TEXAS
The University of Texas at Austin

I320D – Topics in Human Centered Data Science

Text Mining and NLP Essentials

Week 8: Machine Learning for NLP -2, Structured and Unsupervised ML

Dr. Abhijit Mishra

Before we start …

- Ongoing assignment: **Assignment 4 (information extraction)**
  - **Deadline : 03/08**
- **Fill out this form for group formation**
  - https://forms.gle/H9akcB9PGLNEmmUU8
  - Only 3 groups formed so far …hurry up!!!

Rosalyn Lu, Shikha Mheta, Saloni Sharma
Sriya Nimmagadda, Ashwati Manoj, Saathvik Konidena, Kiyonna Kapoor
Fiona Romanoschi, Ethan Wen, Alex Imhoff

Before we start …

- **Graded Quiz in Week 10**
  - 30 MCQs, 15 mins, On-canvas
  - Full points for appearing **In-Class**
  - **Will be counted towards “in-class participation”**

So far in I320D – Text Mining and NLP

- W1. Language and Ambiguity
- W2. Basics of Text Data and Linguistic Concepts
- W3. Text Preprocessing Techniques
- W4. Lexical Analysis
- W5. Syntax Analysis
- W6. Information Extraction

W7. Machine Learning Methods for NLP
W8. Unsupervised ML and Topic Modeling Basics
W10-W11. Deep learning for NLP
W12. NLP Applications
W13. Small and Large Language Models and Prompt Engineering Basics
W14. Knowledge Networks
W15. Evaluation Metrics

Week 7: Recap

- Machine Learning for NLP
  - What is Machine Learning?
  - NLP tasks that require NLP
  - Text Classification with ML

Recap: ML Workflow

Developer

Gather Labeled Data

Feature Engineering

Features, labels

Splitting Data

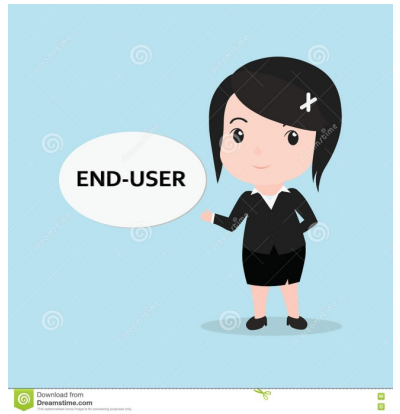
Model

Training + Evaluation

Features , labels

Training

Ship or Deploy best model and feature engineering module



Real time Data

Feature Engineering

Features

Best Model

Predicted label

Deployment and usage

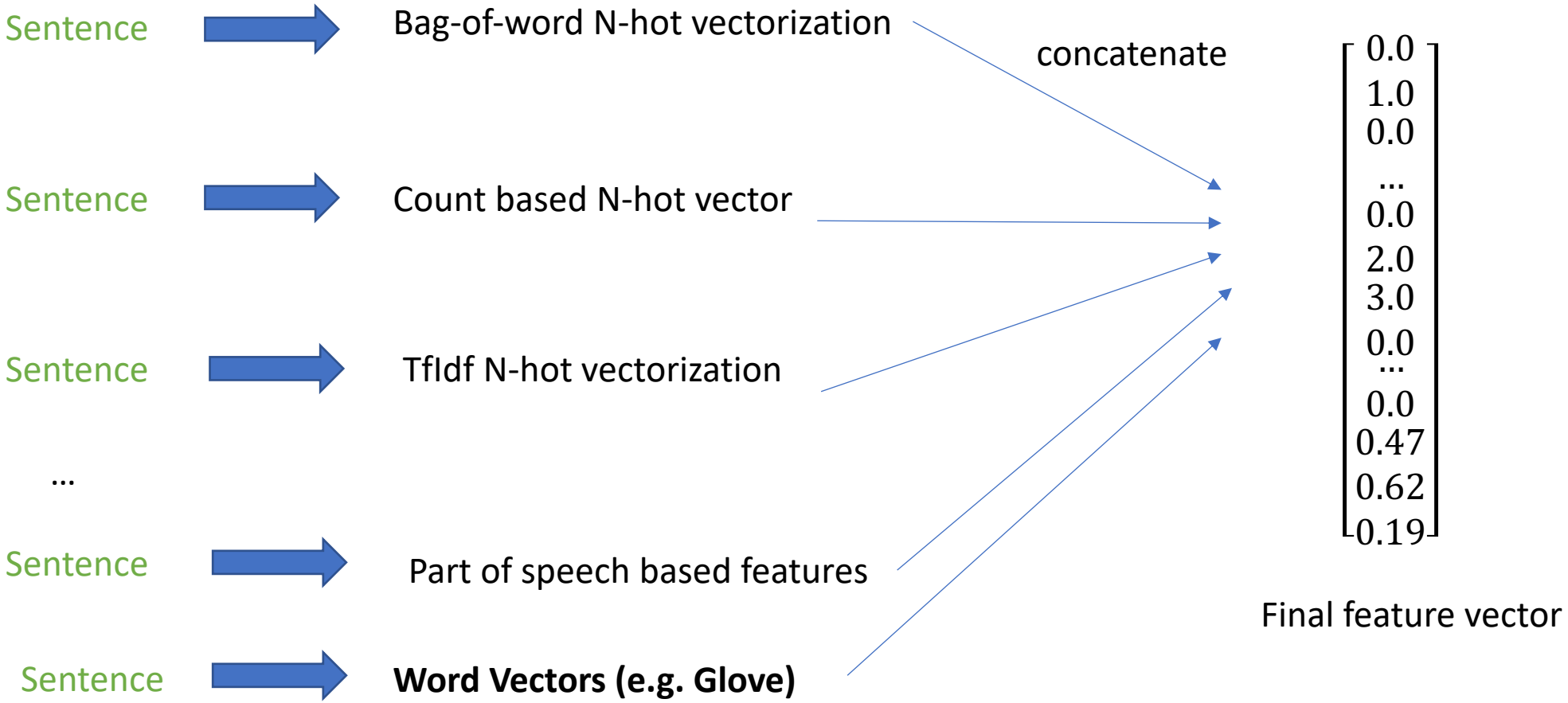
Recap: Feature Engineering

- Numeric feature: Use **as-is**
- String / Categorical Features:
  - Represent categories (e.g., academic_year)
  - Can be:
    - **Nominal**, i.e., not related to each other (e.g., city, country, university name)
    - **Ordinal**, i.e., certain order found between them (e.g., academic_year, letter grade)
- In NLP:
  - Convert strings into vectors using vectorization principles

Recap: Feature Engineering on Text

- **Objective**: Get fixed length vectors from variable length input
- **How** (we have done this in the past)**?**
  - **N-hot vectorization**
  - **TF-IDF vectorization**
  - **Averaged word embeddings (such as GloVE)**
  - **…**
- **We can also add linguistic features based on text processing**
  - E.g., POS / Dependency parse information

Recap: Feature Engineering (overall)



Final feature vector length should be same across all examples (both training and testing)

Recap: Data Splitting

- After transforming, we (randomly) split the data into:
  - *Training set* – used repeatedly for learning (**usually 80% of the data**)
  - *Validation set* – used for checking "goodness" of model intermittently to decide which direction should the training go into (**usually 10% of the data**)
  - *Test set* – used once to evaluate the final model (**usually 10%**)

Recap: Training

- For a set of M training examples, each containing N features , **minimize the average error on all examples**

$$\underset{f}{\text{minimize}} \quad \frac{1}{M}\sum_{\mathrm{i}=1}^{\mathrm{M}} \mathrm{Err}(\mathrm{y}^{\mathrm{i}}_{\mathrm{actual}}, \mathrm{y}^{\mathrm{i}}_{\mathrm{predicted}})$$

$$\Rightarrow \quad \underset{f}{\text{minimize}} \quad \frac{1}{M}\sum_{\mathrm{i}=1}^{\mathrm{M}} \mathrm{Err}(\mathrm{y}^{\mathrm{i}}_{\mathrm{actual}}, \mathrm{f}(x^i_1, x^i_2, \ldots, x^i_N))$$

Recap: What is Err( )?

- Mathematical measure that quantifies **how well a machine learning model's predictions match the actual (true) values of the data it's trying to learn from.**
- Often referred to as "**Loss**" function or "**Empirical Risk**" in ML
- Many possibilities
- Let's start with a simple (and elegant) error function

$$(y_{\text{predicted}} - y_{\text{actual}})^2$$

Recap: What is training – regression?

- For a set of M training examples, each containing N features

$$\underset{f}{\text{minimize}} \; \frac{1}{M} \sum_{\text{i}=1}^{\text{M}} (\text{y}_{\text{actual}}^{\text{i}} - \text{y}_{\text{predicted}}^{\text{i}})^2$$

Mean squared error

Parametric Function

$$\Rightarrow \; \underset{f}{\text{minimize}} \; \frac{1}{M} \sum_{\text{i}=1}^{\text{M}} (\text{y}_{\text{actual}}^{\text{i}} - \text{f}(x_1^i, x_2^i, \ldots, x_N^i))^2$$

Recap: Error for Classification

- For classification, we use a slightly different error (based on probability principles)
- **Cross Entropy Error Example (Suitable for Classificaiton**

$$Err = -\sum_{c \in C} y^c_{actual} . log\ y^c_{predicted}$$

Where C is a collection of all possible classes

Recap: What is training – classification?

- For a set of M training examples, each containing N features

$$\underset{f}{\text{minimize}} \quad -\frac{1}{M}\sum_{i=1}^{M}\sum_{c \in C} y_{actual}^{i,c} \cdot log\, y_{predicted}^{i,c}$$

$$\Rightarrow \quad -\frac{1}{M}\sum_{i=1}^{M}\sum_{c \in C} y_{actual}^{i,c} \cdot \log(\mathrm{f}^{\mathrm{c}}(x_1^i, x_2^i, \ldots, x_N^i))$$

Recap: What is f( )

- Can be any mathematical function
- BUT we restrict it to a certain class of functions
- Example: Logistic Regression
  - $\mathrm{f}^{\mathrm{c}}\left(x_1^i, x_2^i, \ldots, x_N^i\right) = \mathrm{p}(\mathrm{C} \mid x_1^i, x_2^i, \ldots, x_N^i)$

$$= \frac{1}{1 + e^{-(w_1 x_1^i + w_2 x_2^i + \cdots + w_N x_N^i + \beta)}}$$

Recap: Other model types

- **Support vector machines**
  - Modeling to draw margins that separate data
- **Decision Trees**
- **Feed Forward neural Networks**

Recap: What is Evaluation (Testing)?

- Evaluate the performance of a model on a test dataset
- **Classification**
  - **Accuracy:** *how many times predicted class is equal to the actual class in test dataset*
  - *Precision, Recall , F1 scores*
- **Regression**
  - Mean Squared Error
  - Mean Absolute Error
  - Correlation between predicted and actual values

Recap: Examples from Literature

Linguistic Features – Example: Sarcasm Detection (Joshi et al, 2015)

- Objective – classify short sentences as sarcastic / not

Lexical	
Unigrams	Unigrams in the training corpus
Pragmatic	
Capitalization	Numeric feature indicating presence of capital letters
Emoticons & laughter expressions	Numeric feature indicating presence of emoticons and 'lol's
Punctuation marks	Numeric feature indicating presence of punctuation marks
Implicit Incongruity	
Implicit Sentiment Phrases	Boolean feature indicating phrases extracted from the implicit phrase extraction step
Explicit Incongruity	
#Explicit incongruity	Number of times a word is followed by a word of opposite polarity
Largest positive /negative subsequence	Length of largest series of words with polarity unchanged
#Positive words	Number of positive words
#Negative words	Number of negative words
Lexical Polarity	Polarity of a tweet based on words present

Bag of words

Table 1: Features of our sarcasm detection system

Joshi, A., Sharma, V., & Bhattacharyya, P. (2015, July). Harnessing context incongruity for sarcasm detection. In Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers) (pp. 757-762).

Classification Results

- Showing feature importance through ablation studies
- **Ablation studies:** Remove one of more features at a time and repeat training and testing

Features	P	R	F
Original Algorithm by Riloff et al. (2013)			
Ordered	0.774	0.098	0.173
Unordered	0.799	0.337	0.474
Our system			
Lexical (**Baseline**)	0.820	0.867	0.842
Lexical+Implicit	0.822	0.887	0.853
Lexical+Explicit	0.807	0.985	0.8871
All features	0.814	0.976	**0.8876**

Classifier = SVM

Joshi, A., Sharma, V., & Bhattacharyya, P. (2015, July). Harnessing context incongruity for sarcasm detection. In Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers) (pp. 757-762).

Another Example: Readability Assessment of Healthcare Text

Objective – Predict readability score (0-100) of a document with the help of the following features

Feature category	Name
Raw Text	Average number of words per sentence Average number of characters per word
Lexical	Type/Token Ratio Lexical density *Basic Italian Vocabulary (BIV)* (De Mauro, 2000) rate
Morpho–syntactic	Part-Of-Speech unigrams Mood, tense and person of verbs
Syntactic	Distribution of dependency types Depth of the whole parse tree Average depth of embedded complement 'chains' Distribution of embedded complement 'chains' by depth Number of verbal roots Arity of verbal predicates Distribution of verbal predicates by arity Distribution of subordinate vs main clauses Relative ordering with respect to the main clause the Average depth of 'chains' of embedded subordinate clauses Distribution of embedded subordinate clauses 'chains' by depth Length of dependency links feature

Venturi, G., Bellandi, T., Dell'Orletta, F., & Montemagni, S. (2015, September). NLP–based readability assessment of health–related texts: a case study on Italian informed consent forms. In Proceedings of the Sixth International Workshop on Health Text Mining and Information Analysis (pp. 131-141).

Week 8: Roadmap

- **Structured Machine Learning basics**
  - How does a POS tagger work?
- **Unsupervised ML and Topic Modeling**
  - K-means clustering
  - Topic modeling using Latent Dirichlet Allocation

Structured Machine Learning

“Structured prediction or structured (output) machine learning is an umbrella term for supervised machine learning techniques that **involves predicting structured objects, rather than scalar discrete or real values**”

Examples: Sequence Tagging

- For an input *sentence X*, predict Y which is a set of possible Tags
- POS Tagging

This	DT
is	VBZ
a	DT
tagged	JJ
sentence	NN

$$X = \{x_1, x_2 \ \ldots, x_N\} \textbf{ or } \mathbf{X} \ \in \ \mathbb{R}^{\boldsymbol{N}}$$

$$Y = \{y_1, y_2 \ \ldots, y_N\} \textbf{ or } \mathbf{Y} \ \in \ \mathbb{R}^{\boldsymbol{N}}$$

- Named Entity Recognition

Examples: Parsing and Translation

- **Parsing:** For a sentence X, Y is a set of parse trees
- **Machine Translation:** For a sentence X in a source language, Y is another sequence in another language

$$X = \{x_1, x_2 \ \ldots, x_N\} \textbf{ or } \mathbf{X} \in \mathbb{R}^{\boldsymbol{N}}$$

$$Y = \{y_1, y_2 \ \ldots, y_M\} \textbf{ or } \mathbf{Y} \in \mathbb{R}^{\boldsymbol{M}}$$

- M may or may not be equal to N

Let’s dive into Sequence Labeling Problem

- Consider the example of POS tagging

  Sentence 1: The bear chased the bear.

  Sentence 2: She will bear the burden.

  Sentence 3: They bear witness to the bear's presence.

- Training Data (3 labeled sentences)

  1. The_DT bear_NN chased_VBD the_DT bear_NN
  2. She_PRP will_MD bear_VB the_DT burden_NN
  3. They_PRP bear_VBP witness_NN to_TO the_DT bear_NN 's_POS presence_NN .

