

# Part of Speech Tagging: Viterbi, Forward, Backward, Forward- Backward, Baum-Welch

COMP-550

Sept 28, 2017

# Outline

---

Hidden Markov models: review of last class

Things to do with HMMs:

- Forward algorithm
- Backward algorithm
- Viterbi algorithm
- Baum-Welch as Expectation Maximization

# Parts of Speech in English

---

Nouns	<i>restaurant, me, dinner</i>
Verbs	<i>find, eat, is</i>
Adjectives	<i>good, vegetarian</i>
Prepositions	<i>in, of, up, above</i>
Adverbs	<i>quickly, well, very</i>
Determiners	<i>the, a, an</i>

# Sequence Labelling

---

Predict labels for *an entire sequence of inputs*:

? ? ? ? ? ? ? ? ?

Pierre Vinken , 61 years old , will join the board ...



NNP NNP , CD NNS JJ , MD VB DT NN

Pierre Vinken , 61 years old , will join the board ...

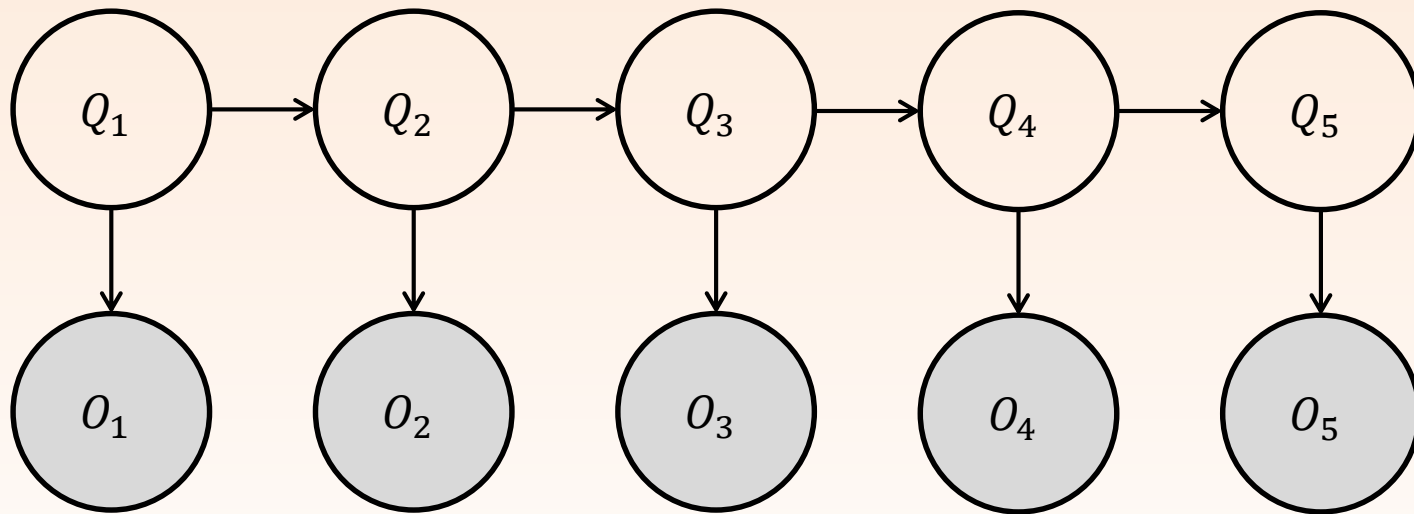
Must consider:

Current word

Previous context

# Decomposing the Joint Probability

Graph specifies how joint probability decomposes



$$P(\mathbf{O}, \mathbf{Q}) = P(Q_1) \prod_{t=1}^{T-1} P(Q_{t+1}|Q_t) \prod_{t=1}^T P(O_t|Q_t)$$

Initial state probability

State transition probabilities

Emission probabilities

# Model Parameters

---

Let there be  $N$  possible tags,  $W$  possible words

Parameters  $\theta$  has three components:

1. Initial probabilities for  $Q_1$ :

$$\Pi = \{\pi_1, \pi_2, \dots, \pi_N\} \quad (\text{categorical})$$

2. Transition probabilities for  $Q_t$  to  $Q_{t+1}$ :

$$A = \{a_{ij}\} \quad i, j \in [1, N] \quad (\text{categorical})$$

3. Emission probabilities for  $Q_t$  to  $O_t$ :

$$B = \{b_i(w_k)\} \quad i \in [1, N], k \in [1, W] \quad (\text{categorical})$$

How many distributions and values of each type are there?

