# Gibbs Sampling in Traditional NLP Tasks

David Mareček

November 25, 2021

EUROPEAN UNION
European Structural and Investment Fund
Operational Programme Research,
Development and Education

Charles University
Faculty of Mathematics and Physics
Institute of Formal and Applied Linguistics

unless otherwise stated

# Unsupervised, Semi-supervised, Supervised...

**Tasks:**

- Word Alignment (unsupervised, Expectation Maximization)
- Part of Speech Tagging (supervised, data: PennTB, PDT, Universal Dependencies, ...)
- Syntactic Parsing (supervised, data: PennTB, PDT, Unicersal Dependencies, ...)

**Methods:**

- *Supervised*: we use only the labelled training data
- *Semi-supervised*: we have only a small portion of labelled data or manually created rules and a lot of unlabelled data (e.g. raw texts).
- *Unsupervised*: we don't use any labelled data and any rules

The boundaries between these concepts are very vague. Sometimes it is better to speak about *degree o supervision*.

# Advantages of Unsupervised Approaches

When to choose an unsupervised approach:

- We have no labelled data.
- Manual annotation of data is very expensive and time consuming.
- The rules for annotators would be very complicated.
- The task itself is hard.
- Inter-annotator agreement is very low.
- We are not sure what the annotation should look like.
- We are not sure what annotation suits best our target application.

# Using Gibbs sampling and Chinese Restaurant Process

1. Devise a generative process that would generate your labelled data.
2. Think about the probability distributions in your model. Which of them are sparse and may be modelled by CRP?
3. What would be the small changes made during the Gibbs sampling. What data and variables are affected by the small change proposed?

Predictive probability:

$$p(item) = \frac{\alpha P_{base}(item) + \mathsf{count}(item)}{\alpha + |data| - 1}$$

$P_{base}$ must sum to one and may be uniform if you have fixed number of classes.

# Word Alignment

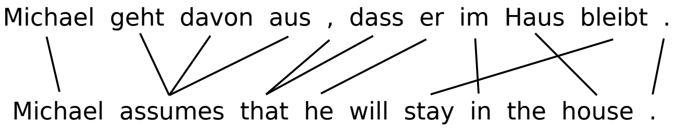
# Word Alignment

**The task**: Given a lot of parallel sentences, find the links between corresponding words

Solved mainly by unsupervised approaches:

- Tools: Fast-Align (`https://github.com/clab/fast_align`), GIZA++
- Expectation-Maximization (taught at NPFL087 Statistical Machine Translation)
- Gibbs Sampling (in this lecture)

**The sub-task**: For each word in one language, find its counterpart in the other language.



# Word Alignment – Generative Model

**Generative story:**

For each word in the source sentence, generate its counterparts in the target sentence.

Probability of data:

$$P(E, A|F) = \prod_{i=1}^{n} p(e_i|f_{a(i)})$$

Application of the Chinese Restaurant Process as a power-law:

$$P(E, A|F) = \prod_{i=1}^{n} \frac{\alpha P_0(e_i|f_{a(i)}) + count([e_i, f_{a(i)}] \in data)}{\alpha + count(f_{a(i)} \in data)}$$

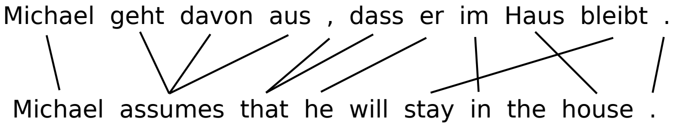
We set the base probability as uniform distributions over the number of unique words in E:

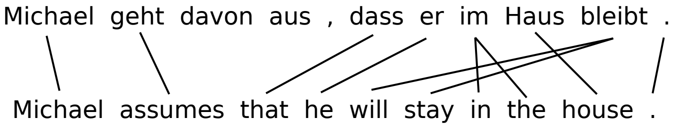
$$P(E, A|F) = \prod_{i=1}^{n} \frac{\beta + count([e_i, f_{a(i)}] \in data)}{|W|\beta + count(f_{a(i)} \in data)}$$

# Word Alignment – What direction to choose?

$$P(E, A|F) = \prod_{i=1}^{n} \frac{\beta + count([e_i, f_{a(i)}] \in data)}{|W|\beta + count(f_{a(i)} \in data)}$$

Should we align each word in E with just one word in F or vice versa? What problem can happen?