- Test Sentence (unseen):
  - She chased the bear

Let’s dive into Sequence Labeling Problem

- We can always add beginning and end tokens and dummy tags.
- Training Data Now

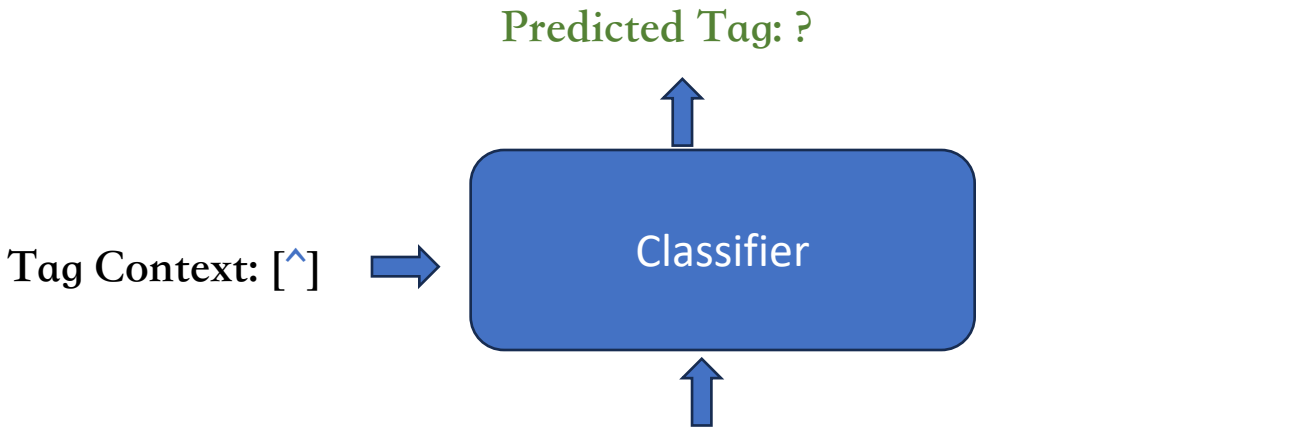
1. ^_^ The_DT bear_NN chased_VBD the_DT bear_NN $_$
2. ^_^ She_PRP will_MD bear_VB the_DT burden_NN $_$
3. ^_^ They_PRP bear_VBP witness_NN to_TO the_DT bear_NN 's_POS presence_NN $_$

At Runtime (testing)

- At runtime:
  - Recurringly process one word at a time
  - Classify a word given contextual cues from the word and surrounding words and tags available so far

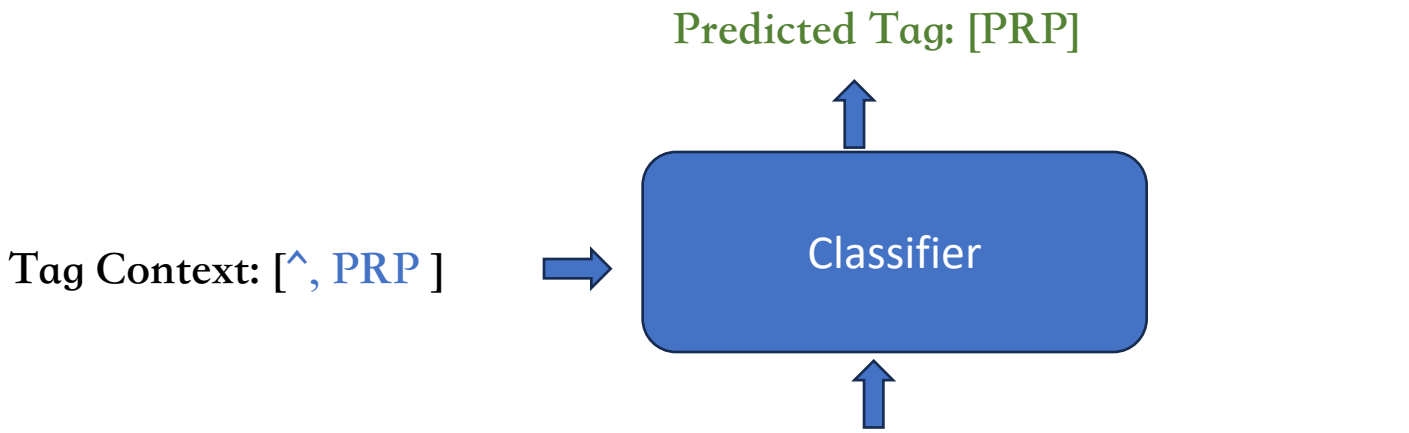
At Runtime (testing) – Step 1

“^_^ She chased the bear $”



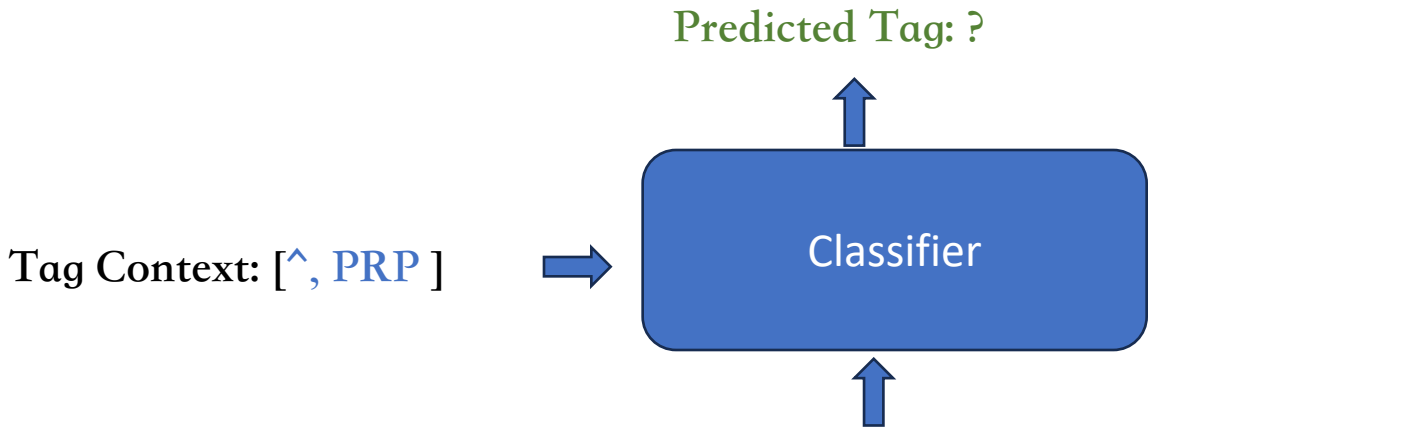
At Runtime (testing) – Step 2

"^_^ She_PRP chased the bear $"



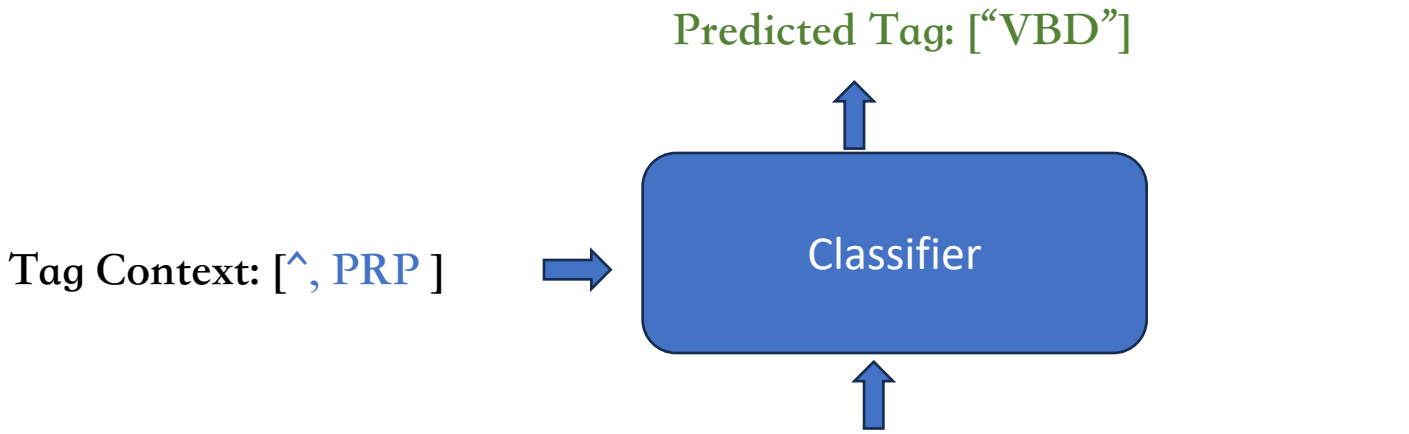
At Runtime (testing) – Step 2

"^_^ She_PRP chased the bear $"



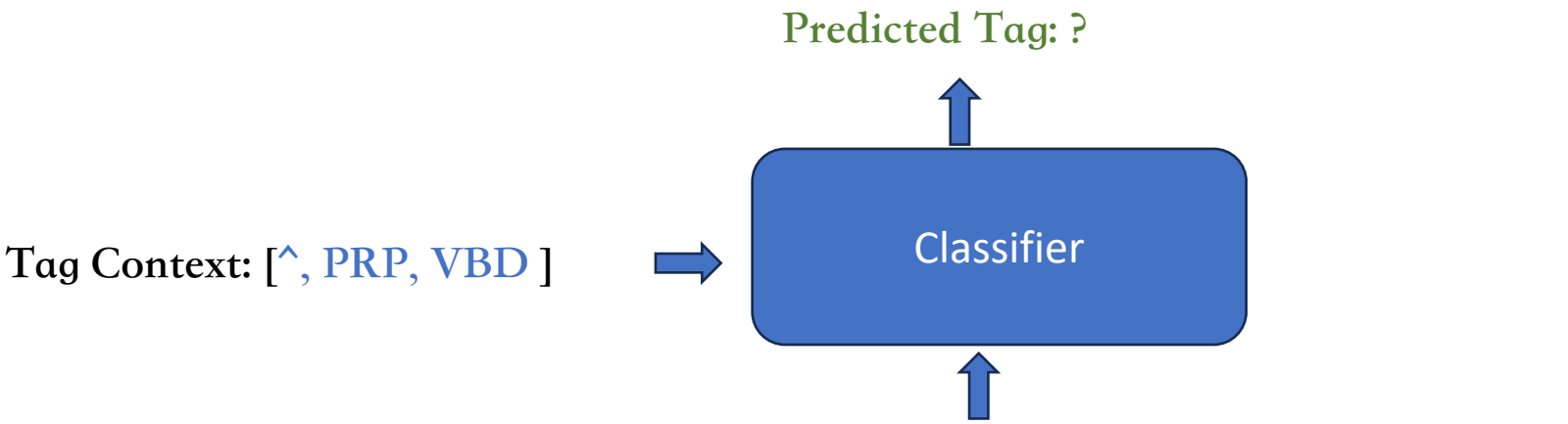
At Runtime (testing) – Step 2

"^_^ She_PRP chased_VBD the bear $"



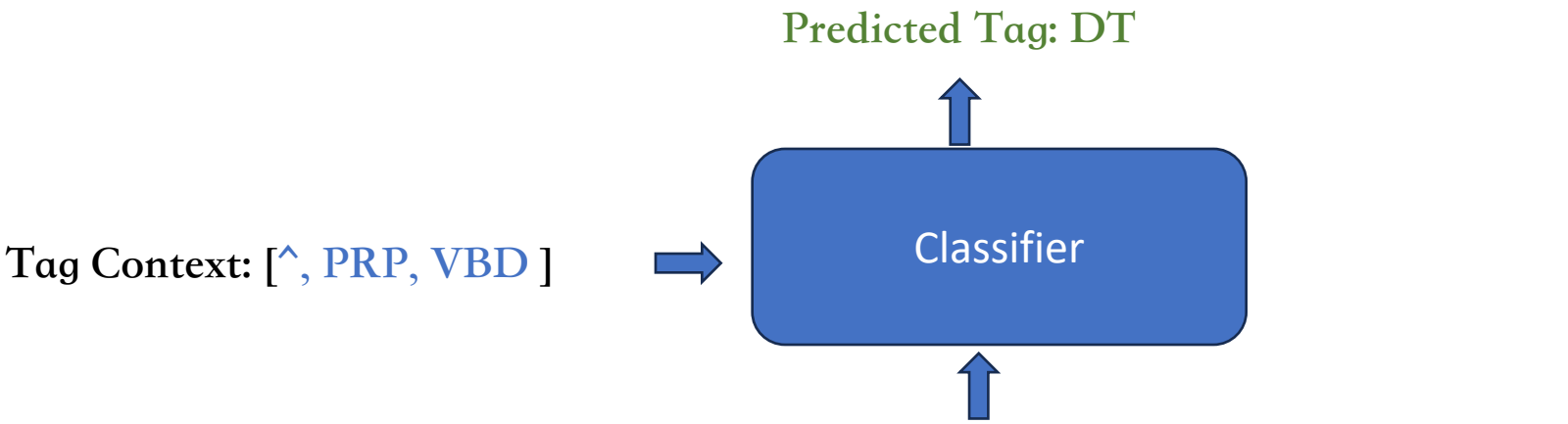
At Runtime (testing) – Step 2

"^_^ She_PRP chased_VBD the bear $"



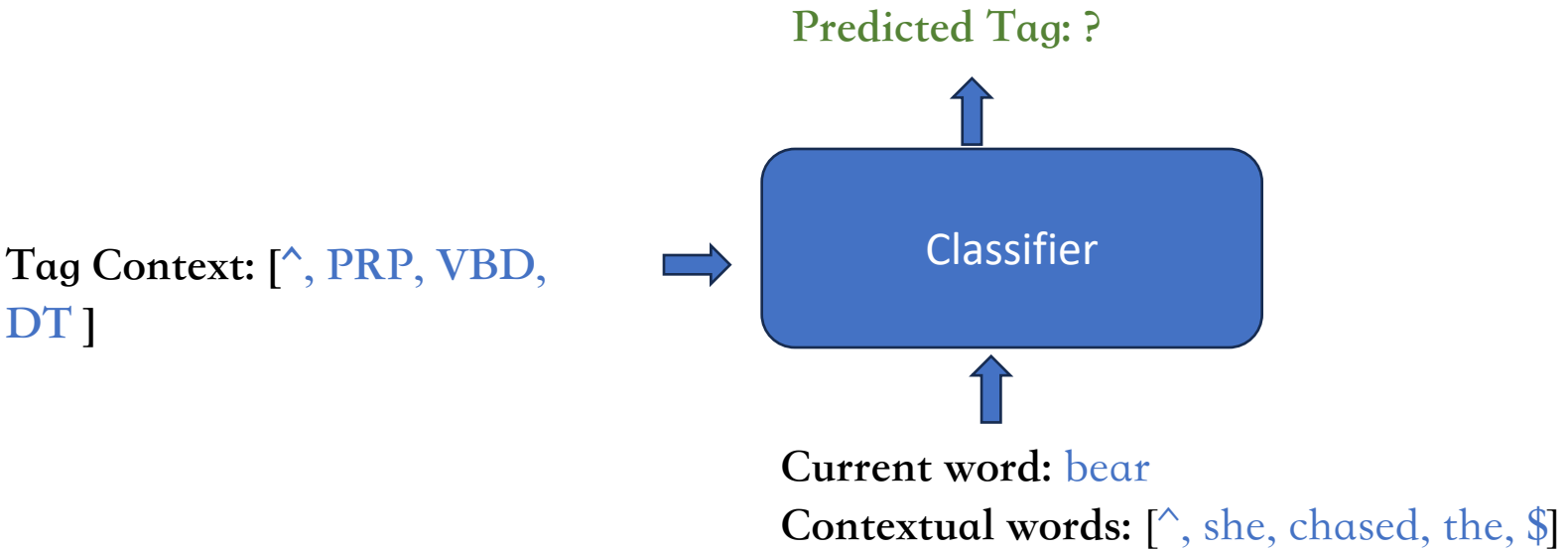
At Runtime (testing) – Step 2

"^_^ She_PRP chased_VBD the_DT bear $"