# Model Parameters' MLE

---

Recall categorical distributions' MLE:

$$P(\text{outcome } i) = \frac{\#(\text{outcome } i)}{\#(\text{all events})}$$

For our parameters:

$$\pi_i = P(Q_1 = i) = \frac{\#(Q_1 = i)}{\#(\text{sentences})}$$

$$a_{ij} = P(Q_{t+1} = j | Q_t = i) = \#(i, j) / \#(i)$$

$$b_{ik} = P(O_t = k | Q_t = i) = \#(\text{word } k, \text{ tag } i) / \#(i)$$

# Questions for an HMM

---

1. Compute likelihood of a sequence of observations,  
 $P(\mathbf{O}|\theta)$  Forward algorithm, backward algorithm

2. What state sequence best explains a sequence of observations?  
 $\operatorname{argmax}_Q P(Q, \mathbf{O}|\theta)$  Viterbi algorithm

3. Given an observation sequence (without labels),  
what is the best model for it?

Forward-backward algorithm  
Baum-Welch algorithm  
Expectation Maximization



# Q1: Compute Likelihood

---

Marginalize over all possible state sequences

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

*Problem:* Exponentially many paths ( $N^T$ )

*Solution:* **Forward algorithm**

- *Dynamic programming* to avoid unnecessary recalculations
- Create a table of all the possible state sequences, annotated with probabilities

# Forward Algorithm

Trellis of possible state sequences

	$O_1$	$O_2$	$O_3$	$O_4$	$O_5$
VB	$\alpha_{VB}(1)$	$\alpha_{VB}(2)$	$\alpha_{VB}(3)$	$\alpha_{VB}(4)$	$\alpha_{VB}(5)$
NN	$\alpha_{NN}(1)$	$\alpha_{NN}(2)$	$\alpha_{NN}(3)$	$\alpha_{NN}(4)$	$\alpha_{NN}(5)$
DT	$\alpha_{DT}(1)$	$\alpha_{DT}(2)$	$\alpha_{DT}(3)$	$\alpha_{DT}(4)$	$\alpha_{DT}(5)$
JJ	$\alpha_{JJ}(1)$	$\alpha_{JJ}(2)$	$\alpha_{JJ}(3)$	$\alpha_{JJ}(4)$	$\alpha_{JJ}(5)$
CD	$\alpha_{CD}(1)$	$\alpha_{CD}(2)$	$\alpha_{CD}(3)$	$\alpha_{CD}(4)$	$\alpha_{CD}(5)$

$\alpha_i(t)$  is  $P(\mathbf{O}_{1:t}, Q_t = i | \theta)$

- Probability of current tag and words up to now

# First Column

$\alpha_i(t)$  is  $P(\mathbf{O}_{1:t}, Q_t = i | \theta)$

	$O_1$	$O_2$	$O_3$	$O_4$	$O_5$
VB	$\alpha_{VB}(1)$	$\alpha_{VB}(2)$	$\alpha_{VB}(3)$	$\alpha_{VB}(4)$	$\alpha_{VB}(5)$
NN	$\alpha_{NN}(1)$	$\alpha_{NN}(2)$	$\alpha_{NN}(3)$	$\alpha_{NN}(4)$	$\alpha_{NN}(5)$
DT	$\alpha_{DT}(1)$	$\alpha_{DT}(2)$	$\alpha_{DT}(3)$	$\alpha_{DT}(4)$	$\alpha_{DT}(5)$
JJ	$\alpha_{JJ}(1)$	$\alpha_{JJ}(2)$	$\alpha_{JJ}(3)$	$\alpha_{JJ}(4)$	$\alpha_{JJ}(5)$
CD	$\alpha_{CD}(1)$	$\alpha_{CD}(2)$	$\alpha_{CD}(3)$	$\alpha_{CD}(4)$	$\alpha_{CD}(5)$

States

Time

$\alpha_j(1) = \pi_j b_j(O_1)$

Consider:

- Initial state probability
- First emission

# Middle Cells

$\alpha_i(t)$  is  $P(\mathbf{O}_{1:t}, Q_t = i | \theta)$

	$O_1$	$O_2$	$O_3$	$O_4$	$O_5$	
<b>States</b>	VB	$\alpha_{VB}(1)$	$\alpha_{VB}(2)$	$\alpha_{VB}(3)$	$\alpha_{VB}(4)$	$\alpha_{VB}(5)$
	NN	$\alpha_{NN}(1)$	$\alpha_{NN}(2)$	$\alpha_{NN}(3)$	$\alpha_{NN}(4)$	$\alpha_{NN}(5)$
	DT	$\alpha_{DT}(1)$	$\alpha_{DT}(2)$	$\alpha_{DT}(3)$	$\alpha_{DT}(4)$	$\alpha_{DT}(5)$
	JJ	$\alpha_{JJ}(1)$	$\alpha_{JJ}(2)$	$\alpha_{JJ}(3)$	$\alpha_{JJ}(4)$	$\alpha_{JJ}(5)$
	CD	$\alpha_{CD}(1)$	$\alpha_{CD}(2)$	$\alpha_{CD}(3)$	$\alpha_{CD}(4)$	$\alpha_{CD}(5)$

$$\alpha_j(t) = \sum_{i=1}^N \alpha_i(t-1) a_{ij} b_j(O_t)$$

**Time**

Consider:

- All possible ways to get to current state
- Emission from current state

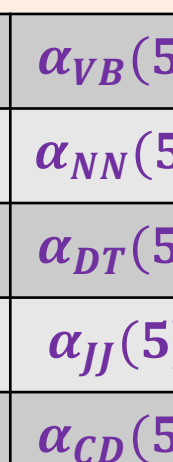
# After Last Column

$\alpha_i(t)$  is  $P(\mathbf{O}_{1:t}, Q_t = i | \theta)$

	$O_1$	$O_2$	$O_3$	$O_4$	$O_5$	
States	VB	$\alpha_{VB}(1)$	$\alpha_{VB}(2)$	$\alpha_{VB}(3)$	$\alpha_{VB}(4)$	$\alpha_{VB}(5)$
	NN	$\alpha_{NN}(1)$	$\alpha_{NN}(2)$	$\alpha_{NN}(3)$	$\alpha_{NN}(4)$	$\alpha_{NN}(5)$
	DT	$\alpha_{DT}(1)$	$\alpha_{DT}(2)$	$\alpha_{DT}(3)$	$\alpha_{DT}(4)$	$\alpha_{DT}(5)$
	JJ	$\alpha_{JJ}(1)$	$\alpha_{JJ}(2)$	$\alpha_{JJ}(3)$	$\alpha_{JJ}(4)$	$\alpha_{JJ}(5)$
	CD	$\alpha_{CD}(1)$	$\alpha_{CD}(2)$	$\alpha_{CD}(3)$	$\alpha_{CD}(4)$	$\alpha_{CD}(5)$

**Time**

$$P(\mathbf{O} | \theta) = \sum_{j=1}^N \alpha_j(T)$$



Sum over last column for overall likelihood

# Forward Algorithm Summary

---

Create trellis  $\alpha_i(t)$  for  $i = 1 \dots N, t = 1 \dots T$

$\alpha_j(1) = \pi_j b_j(O_1)$  for  $j = 1 \dots N$

for  $t = 2 \dots T$ :

for  $j = 1 \dots N$ :

$$\alpha_j(t) = \sum_{i=1}^N \alpha_i(t-1) a_{ij} b_j(O_t)$$

$$P(\mathbf{O}|\theta) = \sum_{j=1}^N \alpha_j(T)$$

Runtime:  $O(N^2 T)$

# Backward Algorithm

---

Nothing stops us from going backwards too!

- This is not just for fun, as we'll see later.

Define new trellis with cells  $\beta_i(t)$

$$\beta_i(t) = P(\mathbf{O}_{t+1:T} | Q_t = i, \theta)$$

- Probability of all subsequent words, given current tag is  $i$ .
- Note that unlike  $\alpha_i(t)$ , it *excludes* the current word

# Backward Algorithm Summary

---

Create trellis  $\beta_i(t)$  for  $i = 1 \dots N, t = 1 \dots T$


$$\beta_i(T) = 1 \text{ for } i = 1 \dots N$$

for  $t = T-1 \dots 1$ :

for  $i = 1 \dots N$ :

$$\beta_i(t) = \sum_{j=1}^N a_{ij} b_j(O_{t+1}) \beta_j(t+1)$$

Remember  $\beta_j(t+1)$  does not include  $b_j(O_{t+1})$ ,  
so we need to add this factor in!



