feature is say **3** for tag VERB.

The weight of this feature is **-1** for prepositions

$f_2(t_i, t_{i-1}, o_{1:N}, i)$ = length of o_i

The weight of this feature is **0** for VERB

The weight of this feature is **-2** for prepositions

Formalizing features and evidence weights

- **How do we compute** $P_{\text{MEMM}}(t_i \mid \boxed{t_{i-1}, o_{1:N}})$

- For given tag t_i and your fixed collection of $\{f_k\}$ and $\{w_k\}$ of feature functions and weights for **that tag**.
- Classic technique: take the exponent of the sum of weighted-feature values, and then normalize.
- $P_{\text{MEMM}}(t_i \mid t_{i-1}, o_{1:N}) = \dfrac{\exp(\sum_k w_k^{t_i} . f_k(t_i, t_{i-1}, o_{1:N}, i))}{Z}$

Formalizing features and evidence weights

Features: $f_1(t_i, t_{i-1}, o_{1:N}, i)$ = 1 if o_i ends in "ing", $f_2(t_i, t_{i-1}, o_{1:N}, i)$ = length of o_i

Weights for VERB $= [w_1^{VERB}, w_2^{VERB}] = [3{,}0]$,

Weights for PP $= [w_1^{PP}, w_2^{PP}] = [-1, -2]$

$o_{1:N}$ = I am sitting in

$P_{\text{MEMM}}(t_3 \,|\, \text{VERB}, o_{1:N})$

Step1: Extract features

$f_1 = 1$

$f_2 = 7$

Step2: Compute Exponentials

$P_{\text{MEMM}}(t_3 = VERB \,|\, \text{VERB}, o_{1:N}) = \exp(3 \times 1 + 0 \times 7)/Z$

$P_{\text{MEMM}}(t_3 = PP \,|\, \text{VERB}, o_{1:N}) = \exp((-1) \times 1 + (-2) \times 7)/Z$

HMMs vs MEMMs as taggers via Viterbi

- HMMs: $v_i(t) = \max_{\text{possible prev}_t} \boxed{P_{\text{HMM}}(t \mid \text{prev}_t)\ P_{\text{HMM}}(o_i \mid t)}\ v_{i-1}(\text{prev}_t)$

No emission/transition matrices

- MEMMs: $v_i(t) = \max_{\text{possible prev}_t} \boxed{P_{\text{MEMMs}}(t \mid \text{prev}_t, o_{1:N})}\ v_{i-1}(\text{prev}_t)$

Logistic Regression Model

- $P(t_i \mid t_{i-1}, o_{1:N}) = \dfrac{\exp(\sum_k w_k . f_k(t_i, t_{i-1}, o_{1:N}, i))}{Z}$ <u>Multinomial logistic regression model</u>

- Applicable to all **text classification** tasks:
  - Input: some text **x** (e.g. documents, sentences)
  - Output: label **y** (finite set of labels)
  - Classifier: Assign $P(y \mid \mathbf{x})$ for all $y \in \mathbf{y}$

Text Classification

Task	Input **x**	Output **x**
Sentiment Analysis	"The movie was great" "The actor is great, movie is dull"	{positive, negative}
Topic Identification	News articles	{politics, sports, health, etc.}
Spam / Not spam	"Win $10Million" "CS4740 announcement"	{spam, not-spam}

- Define features that make sense for the task.
- Learn weights. (**how???**)

Slide Acknowledgements

- Earlier versions of this course offerings including materials from Claire Cardie, Marten van Schijndel, Lillian Lee.