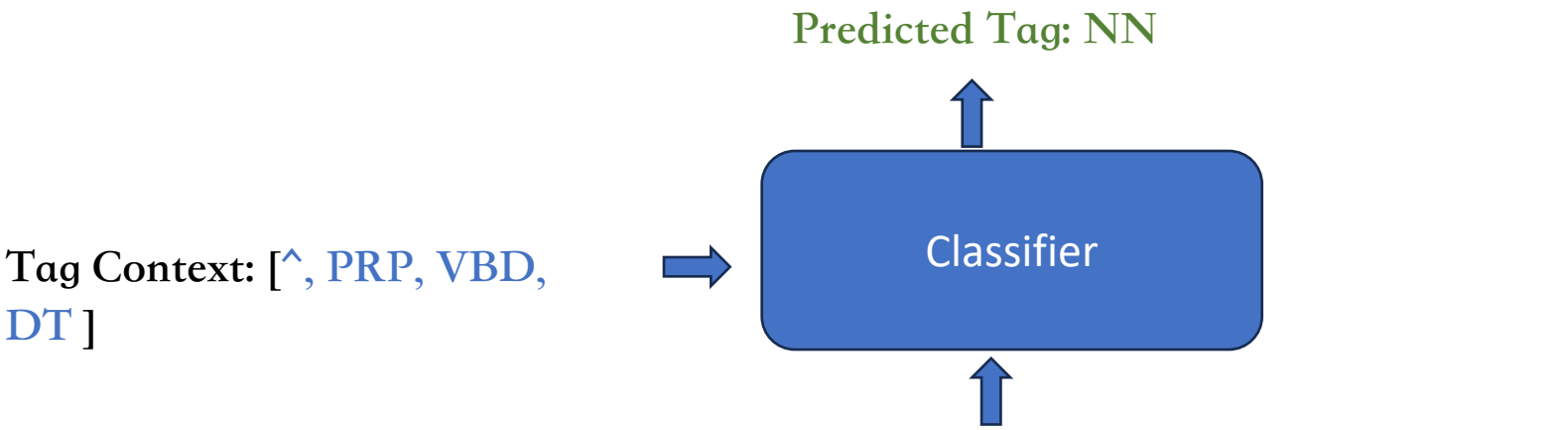
At Runtime (testing) – Step 2

"^_^ She_PRP chased_VBD the_DT bear $"



At Runtime (testing) – Step 2

"^_^ She_PRP chased_VBD the_DT bear_NN $"



At Runtime (testing) – Step 2

Final output:

“^_^ She_PRP chased_VBD the_DT bear_NN $_$”

Algorithmically

- For a given sequence of tokens $X = \{x_1, x_2 \ \ldots, x_N\}$
- Find out the most probable tag sequence $\mathrm{T} = \{t_1, t_2 \ \ldots, t_N\}$

for i in $\{0,1,2\cdots, \mathrm{N}\text{-}1\}$

$$\mathrm{t}_\mathrm{i}^* = \mathrm{argmax}_\mathrm{t}\ \mathrm{p}(\mathrm{t}_\mathrm{i} \mid \mathrm{x}_1, \mathrm{x}_2, \ldots, \mathrm{x}_\mathrm{i}, \mathrm{t}_1, \mathrm{t}_2, \mathrm{t}_3, \ldots, \mathrm{t}_{\mathrm{i}-1})$$

t_i^* → Most probabe tag

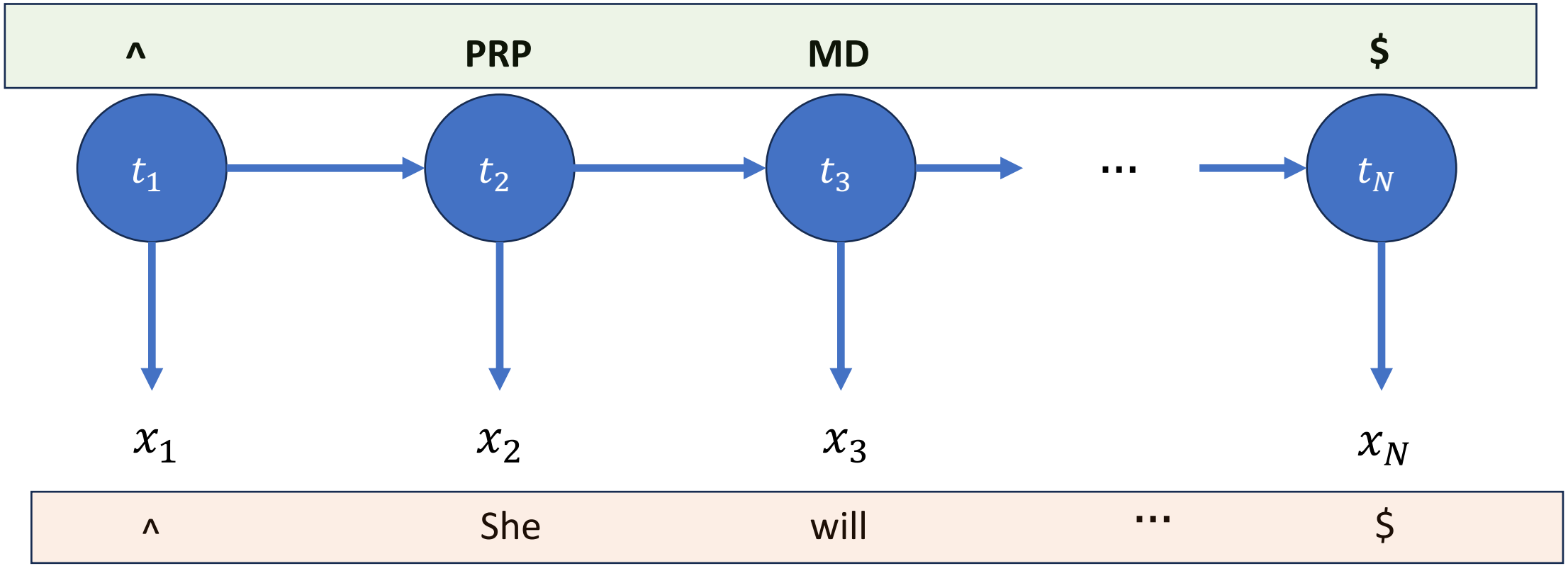
Estimating $p(t_i | x_1, x_2, \ldots, x_i, t_1, t_2, t_3, \ldots, t_{i-1})$

- Estimating the probability of an event given context is vague and quite hard from data
- Many simplified forms:
  - Hidden Markov Model (HMM)
  - Conditional Random Field (CRF)
  - Maximum Entropy Markov Model (MEMM)
- Modern Solutions:
  - Recurrent Neural Networks
  - Transformer Based Neural Nets

Hidden Markov Model

Example: She will bear the burden

What is hidden? : tags



What is observed? : sentence

HMMs for Estimating $p(t \mid x_1, x_2, \ldots, x_i, t_1, t_2, t_3, \ldots, t_{i-1})$

Simply applying Bayes Rule

$$t_i^* = \text{argmax}_t \; p(x_1, x_2, x_3, \ldots x_i | t_i) \; p(t_i | t_1, t_2, \ldots, t_{i-1})$$
$$= \text{argmax}_t \; p\,(x_i | t_i) \; p(t_i | t_{i-1})$$

Here we assumed:

1. A word is determined completely by its tag (lexical assumption)
2. Tag in present is determined by the tag in immediate past

Training HMMs

- Involves computing $p\,(x_i|t_i)$ and $p(t_i|t_{i-1})$ for all possible combinations

$$p\,(x_i|t_i) = \frac{\text{number of times word } w_i \text{ appeared with tag } t_i}{\text{Total number of times word all words appeared with tag } t_i}$$

$$p\,(t_i|t_{i-1}) = \frac{\text{number of times tag } t_i \text{ appeared with tag } t_{i-1}}{\text{Total number of times tag } t_i \text{ appeared with } \textit{all other tags}}$$

Training HMMs

Computing $p(x_i|t_i)$ and $p(t_i|t_{i-1})$ for our small training data

```
Transition Probabilities:
^ -> DT: 0.3333
^ -> PRP: 0.6667
DT -> NN: 1.0000
NN -> VBD: 0.1667
NN -> $: 0.5000
NN -> TO: 0.1667
NN -> POS: 0.1667
VBD -> DT: 1.0000
PRP -> MD: 0.5000
PRP -> VBP: 0.5000
MD -> VB: 1.0000
VB -> DT: 1.0000
VBP -> NN: 1.0000
TO -> DT: 1.0000
POS -> NN: 1.0000
```

```
Emission Probabilities:
^ emits ^: 1.0000
DT emits The: 0.2500
DT emits the: 0.7500
NN emits bear: 0.5000
NN emits burden: 0.1667
NN emits witness: 0.1667
NN emits presence: 0.1667
VBD emits chased: 1.0000
$ emits $: 1.0000
PRP emits She: 0.5000
PRP emits They: 0.5000
MD emits will: 1.0000
VB emits bear: 1.0000
VBP emits bear: 1.0000
TO emits to: 1.0000
POS emits 's: 1.0000
```

Inference

- Once the probability values are estimated, for any given unseen sentence, estimate the most likelihood class one at a time and use the outcome as contextual tag for next word

$$\text{for i in } \{0,1,2\cdots, \text{N-1}\}$$
$$t_i^* = \text{argmax}_t\ p\ (x_i|t_i)\ p(t_i|t_{i-1})$$

- This process is also known as **decoding**

Inference in HMM

- Test example: “She will bear the burden”
- After adding start and end characters: “^ She will bear the burden $”
- All possible tags:

```
{'$', 'MD', '^', 'VBD', 'POS', 'NN', 'TO', 'VBP', 'DT', 'PRP', 'VB'}
```

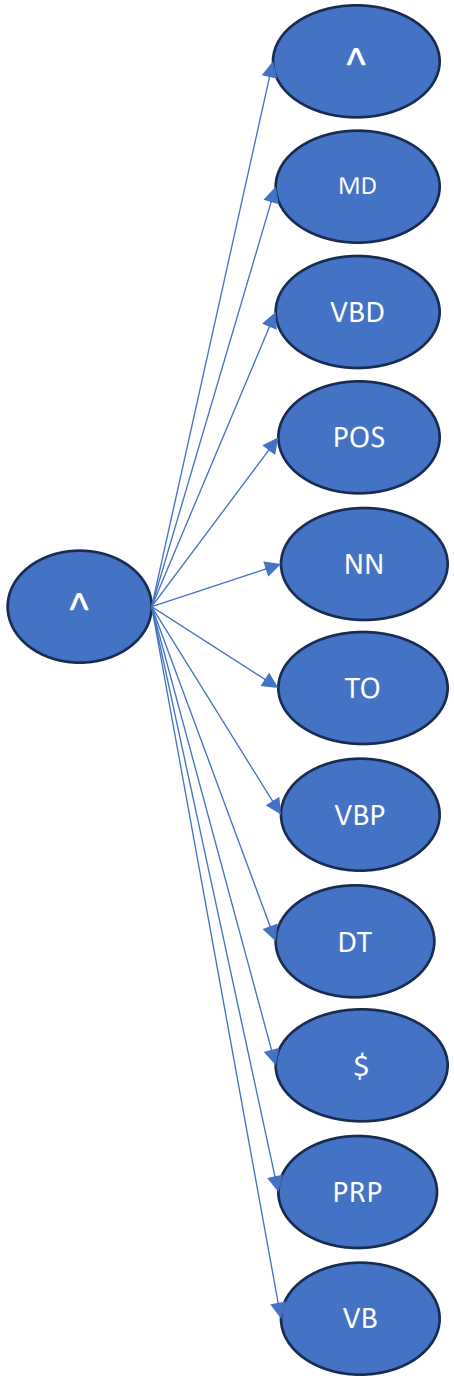
- Decoding happens using **Viterbi algorithm**

Word = “^”

$p(\text{“^”}|\text{“^”}) = 1.0$

Word = “^”

Word = “She”



$p(\text{“She”} \mid \text{“^”}) * p(\text{“^”} \mid \text{“^”}) = 0$

$p(\text{“She”} \mid \text{“MD”}) * p(\text{“MD”} \mid \text{“^”}) = 0$

$p(\text{“She”} \mid \text{“VBD”}) * p(\text{“VBD”} \mid \text{“^”}) = 0$

$p(\text{“She”} \mid \text{“POS”}) * p(\text{“POS”} \mid \text{“^”}) = 0$

$p(\text{“She”} \mid \text{“NN”}) * p(\text{“NN”} \mid \text{“^”}) = 0$

$p(\text{“She”} \mid \text{“TO”}) * p(\text{“TO”} \mid \text{“^”}) = 0$

$p(\text{“She”} \mid \text{“VBP”}) * p(\text{“VBP”} \mid \text{“^”}) = 0$

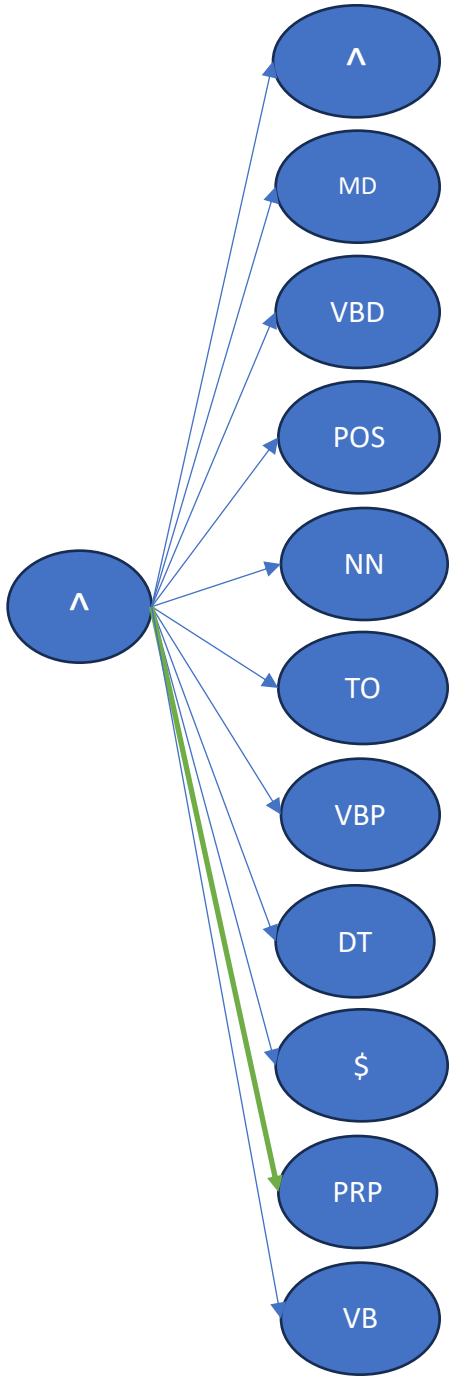
$p(\text{“She”} \mid \text{“DT”}) * p(\text{“DT”} \mid \text{“^”}) = 0$

$p(\text{“She”} \mid \text{“\$”}) * p(\text{“\$”} \mid \text{“^”}) = 0$

$p(\text{“She”} \mid \text{“PRP”}) * p(\text{“PRP”} \mid \text{“^”}) = 0.5 * 0.667$

$p(\text{“She”} \mid \text{“VB”}) * p(\text{“VB”} \mid \text{“^”}) = 0$

Word = “She”