$$P(\mathbf{O}|\theta) = \sum_{i=1}^N \pi_i b_i(O_1) \beta_i(1)$$

Runtime:  $O(N^2T)$



# Forward and Backward

---

$$\alpha_i(t) = P(\mathbf{O}_{1:t}, Q_t = i | \theta)$$

$$\beta_i(t) = P(\mathbf{O}_{t+1:T} | Q_t = i, \theta)$$

That means

$$\alpha_i(t)\beta_i(t) = P(\mathbf{O}, Q_t = i | \theta)$$

Probability of the *entire* sequence of observations, and we are in state  $i$  at timestep  $t$ .

Thus,

$$P(\mathbf{O} | \theta) = \sum_{i=1}^N \alpha_i(t)\beta_i(t) \text{ for any } t = 1 \dots T$$

# Working in the Log Domain

---

Practical note: need to avoid underflow—work in log domain

$$\log\left(\prod p_i\right) = \sum \log p_i = \sum a_i$$

Log sum trick

$$\log\left(\sum p_i\right) = \log\left(\sum \exp a_i\right)$$

$$a_i = \log p_i$$


Let  $b = \max a_i$

$$\begin{aligned}\log\left(\sum \exp a_i\right) &= \log \exp(b) \sum \exp(a_i - b) \\ &= b + \log \sum \exp(a_i - b)\end{aligned}$$

## Q2: Sequence Labelling

---

Find most likely state sequence for a sample

$$Q^* = \operatorname{argmax}_Q P(Q, O|\theta)$$

*Intuition:* use forward algorithm, but replace summation with max

This is now called the **Viterbi algorithm**.

Trellis cells:

$$\delta_i(t) = \max_{Q_{1:t-1}} P(Q_{1:t-1}, O_{1:t}, Q_t = i|\theta)$$

# Viterbi Algorithm Summary

---

Create trellis  $\delta_i(t)$  for  $i = 1 \dots N, t = 1 \dots T$

$$\delta_j(1) = \pi_j b_j(O_1) \text{ for } j = 1 \dots N$$

for  $t = 2 \dots T$ :

for  $j = 1 \dots N$ :

$$\delta_j(t) = \max_i \delta_i(t-1) a_{ij} b_j(O_t)$$

Take  $\max_i \delta_i(T)$

Runtime:  $O(N^2 T)$

# Backpointers

	$O_1$	$O_2$	$O_3$	$O_4$	$O_5$
VB	$\delta_{VB}(1)$	$\delta_{VB}(2)$	$\delta_{VB}(3)$	$\delta_{VB}(4)$	$\delta_{VB}(5)$
NN	$\delta_{NN}(1)$	$\delta_{NN}(2)$	$\delta_{NN}(3)$	$\delta_{NN}(4)$	$\delta_{NN}(5)$
DT	$\delta_{DT}(1)$	$\delta_{DT}(2)$	$\delta_{DT}(3)$	$\delta_{DT}(4)$	$\delta_{DT}(5)$
JJ	$\delta_{JJ}(1)$	$\delta_{JJ}(2)$	$\delta_{JJ}(3)$	$\delta_{JJ}(4)$	$\delta_{JJ}(5)$
CD	$\delta_{CD}(1)$	$\delta_{CD}(2)$	$\delta_{CD}(3)$	$\delta_{CD}(4)$	$\delta_{CD}(5)$

Time

- Keep track of where the max entry to each cell came from
- Work backwards to recover best label sequence

# Exercise

---

3 states (X, Y, Z), 2 possible emissions (!,@)

$$\pi = \langle 0.2 \quad 0.5 \quad 0.3 \rangle$$

$$A = \begin{bmatrix} 0.5 & 0.4 & 0.1 \\ 0.2 & 0.3 & 0.5 \\ 0.1 & 0.1 & 0.8 \end{bmatrix}$$

$$B = \begin{bmatrix} 0.1 & 0.9 \\ 0.5 & 0.5 \\ 0.7 & 0.3 \end{bmatrix}$$

Run the forward, backward, and Viterbi algorithms on the emission sequence "!@@"