# Word Alignment – What direction to choose?

$$P(E, A|F) = \prod_{i=1}^{n} \frac{\beta + count([e_i, f_{a(i)}] \in data)}{|W|\beta + count(f_{a(i)} \in data)}$$

The generative model works in one-to-many scenario. Each word in E is aligned with just one word in F. The $f_{a(i)}$ may occur in more than one factor in the formula or not at all.

We can use both Expectation Maximization algorithm and Gibbs Sampling to estimate the model.

Usually we estimate models in both the directions E→F and F→E and use so ccalled symmetrization method to get a bidirectional word alignment.

# Word Alignment – Gibbs Sampling

1. Initialization: randomly assign one counterpart word in F for each word in E.
2. Iterate across all the words in E:
   2.1 Remove the link $a(i)$ between $e_i$ and $f_a(i)$ from the data.
   2.2 Compute the link probabilities to all words in the corresponding sentence in F.
   2.3 Sample one of the words from F and update the link $a(i)$.
   2.4 Add the link $a(i)$ to data.
3. Repeat until convergence.

# Part of Speech Tagging

# Part-of-Speech tagging

Generally trained in supervised or semi-supevised way:

- Universal Dependencies (common annotation for 90 different languages)
- Pre-trained contextual embeddings on large raw data (mBERT)
- UDPipe tool (`http://ufal.mff.cuni.cz/udpipe`)

Unsupervised PoS tagging = Word Clustering in to N classes

- Expectation-Maximization (taught at NPFL067 Statistical Methods in NLP I)
- Gibbs Sampling (in this lecture)
- Both the methods may be also semi-supervised - the classes of the known words are fixed.

# POS tagging – Generative Model

**Generative story:**

1. Start with a start-symbol tag $t_0 =$ <S>.
2. Generate the next PoS tag from a probability distribution conditioned on the previous tag $p(t_i | t_{i-1})$.
3. Generate the word from a probability distribution conditioned on the current tag $p(w_i | t_i)$.
4. Go to step 2 and repeat until the end-symbol tag is generated.

We observe only the words. Part-of-speech tags are our hidden variables.

# POS tagging – Sparse Distributions

**Sparse distributions:**
The probability of a tag is highly determined by the previous tag.

- nouns after determiners, prepositions, possessives ...
- $p(t_i|t_{i-1})$ can have symetric Dirichlet prior with $\alpha_T < 1$

The probability of a word is determined by its POS tag.

- In Czech: you can label many words without the knowledge of their context
- In English: not as strong
- $p(w_i|t_i)$ can have symetric Dirichlet prior with $\alpha_W < 1$

# POS tagging – Probability of Data

The overall probability of the whole text together with the POS tags:

$$p(T,W) = p(T) \cdot p(W|T) = \prod_{i=1}^{n} p(t_i|t_{i-1}) \prod_{i=1}^{n} p(w_i|t_i)$$

Application of the Chinese Restaurant Process as a power-law:

$$p(T,W) = \prod_{i=1}^{n} \frac{\alpha_T P_0(t_i|t_{i-1}) + count([t_{i-1}, t_i] \in data)}{\alpha_T + count(t_{i-1} \in data)} \prod_{i=1}^{n} \frac{\alpha_W P_0(w_i|t_i) + count([w_i, t_i] \in dat}{\alpha_T + count(t_i \in data)}$$

We set the base probabilities as uniform distributions over numbers of tags and words.

$$p(T,W) = \prod_{i=1}^{n} \frac{\beta_T + count([t_{i-1}, t_i] \in data)}{|T|\beta_T + count(t_{i-1} \in data)} \prod_{i=1}^{n} \frac{\beta_W + count([w_i, t_i] \in data)}{|W|\beta_T + count(t_i \in data)}$$

# POS tagging – Gibbs Sampling

**Small change**: Change the tag of one chosen word.

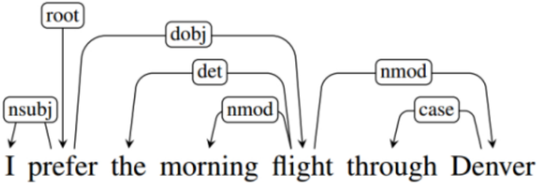
**Gibbs sampling**: We will iteratively take out one word from the data, compute probability distribution across all possible tags on that position and sample one tag from that distribution.