$p(\text{“She”} \mid \text{“^”}) * p(\text{“^”} \mid \text{“^”}) = 0$

$p(\text{“She”} \mid \text{“MD”}) * p(\text{“MD”} \mid \text{“^”}) = 0$

$p(\text{“She”} \mid \text{“VBD”}) * p(\text{“VBD”} \mid \text{“^”}) = 0$

$p(\text{“She”} \mid \text{“POS”}) * p(\text{“POS”} \mid \text{“^”}) = 0$

$p(\text{“She”} \mid \text{“NN”}) * p(\text{“NN”} \mid \text{“^”}) = 0$

$p(\text{“She”} \mid \text{“TO”}) * p(\text{“TO”} \mid \text{“^”}) = 0$

$p(\text{“She”} \mid \text{“VBP”}) * p(\text{“VBP”} \mid \text{“^”}) = 0$

$p(\text{“She”} \mid \text{“DT”}) * p(\text{“DT”} \mid \text{“^”}) = 0$

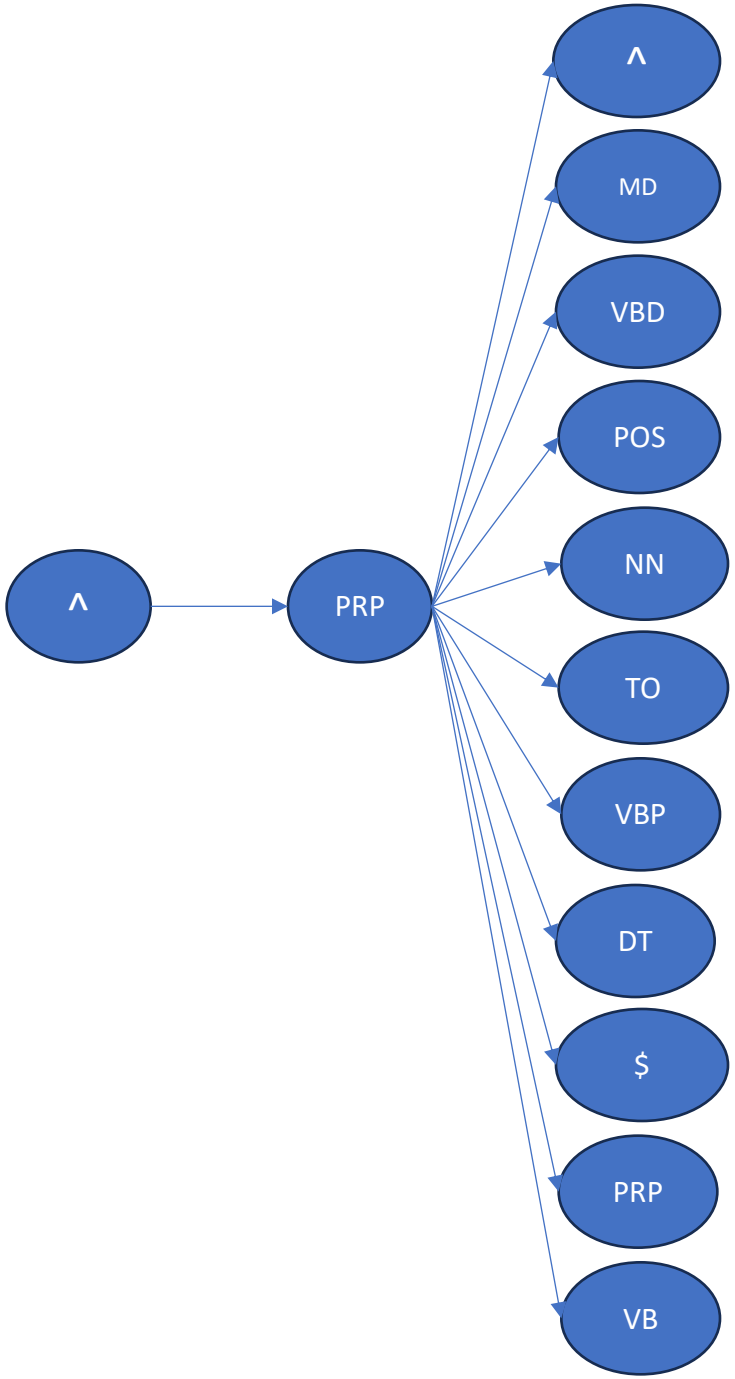
$p(\text{“She”} \mid \text{“\$”}) * p(\text{“\$”} \mid \text{“^”}) = 0$

$p(\text{“She”} \mid \text{“PRP”}) * p(\text{“PRP”} \mid \text{“^”}) = 0.5 * 0.667$

$p(\text{“She”} \mid \text{“VB”}) * p(\text{“VB”} \mid \text{“^”}) = 0$

Maximizes

Word = “will”



$p(\text{“will”} \mid \text{“^”}) * p(\text{“^”} \mid \text{“PRP”}) = 0$

$p(\text{“will”} \mid \text{“MD”}) * p(\text{“MD”} \mid \text{“PRP”}) = 1.0 * 0.5$

$p(\text{“will”} \mid \text{“VBD”}) * p(\text{“VBD”} \mid \text{“PRP”}) = 0$

$p(\text{“will”} \mid \text{“POS”}) * p(\text{“POS”} \mid \text{“PRP”}) = 0$

$p(\text{“will”} \mid \text{“NN”}) * p(\text{“NN”} \mid \text{“PRP”}) = 0$

$p(\text{“will”} \mid \text{“TO”}) * p(\text{“TO”} \mid \text{“PRP”}) = 0$

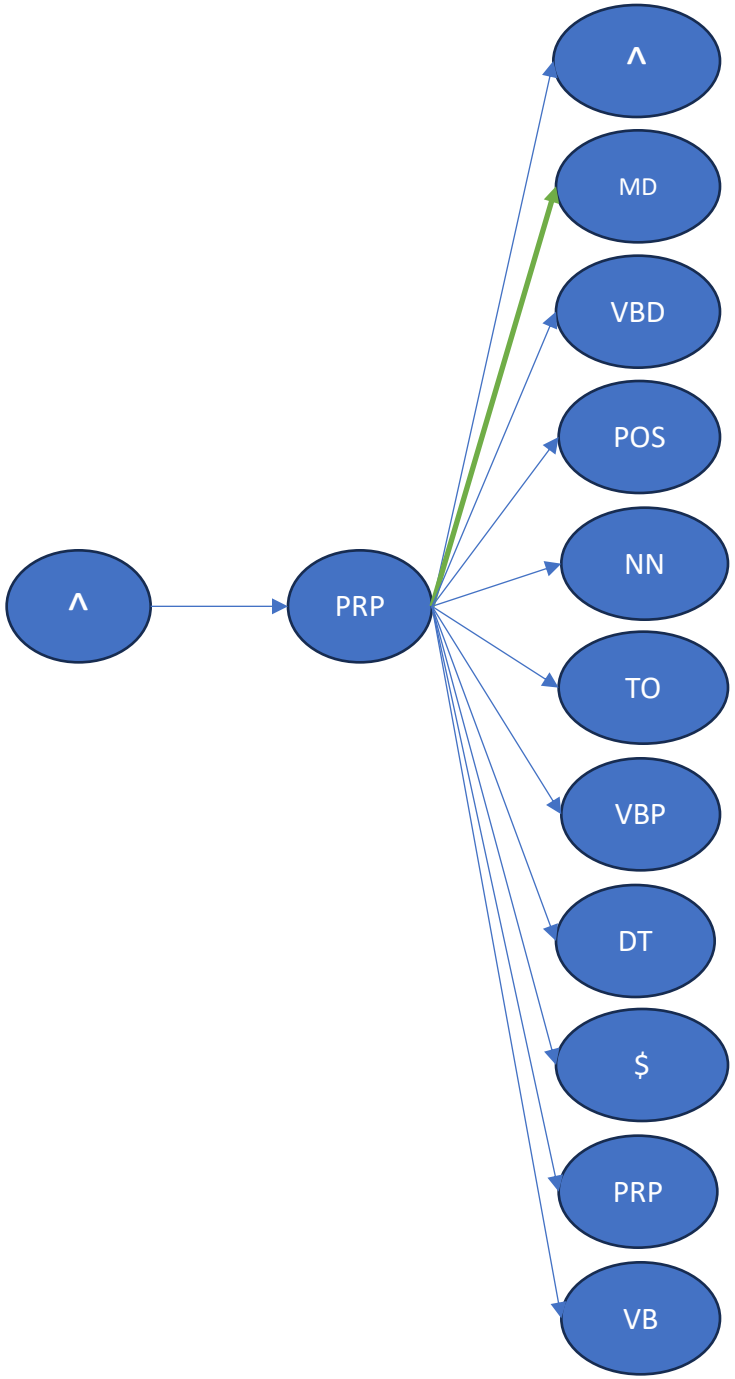
$p(\text{“will”} \mid \text{“VBP”}) * p(\text{“VBP”} \mid \text{“PRP”}) = 0$

$p(\text{“will”} \mid \text{“DT”}) * p(\text{“DT”} \mid \text{“PRP”}) = 0$

$p(\text{“will”} \mid \text{“\$”}) * p(\text{“\$”} \mid \text{“PRP”}) = 0$

$p(\text{“will”} \mid \text{“PRP”}) * p(\text{“PRP”} \mid \text{“PRP”}) =$

$p(\text{“will”} \mid \text{“VB”}) * p(\text{“VB”} \mid \text{“PRP”}) = 0$



$p(\text{“will”} \mid \text{“^”}) * p(\text{“^”} \mid \text{“PRP”}) = 0$

$p(\text{“will”} \mid \text{“MD”}) * p(\text{“MD”} \mid \text{“PRP”}) = 1.0 * 0.5$

$p(\text{“will”} \mid \text{“VBD”}) * p(\text{“VBD”} \mid \text{“PRP”}) = 0$

$p(\text{“will”} \mid \text{“POS”}) * p(\text{“POS”} \mid \text{“PRP”}) = 0$

$p(\text{“will”} \mid \text{“NN”}) * p(\text{“NN”} \mid \text{“PRP”}) = 0$

$p(\text{“will”} \mid \text{“TO”}) * p(\text{“TO”} \mid \text{“PRP”}) = 0$

$p(\text{“will”} \mid \text{“VBP”}) * p(\text{“VBP”} \mid \text{“PRP”}) = 0$

$p(\text{“will”} \mid \text{“DT”}) * p(\text{“DT”} \mid \text{“PRP”}) = 0$

$p(\text{“will”} \mid \text{“\$”}) * p(\text{“\$”} \mid \text{“PRP”}) = 0$

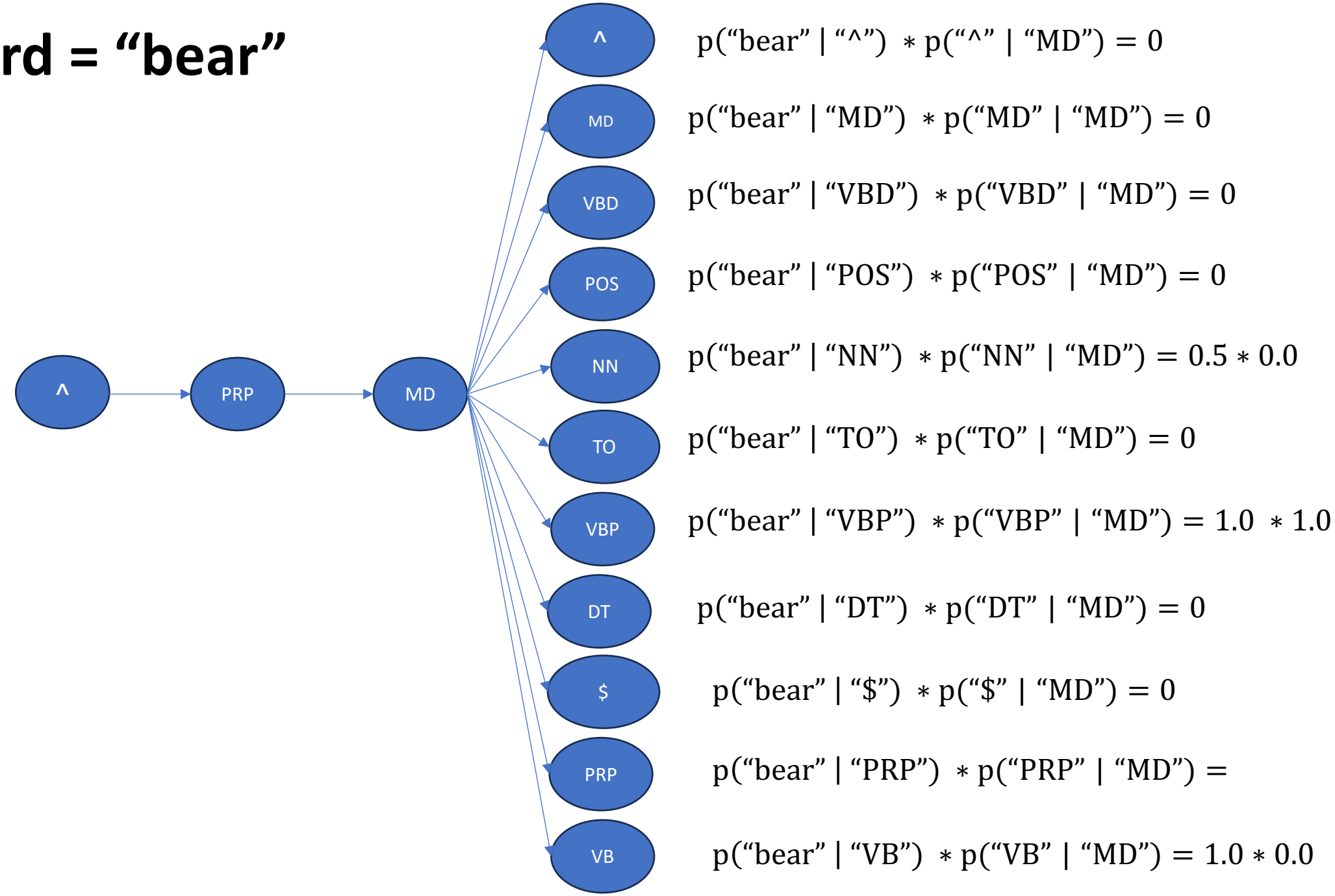
$p(\text{“will”} \mid \text{“PRP”}) * p(\text{“PRP”} \mid \text{“PRP”}) = 0$

$p(\text{“will”} \mid \text{“VB”}) * p(\text{“VB”} \mid \text{“PRP”}) = 0$

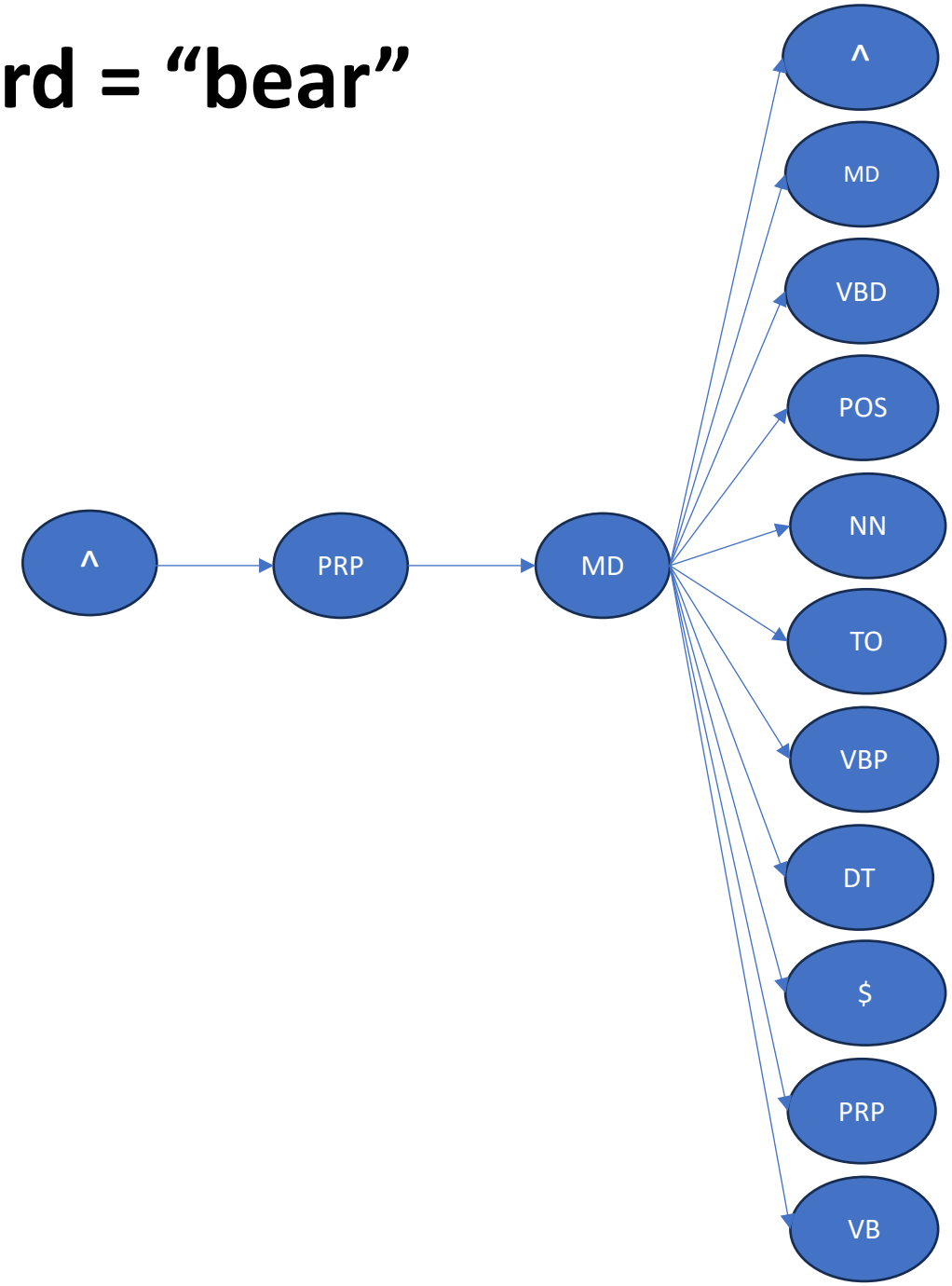
Word = “will”