# Q3: Unsupervised Training

---

*Problem:* No state sequences to compute above

*Solution:* Guess the state sequences

Initialize parameters randomly

Repeat for a while:

1. Predict the current state sequences using the current model
2. Update the current parameters based on current predictions

# "Hard EM" or Viterbi EM

---

Initialize parameters randomly

Repeat for a while:

1. Predict the current state sequences using the current model with the Viterbi algorithm
2. Update the current parameters using the current predictions as in the supervised learning case

Can also use "soft" predictions; i.e., the probabilities of all the possible state sequences



# Baum-Welch Algorithm

---

a.k.a., **Forward-backward** algorithm

EM algorithm applied to HMMs:

- |                     |                                                                                                                |
|---------------------|----------------------------------------------------------------------------------------------------------------|
| <b>Expectation</b>  | Get <i>expected</i> counts for hidden structures using current $\theta^k$ .                                    |
| <b>Maximization</b> | Find $\theta^{k+1}$ to maximize the likelihood of the training data given the expected counts from the E-step. |

# Responsibilities Needed

---

Supervised/Hard EM:      A single tag

Baum-Welch:              *Distribution* over tags

Call such probabilities **responsibilities**.

$$\gamma_i(t) = P(Q_t = i | \mathbf{O}, \theta^k)$$

- Probability of being in state  $i$  at time  $t$  given the observed sequence under the current model.

$$\xi_{ij}(t) = P(Q_t = i, Q_{t+1} = j | \mathbf{O}, \theta^k)$$

- Probability of transitioning from  $i$  at time  $t$  to  $j$  at time  $t + 1$ .

## E-Step $\gamma$

---

$$\begin{aligned}\gamma_i(t) &= P(Q_t = i | \mathbf{O}, \theta^k) \\ &= \frac{P(Q_t = i, \mathbf{O} | \theta^k)}{P(\mathbf{O} | \theta^k)} \\ &= \frac{\alpha_i(t)\beta_i(t)}{P(\mathbf{O} | \theta^k)}\end{aligned}$$

# E-Step $\xi$

$$\begin{aligned}\xi_{ij}(t) &= P(Q_t = i, Q_{t+1} = j | \mathbf{O}, \theta^k) \\ &= \frac{P(Q_t = i, Q_{t+1} = j, \mathbf{O} | \theta^k)}{P(\mathbf{O} | \theta^k)} \\ &= \frac{\alpha_i(t) a_{ij} b_j(O_{t+1}) \beta_j(t+1)}{P(\mathbf{O} | \theta^k)}\end{aligned}$$

Beginning to state  $i$  at time  $t$

Transition from  $i$  to  $j$

Emit  $O_{t+1}$

Rest of the sequence

# M-Step

"Soft" versions of the MLE updates:

$$\pi_i^{k+1} = \gamma_i(1)$$

$$a_{ij}^{k+1} = \frac{\sum_{t=1}^{T-1} \xi_{ij}(t)}{\sum_{t=1}^{T-1} \gamma_i(t)}$$

$$b_i^{k+1}(w_k) = \frac{\sum_{t=1}^T \gamma_i(t) |_{o_t=w_k}}{\sum_{t=1}^T \gamma_i(t)}$$

*Compare:*

$$\frac{\#(i, j)}{\#(i)}$$

$$\frac{\#(\text{word } k, \text{tag } i)}{\#(i)}$$

- With multiple sentences, sum up expected counts over all sentences

# When to Stop EM?

---

## Multiple options

When training set likelihood stops improving

When prediction performance on held-out **development** or **validation** set stops improving