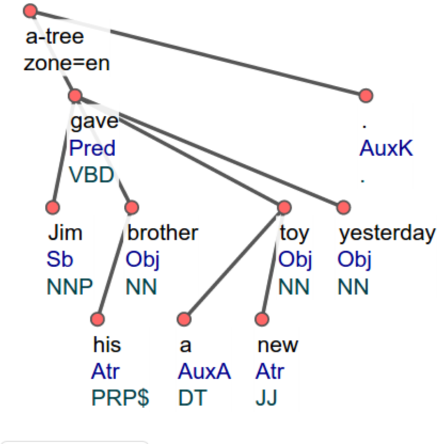
**Affected factors**: The change of tag $t_i$ will affect three factors in the overall probability: $p(t_i|t_{i-1})$, $p(t_{i+1}|t_i)$, $p(w_i|t_i)$. All the three factors must be removed from the data. To compute the predictions for sampling the new tag.

**Exchangeability**: The factors are exchangeable (CRP), so we may act as we are changing the tag at the end of the sequence. The overall probability of data is equal.

# Dependency Parsing

# Dependency Parsing



# Dependency Parsing

Generally trained in supervised or semi-supevised way:

- Universal Dependencies (common annotation for 90 different languages)
- Pre-trained contextual embeddings on large raw data (mBERT)
- UDPipe tool (`http://ufal.mff.cuni.cz/udpipe`)

Unsupervised approaches:

- Expectation-Maximization, Variational Inference, Gibbs Sampling
- Dependency Model with Valence
- May be also semi-supervised. The annotation of some of the trees or phrases in the data may be fixed.

# Dependency Parsing – Generative Model

**Dependency Model with Valence:**

- Generate the root node (word).
- For each node, generate its edges on the left, then STOP.
- For each node, generate its odges on the right, then STOP.
- For each generated edge, generate the respective dependent word.

Two types of conditional probabilities:

- $p(STOP|w_p, dir)$ – probability that no other dependents of $w$ in the direction $dir \in [L, R]$ will be generated.
- $p(w_d|w_p, dir)$ – probability, that the left dependent of the word $w_p$ in the direction $dir$ is the $w_d$.

# Dependency Parsing – Generative Model

Overall probability of the Treebank data:

$$P(T) = \prod_{i=1}^{n} \Big( p(w_i|w_{g(i)}, d(i)) \cdot (p(\neg STOP|w_{g(i)}, d(i)) \cdot p(STOP|w_i, L) \cdot p(STOP|w_i, R) \Big)$$

Application of the Chinese Restaurant Process as a power-law:

$$p(\neg STOP|w_i, d(i)) = \frac{\alpha_S + count([w_i, d(i)])}{2 \cdot \alpha_S + count([w_i, d(i)]) + count(w_i)}$$

$$p(STOP|w_i, d_i) = 1 - p(\neg STOP|w_i, d_i)$$

$$p(w_i|w_{g(i)}, d(i)) = \frac{\alpha_A + count([w_i, w_{g(i)}, d(i)])}{|W| \cdot \alpha_A + count([w_{g(i)}, d(i)])}$$

- $g(i)$ – The index of the word governing the word $w_i$
- $d(i) \in [L, R]$ – The direction of the edge attaching the word $w_i$ (left or right).

# Dependency Parsing – Gibbs Sampling

**Small change:**

Remove one dependency between two words and sample a new parent word?

- Sample one from all the words in the sentence?
- Sample only from possible words preserving the tree structure?

Do not converge to a good solutions.

**Bigger change:**

The whole tree must be removed from the data and the whole new tree must be sampled from all possible trees.

Too many possible trees? $\rightarrow$ Dynamic programming algorithm.

**Gibbs Sampling:**

1. Initialize the treebank by random trees.
2. Go through the sentences in many iterations.
3. For each sentence, resample its tree based on all the other trees in the current treebank.

# Hidden Structures in Deep Learning Models

# Hidden Structures in Deep Learning Models

Kinds of these traditional linguistic tasks solved in an unsupervised way are probably hidden inside many of currently widely used Deep Neural-Network models.

**Neural Machine Translation using Transformer architecture**:

| | | |
|---|---|---|
| Word alignment | $\sim$ | Cross-lingual attentions |
| Dependency parsing | $\sim$ | Self-attentions |
| POS tagging | $\sim$ | Contextual embeddings of words |

More information will be given at the last lecture (in January).