Maximizes

Word = “bear”


^
PRP
MD
^
MD
VBD
POS
NN
TO
VBP
DT
$
PRP
VB
p(“bear” | “^”) ∗ p(“^” | “MD”) = 0
p(“bear” | “MD”) ∗ p(“MD” | “MD”) = 0
p(“bear” | “VBD”) ∗ p(“VBD” | “MD”) = 0
p(“bear” | “POS”) ∗ p(“POS” | “MD”) = 0
p(“bear” | “NN”) ∗ p(“NN” | “MD”) = 0.5 ∗ 0.0
p(“bear” | “TO”) ∗ p(“TO” | “MD”) = 0
p(“bear” | “VBP”) ∗ p(“VBP” | “MD”) = 1.0 ∗ 1.0
p(“bear” | “DT”) ∗ p(“DT” | “MD”) = 0
p(“bear” | “$”) ∗ p(“$” | “MD”) = 0
p(“bear” | “PRP”) ∗ p(“PRP” | “MD”) =
p(“bear” | “VB”) ∗ p(“VB” | “MD”) = 1.0 ∗ 0.0

Word = “bear”



$p(\text{“bear”} \mid \text{“^”}) * p(\text{“^”} \mid \text{“MD”}) = 0$

$p(\text{“bear”} \mid \text{“MD”}) * p(\text{“MD”} \mid \text{“MD”}) = 0$

$p(\text{“bear”} \mid \text{“VBD”}) * p(\text{“VBD”} \mid \text{“MD”}) = 0$

$p(\text{“bear”} \mid \text{“POS”}) * p(\text{“POS”} \mid \text{“MD”}) = 0$

$p(\text{“bear”} \mid \text{“NN”}) * p(\text{“NN”} \mid \text{“MD”}) = 0.5 * 0.0$

$p(\text{“bear”} \mid \text{“TO”}) * p(\text{“TO”} \mid \text{“MD”}) = 0$

Maximizes

$p(\text{“bear”} \mid \text{“VBP”}) * p(\text{“VBP”} \mid \text{“MD”}) = 1.0 * 1.0$

$p(\text{“bear”} \mid \text{“DT”}) * p(\text{“DT”} \mid \text{“MD”}) = 0$

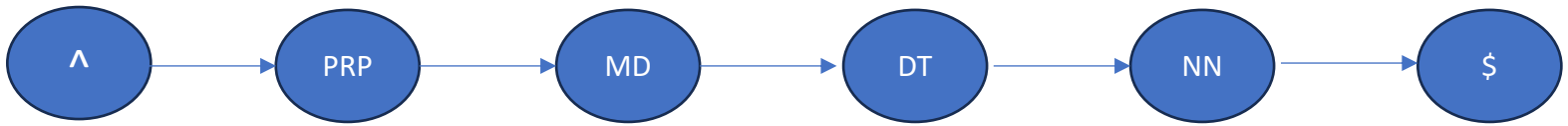
$p(\text{“bear”} \mid \text{“\$”}) * p(\text{“\$”} \mid \text{“MD”}) = 0$

$p(\text{“bear”} \mid \text{“PRP”}) * p(\text{“PRP”} \mid \text{“MD”}) =$

$p(\text{“bear”} \mid \text{“VB”}) * p(\text{“VB”} \mid \text{“MD”}) = 1.0 * 0.0$

...

Word = “$”



$$\mathbf{p}(\text{“\$”}|\text{”NN”}) = \mathbf{0.5}$$

HMM Extended

- We can extend the markov process to model second order dependencies
  - A tag is dependent on the previous two tags
- Also known as “trigram assumption”

$$\text{for i in } \{0,1,2\cdots, \text{N-1}\}$$

$$t_i^* = \text{argmax}_t \; p\,(x_i|t_i)\; p(t_i|t_{i-1}, t_{i-2})$$

Advantages of HMMs

- Simple formulation
  - Offers high accuracy for multiclass sequence labeling
- Very intuitive and interpretable
  - You know what these probability values mean
- Super fast:
  - Viterbi algorithm says “DON’T COMPUTE all SEQUENCES” → $O(T^S)$
  - INSTEAD, compute and retain optimal sequences → $O(T * S)$

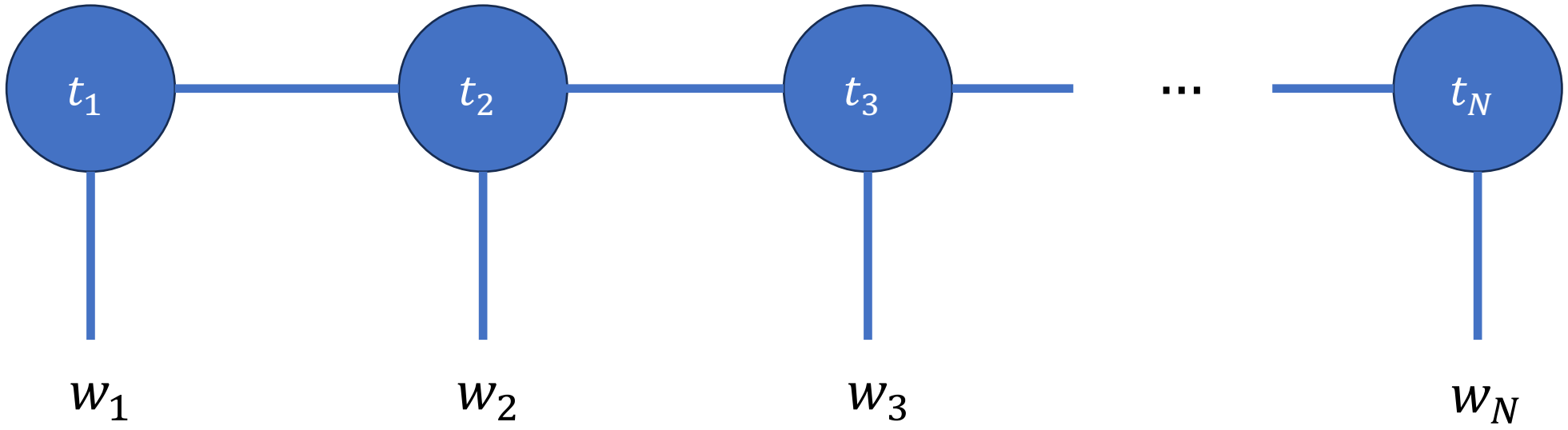
Disadvantages of HMMs

- Limited Modeling of Dependencies
- Fixed State Space
  - How about unknown tags?
- Inflexibility in Handling Continuous Data
- Assumption of Stationarity
  - Assumes that statistical properties of the system does not change over time
- Difficulty in Handling Missing Data
- Difficulty in Handling Handcrafted Features / External Domain Knowledge

Questions

Other ways to estimate

$p(t_i \mid x_1, x_2, \ldots, x_i, t_1, t_2, t_3, \ldots, t_{i-1}) \Rightarrow$ **CRF**



$$p(t_i \mid x_1, x_2, \ldots, x_i, t_1, t_2, t_3, \ldots, t_{i-1}) \approx p(t_i \mid t_{i-1}, w_1, w_2, \ldots, w_{t-1})$$
$$= \frac{1}{Z} \exp\left(\theta_1 f_1(t_i, t_i - 1) + \theta_2 f_2(t_i, w_1, w_2, \ldots w_N)\right)$$

f1, f2 are features . Thetas are weights and Z is a normalization constant

Also Known as "Conditional random field" formulation

No Arrows / Directionality:
Each tag is dependent on all of it's neighborhood

Conditional Random Fields

Remember, ? we are estimating this $p(t_i | x_1, x_2, \ldots, x_i, t_1, t_2, t_3, \ldots, t_{i-1})$

$$p(\mathbf{y}|\mathbf{x}) = \underbrace{\frac{1}{Z(\mathbf{x})}}_{\text{Normalization}} \prod_{t=1}^{T} \exp \left\{ \sum_{k=1}^{K} \underbrace{\theta_k}_{\text{Weight}} \underbrace{f_k(y_t, y_{t-1}, \mathbf{x}_t)}_{\text{Feature}} \right\}$$

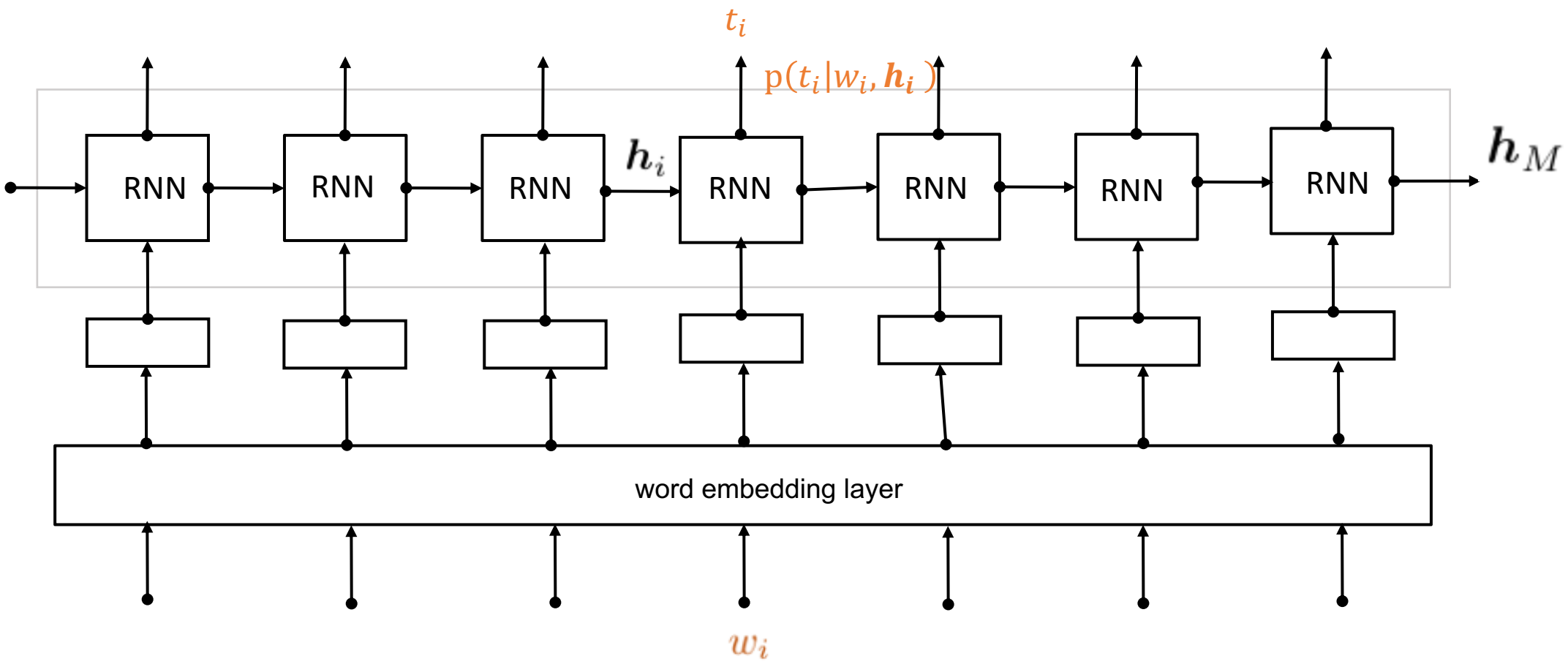
https://www.davidsbatista.net/blog/2017/11/13/Conditional_Random_Fields/

Features

- Handcrafted
- We can use transition and emission probabilities as features
- We can also define linguistic features:
  - E.g., ***“Does an input word end with the suffix “ed”””***
  - *NER example: “Is the previous POS tag “NNP” and the word ended with a suffix “tion” and the previous NER tag is “I_LOC”*

Other ways to estimate

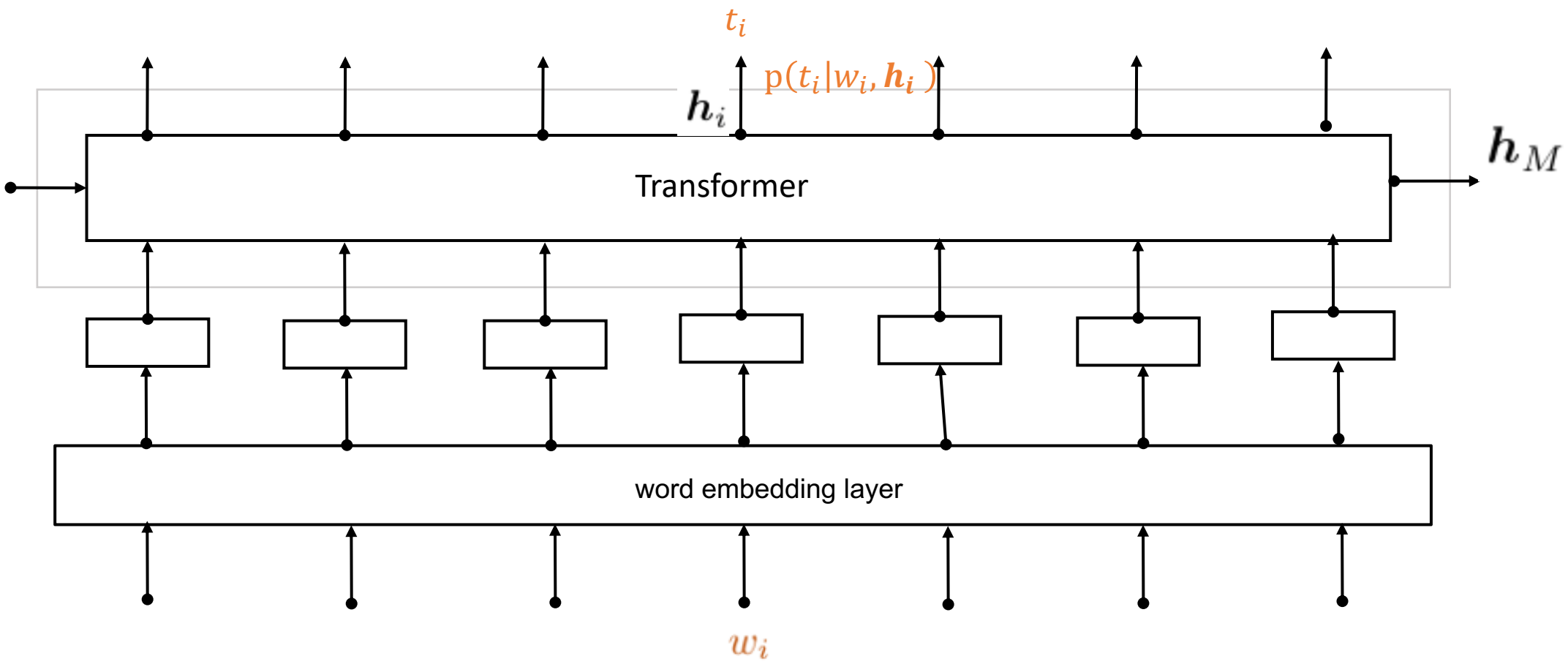
$p(t_i \mid x_1, x_2, \ldots, x_i, t_1, t_2, t_3, \ldots, t_{i-1})$ ⇒ **RNNs**