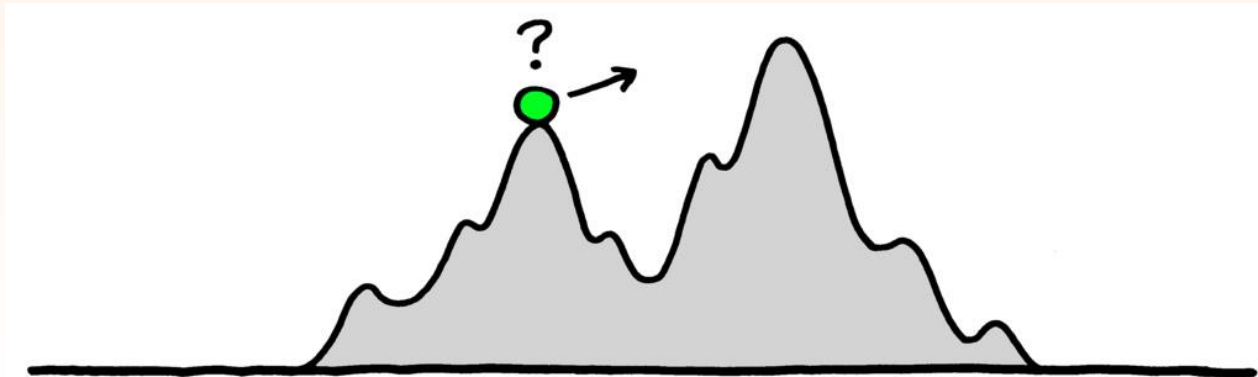
# Why Does EM Work?

EM finds a local optimum in  $P(\mathbf{O}|\theta)$ .

You can show that after each step of EM:

$$P(\mathbf{O}|\theta^{k+1}) > P(\mathbf{O}|\theta^k)$$

However, this is not necessarily a global optimum.



# Proof of Baum-Welch Correctness

---

**Part 1:** Show that in our procedure, one iteration corresponds to this update:

$$\theta^{k+1} = \operatorname{argmax}_{\theta} \sum_{\mathbf{Q}} \log[P(\mathbf{O}, \mathbf{Q}|\theta)]P(\mathbf{Q}|\mathbf{O}, \theta^k)$$

<http://www.cs.berkeley.edu/~stephentu/writeups/hmm-baum-welch-derivation.pdf>

**Part 2:** Show that improving the quantity

$$\sum_{\mathbf{Q}} \log[P(\mathbf{O}, \mathbf{Q}|\theta)]P(\mathbf{Q}|\mathbf{O}, \theta^k)$$

corresponds to improving  $P(\mathbf{O}|\theta)$

[https://en.wikipedia.org/wiki/Expectation%E2%80%93maximization\\_algorithm#Proof\\_of\\_correctness](https://en.wikipedia.org/wiki/Expectation%E2%80%93maximization_algorithm#Proof_of_correctness)



# Dealing with Local Optima

---

## Random restarts

- Train multiple models with different random initializations
- Model selection on development set to pick the best one

## Biased initialization

- Bias your initialization using some external source of knowledge (e.g., external corpus counts or clustering procedure, expert knowledge about domain)
- Further training will hopefully improve results

# Caveats

---

Baum-Welch with no labelled data generally gives poor results, at least for linguistic structure (~40% accuracy, according to Johnson, (2007))

**Semi-supervised** learning: combine small amounts of labelled data with larger corpus of unlabelled data

# In Practice

---

Per-token (i.e., per word) accuracy results on WSJ corpus:

Most frequent tag baseline	~90—94%
HMMs (Brants, 2000)	96.5%
Stanford tagger (Manning, 2011)	97.32%

# Other Sequence Modelling Tasks

---

## Chunking (a.k.a., shallow parsing)

- Find syntactic chunks in the sentence; not hierarchical  
[<sub>NP</sub>The chicken] [<sub>V</sub>crossed] [<sub>NP</sub>the road] [<sub>P</sub>across] [<sub>NP</sub>the lake].

## Named-Entity Recognition (NER)

- Identify elements in text that correspond to some high level categories (e.g., PERSON, ORGANIZATION, LOCATION)  
[<sub>ORG</sub>McGill University] is located in [<sub>LOC</sub>Montreal, Canada].
- Problem: need to detect spans of multiple words

# First Attempt

---

Simply label words by their category

ORG    ORG    ORG   -    ORG    -    -    -    LOC

*McGill, UQAM, UdeM, and Concordia are located in Montreal.*

What is the problem with this scheme?

# IOB Tagging

---

Label whether a word is inside, outside, or at the beginning of a span as well.

For  $n$  categories,  $2n+1$  total tags.

$B_{ORG}$        $I_{ORG}$      $O$      $O$      $O$        $B_{LOC}$        $I_{LOC}$

McGill University is located in Montreal, Canada

$B_{ORG}$     $B_{ORG}$     $B_{ORG}$     $O$      $B_{ORG}$     $O$      $O$      $O$      $B_{LOC}$

*McGill, UQAM, UdeM, and Concordia are located in Montreal.*