$h_i \rightarrow$ contextual *vector* representations from previous words

Other ways to estimate

$p(t_i | x_1, x_2, \ldots, x_i, t_1, t_2, t_3, \ldots, t_{i-1})$ ⇒ **Transformers**



$h_i \rightarrow$ contextual *vector* representations from **previous** and **future** words

Evaluation of Taggers

- Often reported through class wise accuracy on test data

Evaluation

- Confusion Matrix

	AJ0	AJ0-AV0	AJ0-NN1	AJ0-VVD	AJ0-VVG	AJ0-VVN	AJC	AJS	AT0	AV0	AV0-AJ0	AVP
AJ0	2899	20	32	1	3	3	0	0	18	35	27	1
AJ0-AV0	31	18	2	0	0	0	0	0	0	1	15	0
AJ0-NN1	161	0	116	0	0	0	0	0	0	0	1	0
AJ0-VVD	7	0	0	0	0	0	0	0	0	0	0	0
AJ0-VVG	8	0	0	0	2	0	0	0	1	0	0	0
AJ0-VVN	8	0	0	3	0	2	0	0	1	0	0	0
AJC	2	0	0	0	0	0	69	0	0	11	0	0
AJS	6	0	0	0	0	0	0	38	0	2	0	0
AT0	192	0	0	0	0	0	0	0	7000	13	0	0
AV0	120	8	2	0	0	0	15	2	24	2444	29	11
AV0-AJ0	10	7	0	0	0	0	0	0	0	16	33	0
AVP	24	0	0	0	0	0	0	0	1	11	0	737

Questions?

Unsupervised ML for NLP

Unsupervised Learning

- In supervised learning, we have a target variable that we are trying to predict, and use labels to construct a model
- In unsupervised learning,
  - we have no labels
  - our goals are less concrete

What is unsupervised learning used for?

- Discover groupings within our data - **Clustering**
- Find similar examples to a given example - **Similarity Detection**
- Find unusual examples within our data - **Anomaly Detection**

How similar / dissimilar are the vectors?

- Intuition: Farther points are “dissimilar” in nature
- Distance between two N-dimenstional points explains (dis)similarity
- Why similarity?
  - Quantifies implicit relationship between data points
  - E.g.,
    - Two “similar” students should receive similar attention in teaching
  - Very useful for search, trend analysis *etc.*

How similar / dissimilar are the vectors?

- Given two N-dimensional vectors (X^1, X^2)
  - $X^1 = [x_1^1, x_2^1, x_3^1, \ldots, x_N^1]$
  - $X^2 = [x_1^2, x_2^2, x_3^2, \ldots, x_N^2]$

- Distance between two vectors can be computed using
  - Euclidean Distance
  - Cosine Distance

Distance functions

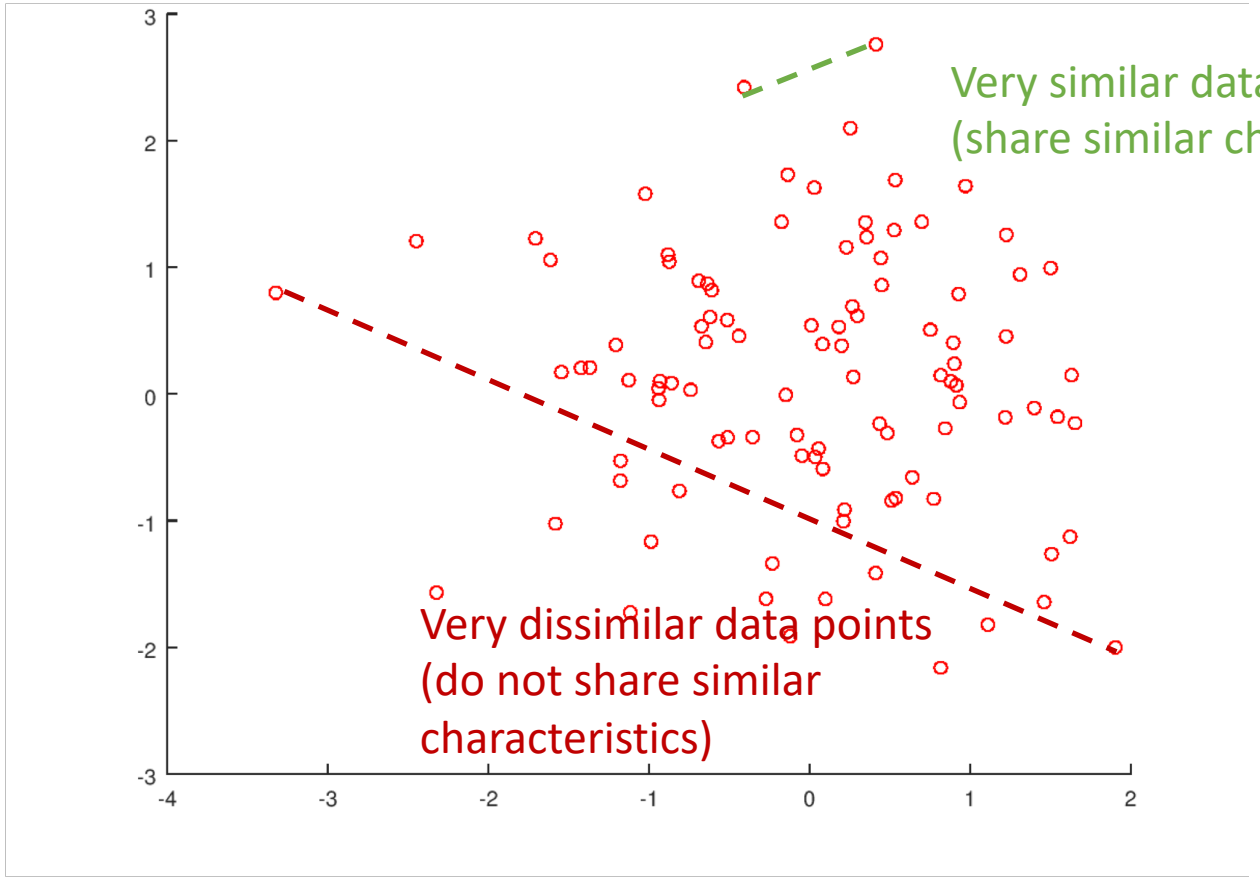
- Euclidean Distance between two N-dimensional feature vectors X^1, X^2

$$d(X^1, X^2) = \sqrt{\sum_{i=1}^{N} (x_i^1 - x_i^2)^2}$$

- Cosine Distance

$$\text{cosine}\ (X^{1``}\ X^2) = \ 1 - \frac{\sum_{i=1}^{N} x_i^1 x_i^2}{\sqrt{\sum_{i=1}^{N} (x_i^1)^2} \cdot \sqrt{\sum_{i=1}^{N} (x_i^2)^2}}$$

What does distance mean (again)?

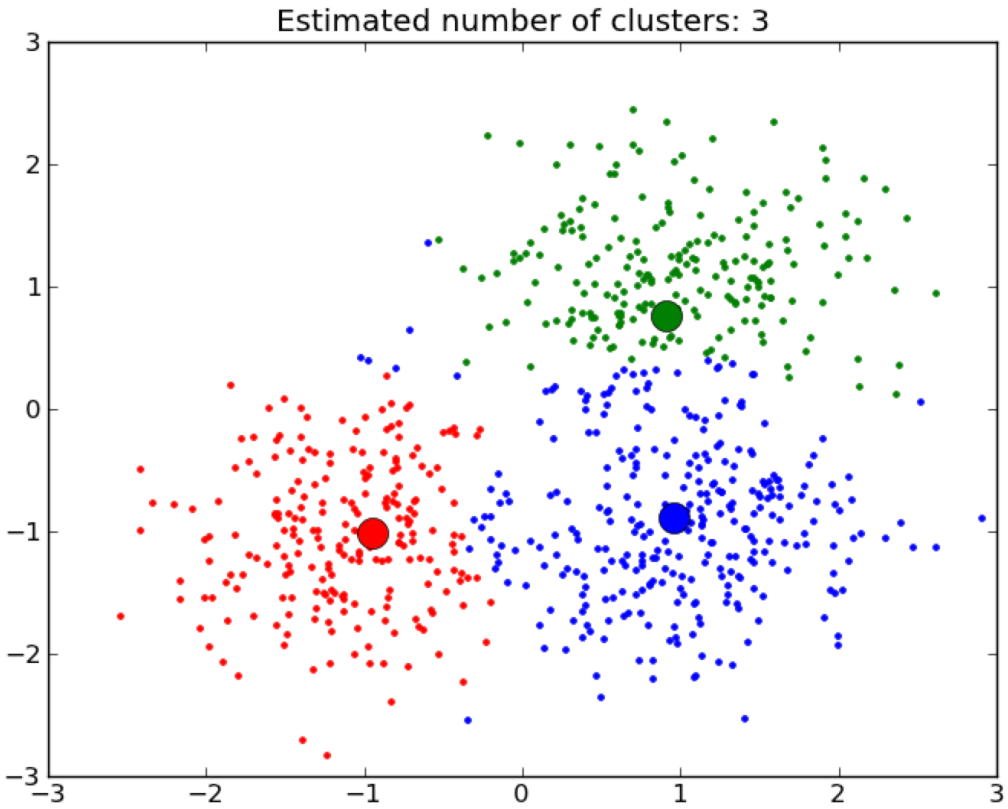


Time payment was overdue

Loan Size

Distance / Similarity is the key in Unsupervised Methods

- Using distance to define “clusters”



Clustering

- Divide data into K-clusters
- Some algorithms:
  - K-means clustering
  - Hierarchical clustering
  - DBScan

Clustering

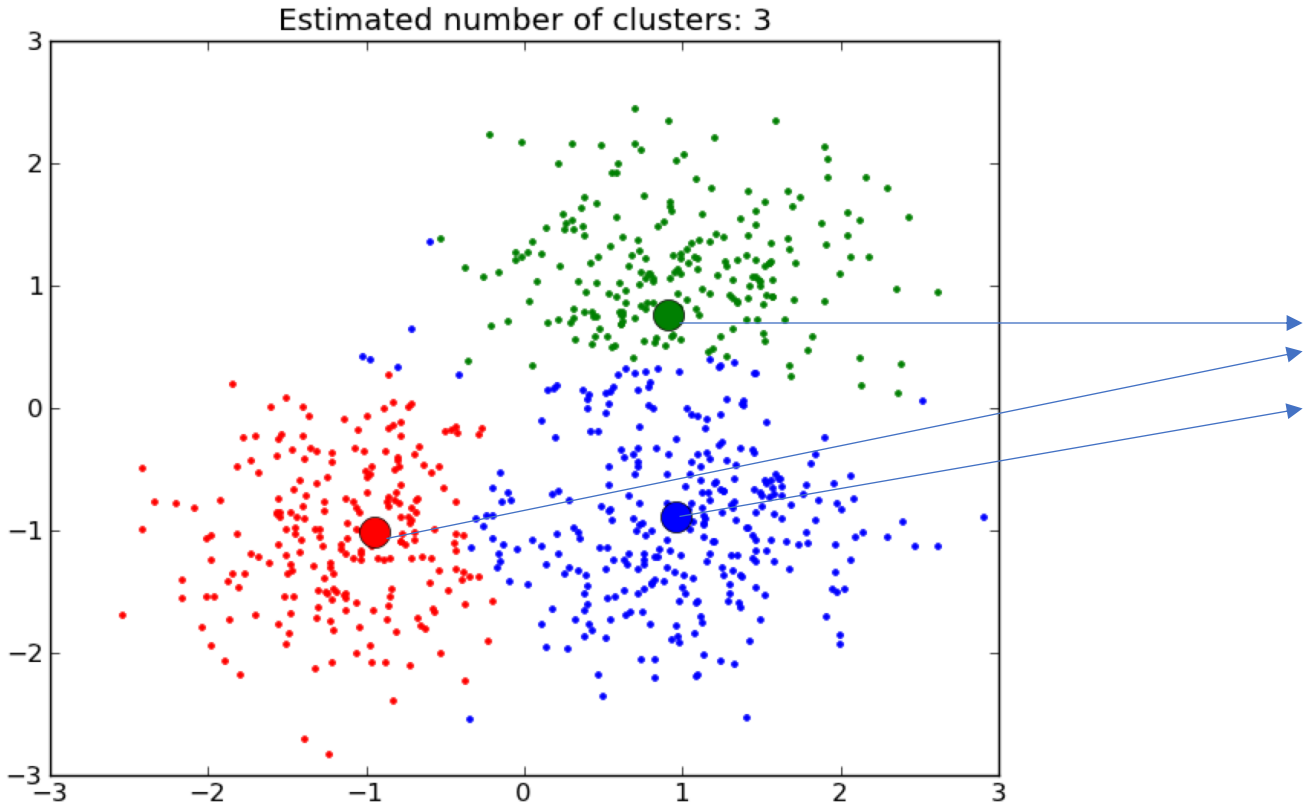
- Divide data into K-clusters
- Some algorithms:
  - K-means clustering
  - Hierarchical clustering
  - DBScan

K-means clustering

- Method to cluster unlabeled data points into K clusters
- K >= 2, but we want to make sure that clusters are interpretable and K is not too high
- Often try algorithm with multiple values of K to see how results compare

What is “means” in k means

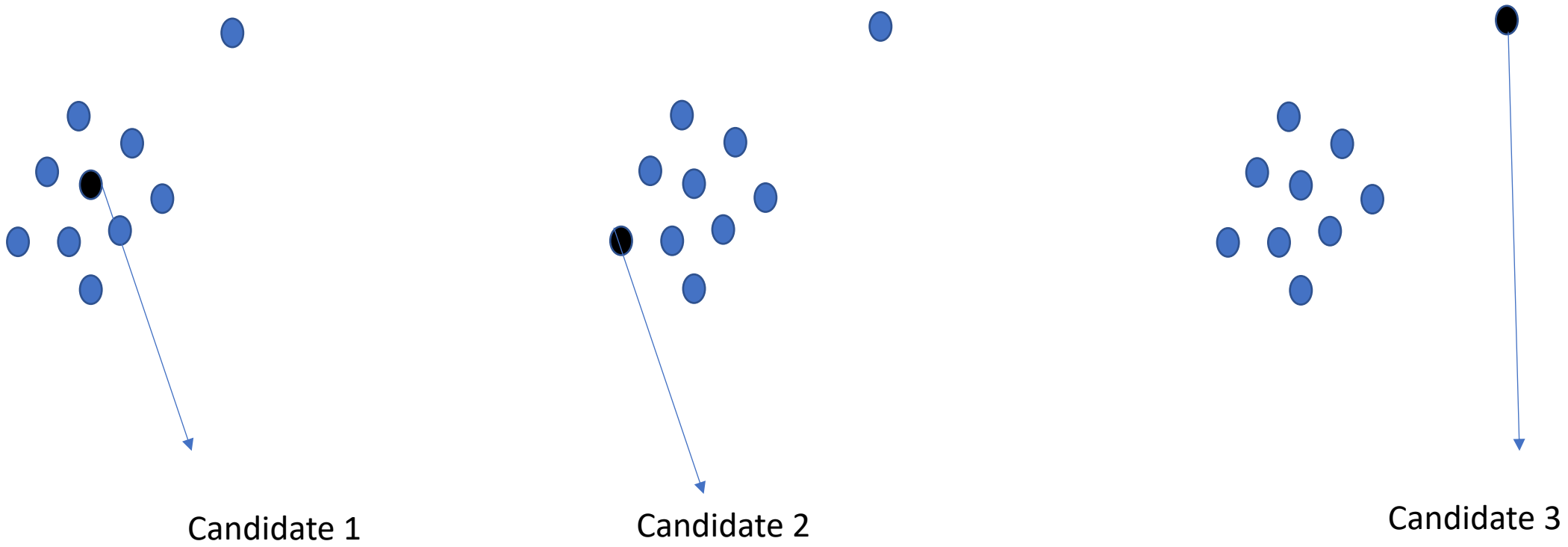
- Each cluster should have a centroid or mean



3 centroids for K=3

Exercise: Which one is a better mean?

- Candidate 1 or candidate 2 or candidate 3?



Why?

- Candidate 1 is somewhat centrally located
- Or, the average distance between candidate 1 and other points is minimum
- Mathematically,
  - Find a point $X^i \; such \; that$

$$minimize_{X^i} \; \sum_{j=1}^{N} dist(X^i, X^j)$$

K-means algorithm

- Step 0: Select K
- Step 1: Randomly select initial cluster “centroids”

K-means algorithm

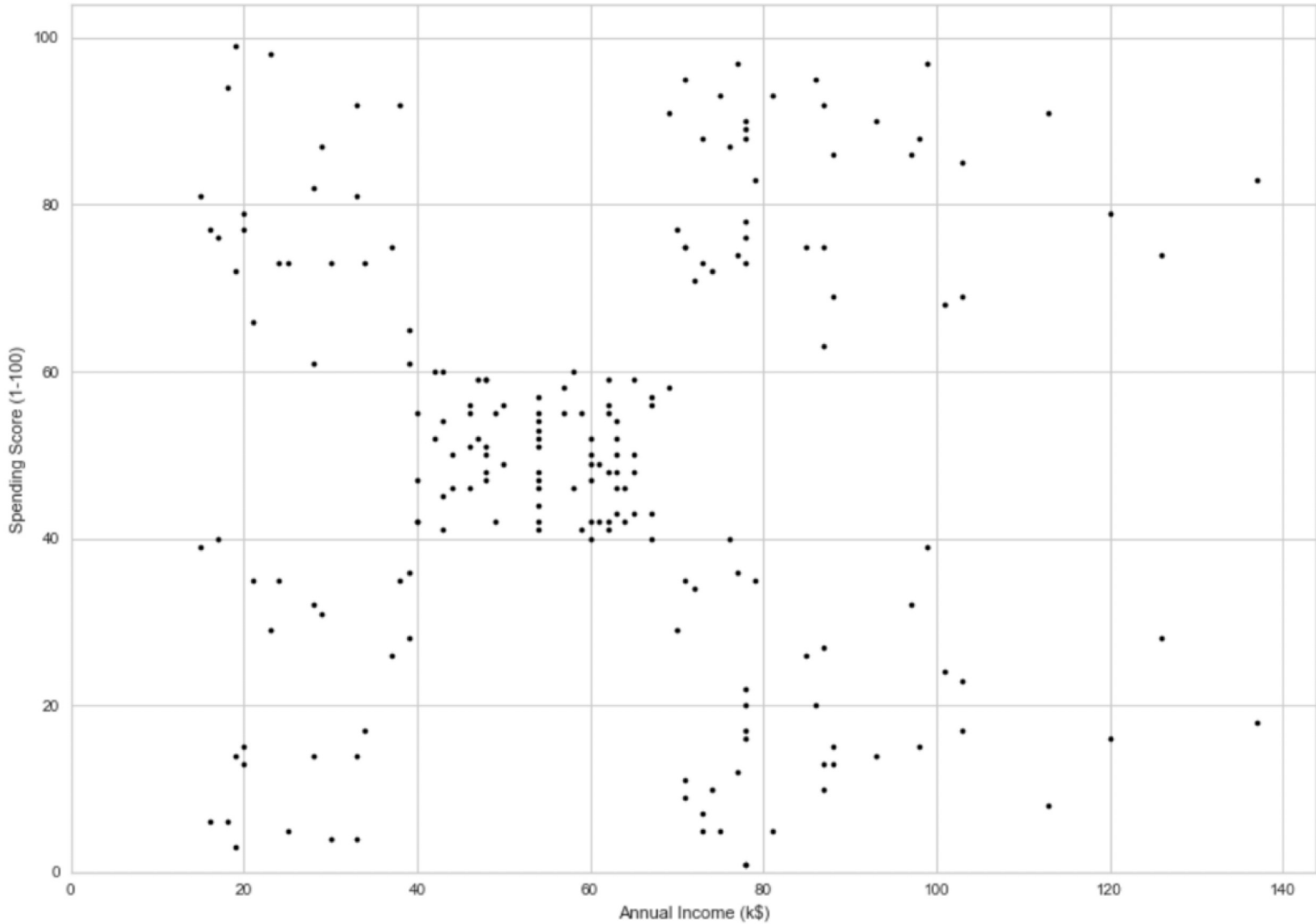
- Step 0: Select K
- Step 1: Randomly select initial cluster “centroids”
- Step 2: Assign all data points to the cluster of the nearest centroids

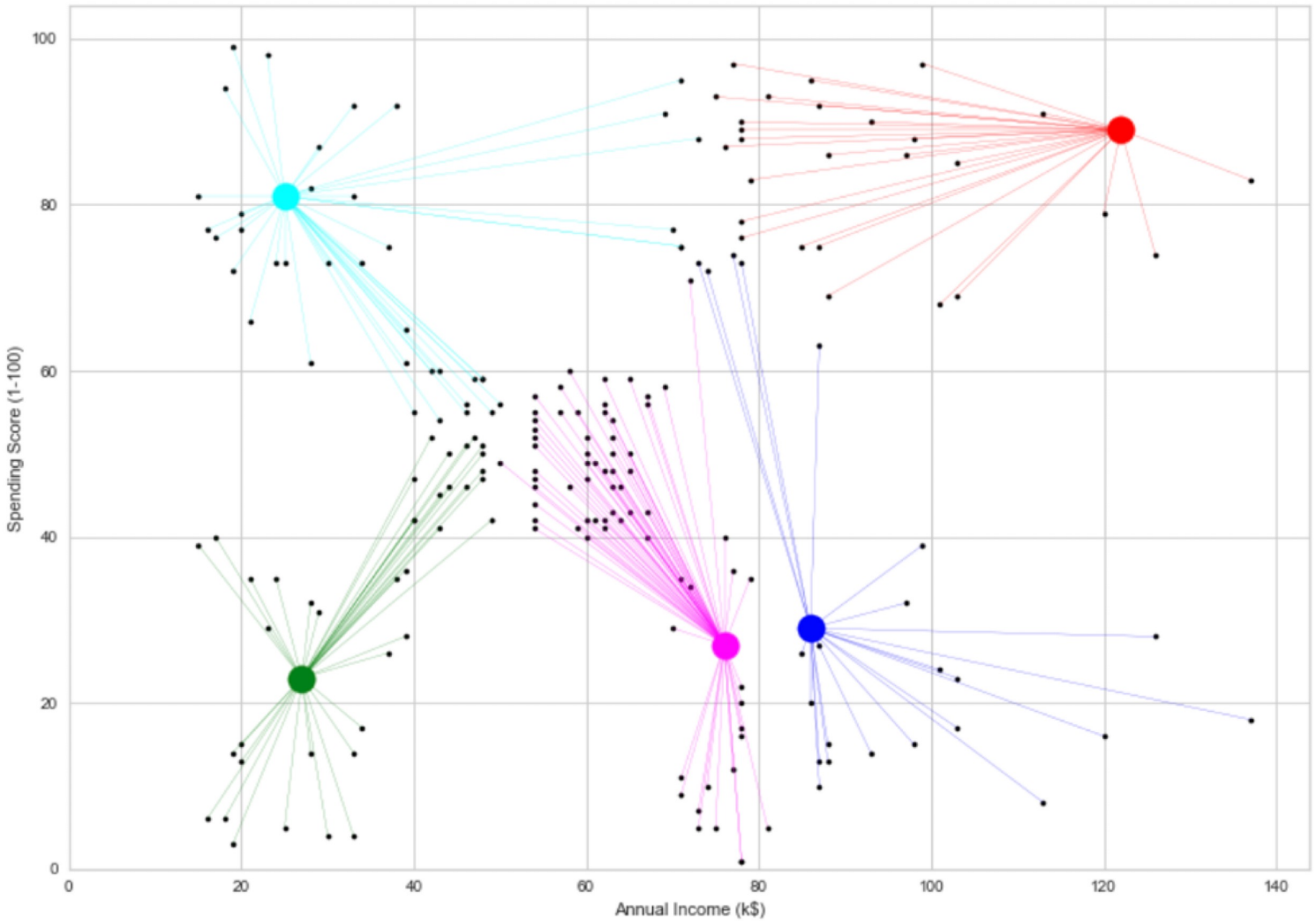
K-means algorithm

- Step 0: Select K
- Step 1: Randomly select initial cluster “centroids”
- Step 2: Assign all data points to the cluster of the nearest centroids
- Step 3: Compute the mean value of all clusters; these are the new centroids

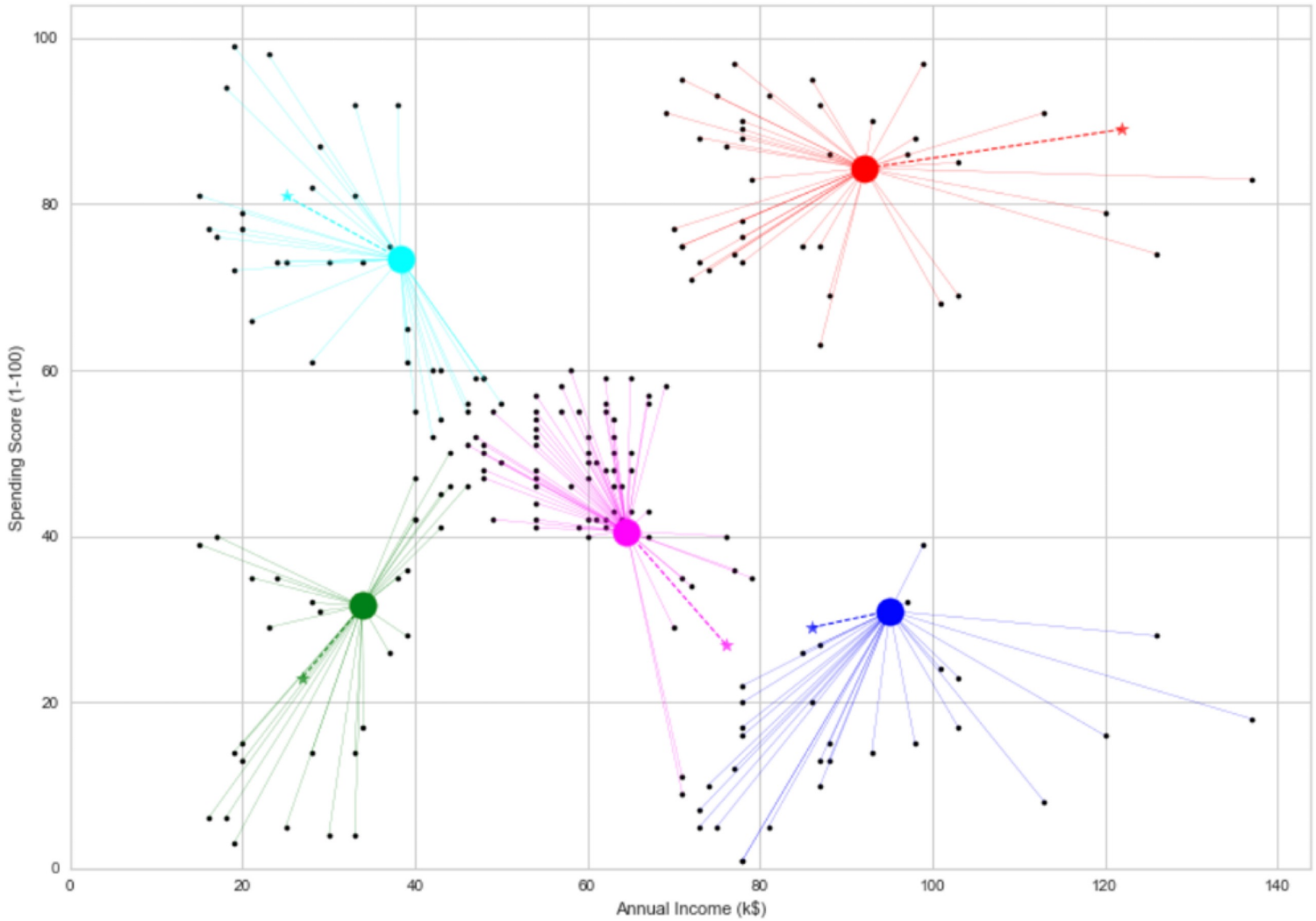
K-means algorithm

- Step 0: Select K
- Step 1: Randomly select initial cluster “centroids”
- Step 2: Assign all data points to the cluster of the nearest centroids
- Step 3: Compute the mean value of all clusters; these are the new centroids
- Step 4: If all the new centroids are same as old centroids, STOP else repeat from step 2

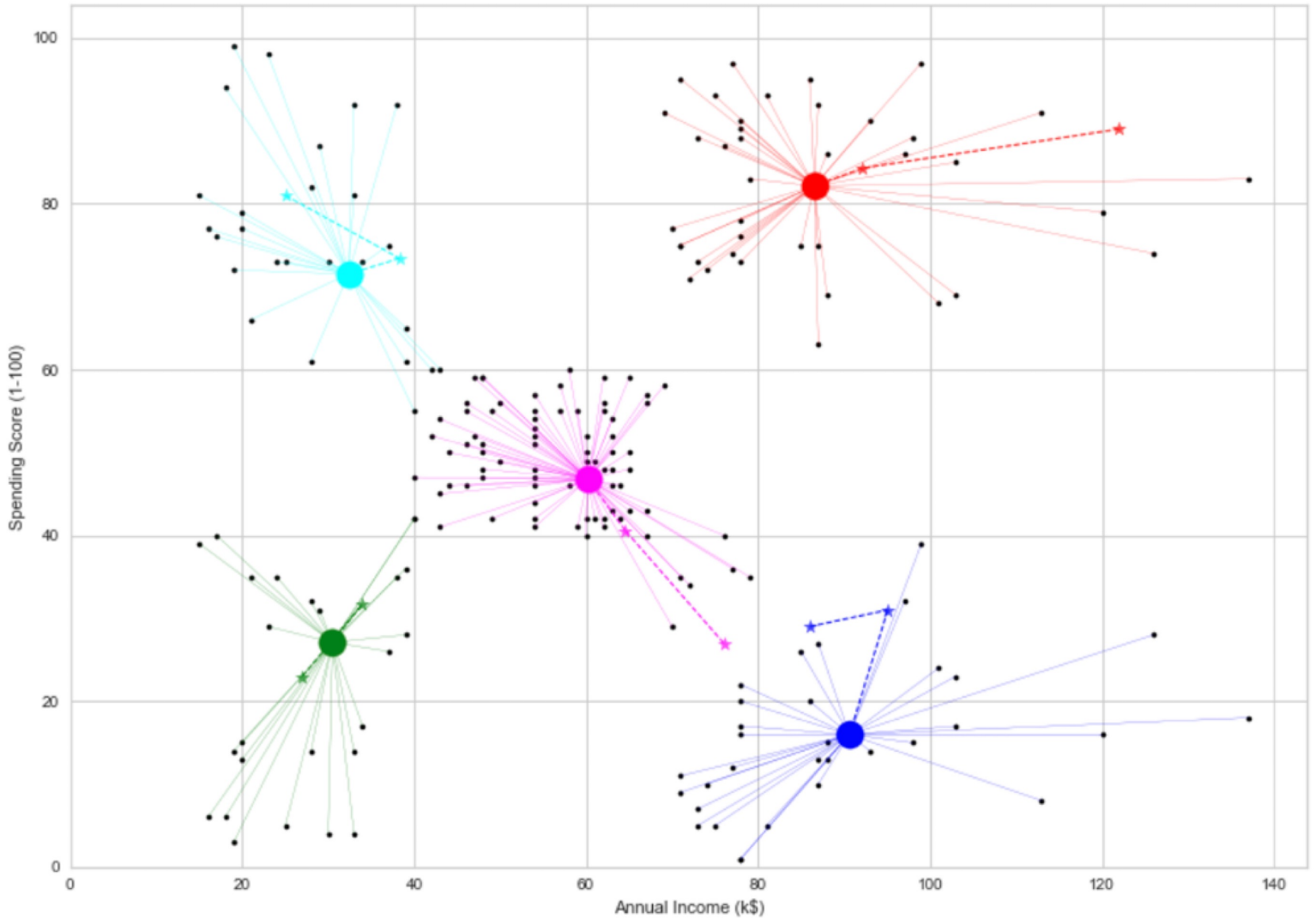

Spending Score (1-100)
Annual Income (k$)
0
20
40
60
80
100
120
140



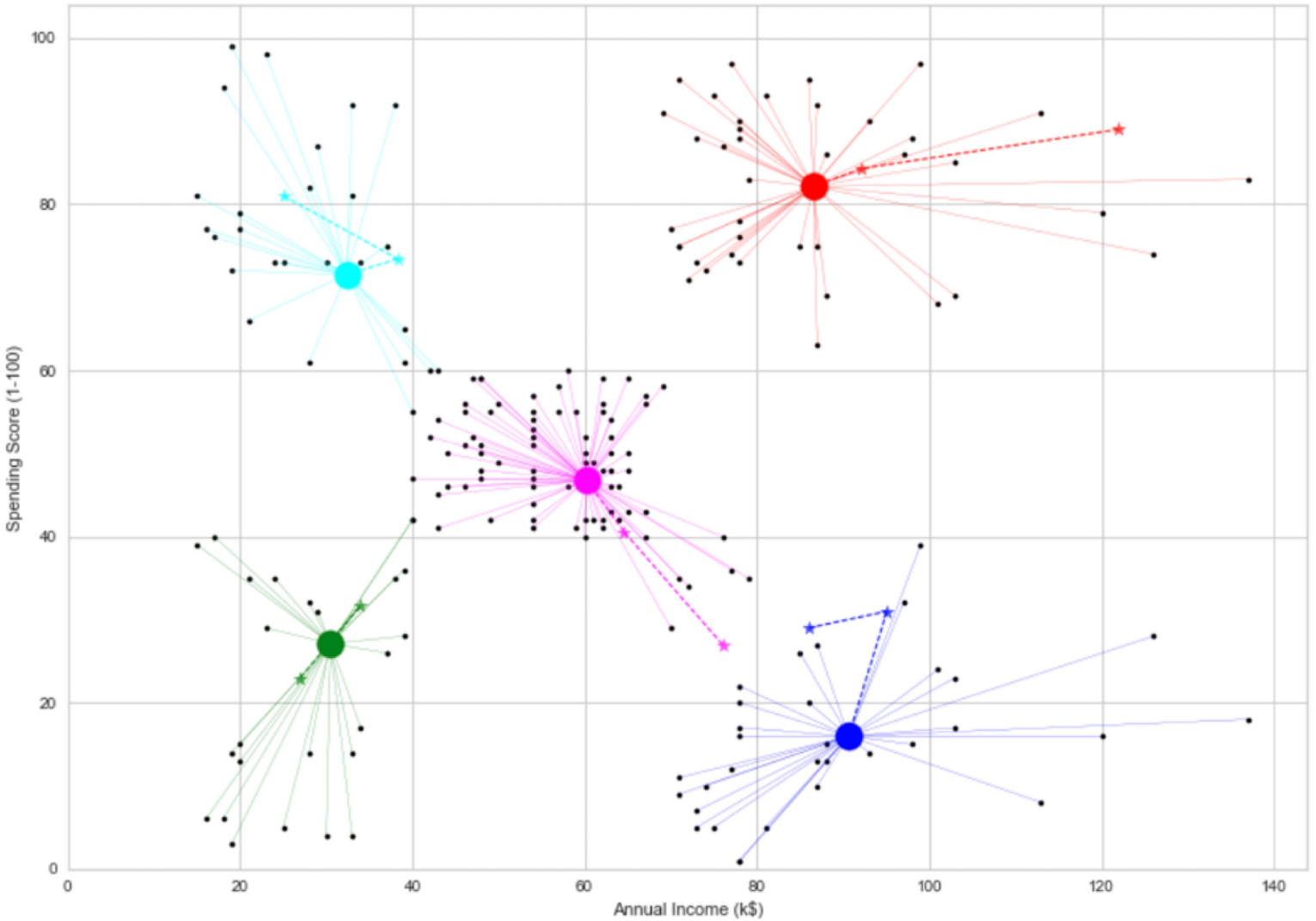
Iteration 1



Iteration 2



Iteration 3



Iteration 4

Convergence attained

K-means visualizer

http://shabal.in/visuals/kmeans/1.html

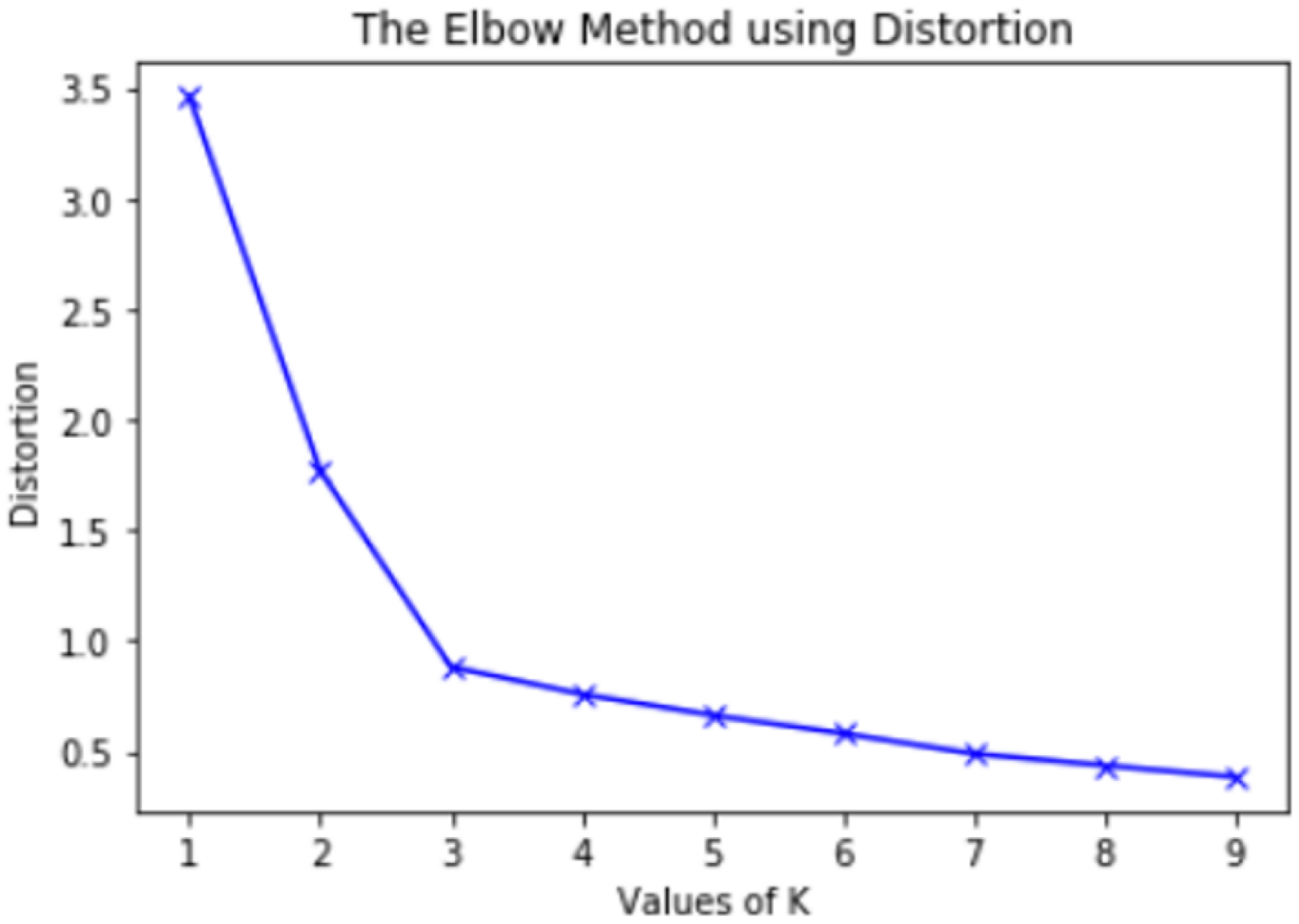
What does it mean in the context of NLP

- We know several ways to extract feature vectors from words / sentences / documents
- If we know how to cluster features, we can:
  - Cluster documents based on topics
  - Cluster words in a document / across documents
  - Analyze clusters
- E.g., Topic extraction from text using clustering of Glove vectors extracted for unique words in the document

Evaluating Clusters

- How do we know when our clusters are good?
- We optimize for cluster “tightness,” or mean distance from the points in the cluster to their center
- How do we know when we’ve chosen K well?
- Average distance from center will always go down as K increases
- Typically, there is a point at which this reduction flattens out

The elbow method



K = 4 is ideal

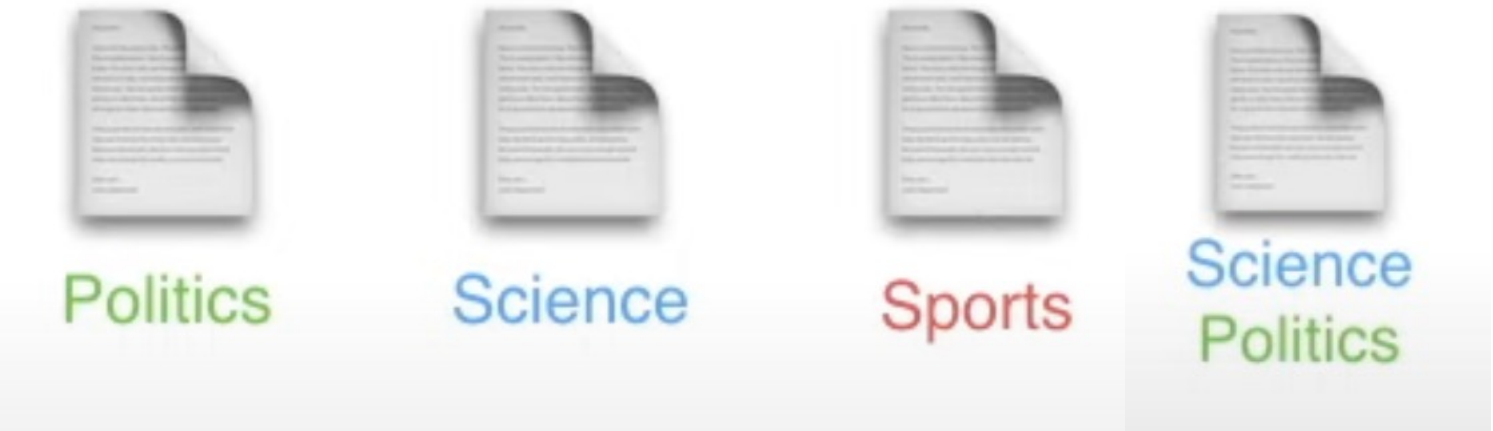
Topic Modeling

Topic Modeling

- Another unsupervised approach based on generative modeling
- Given a set of documents:

Topic Modeling

- We are interested in learning:
  - Given a document
  - How topics are distributed in documents or **P(topic | document)**



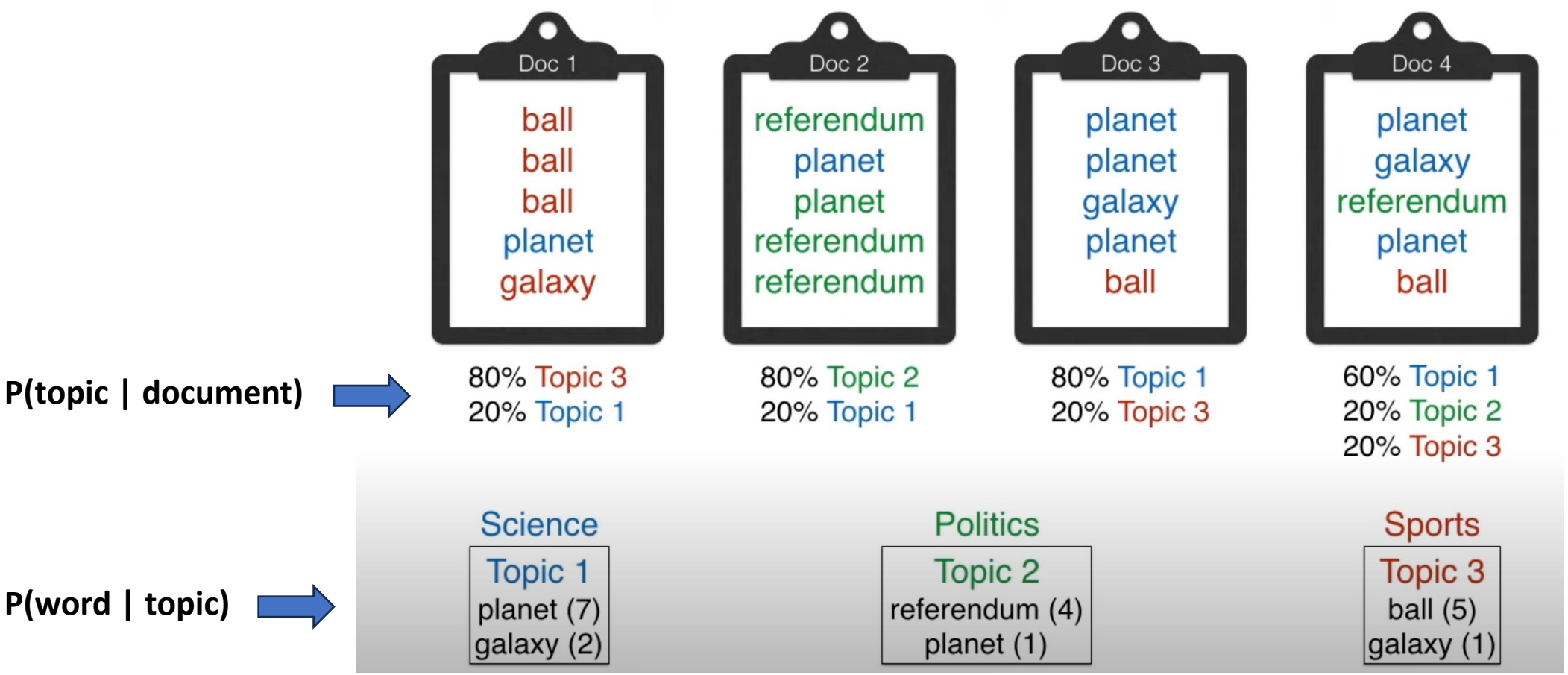
https://www.youtube.com/watch?v=BaM1uiCpj_E

Topic Modeling

- We are interested in learning:
  - Given a document
  - How topics are distributed in documents
  - And how each word contributes to topic **P(word | topic)**

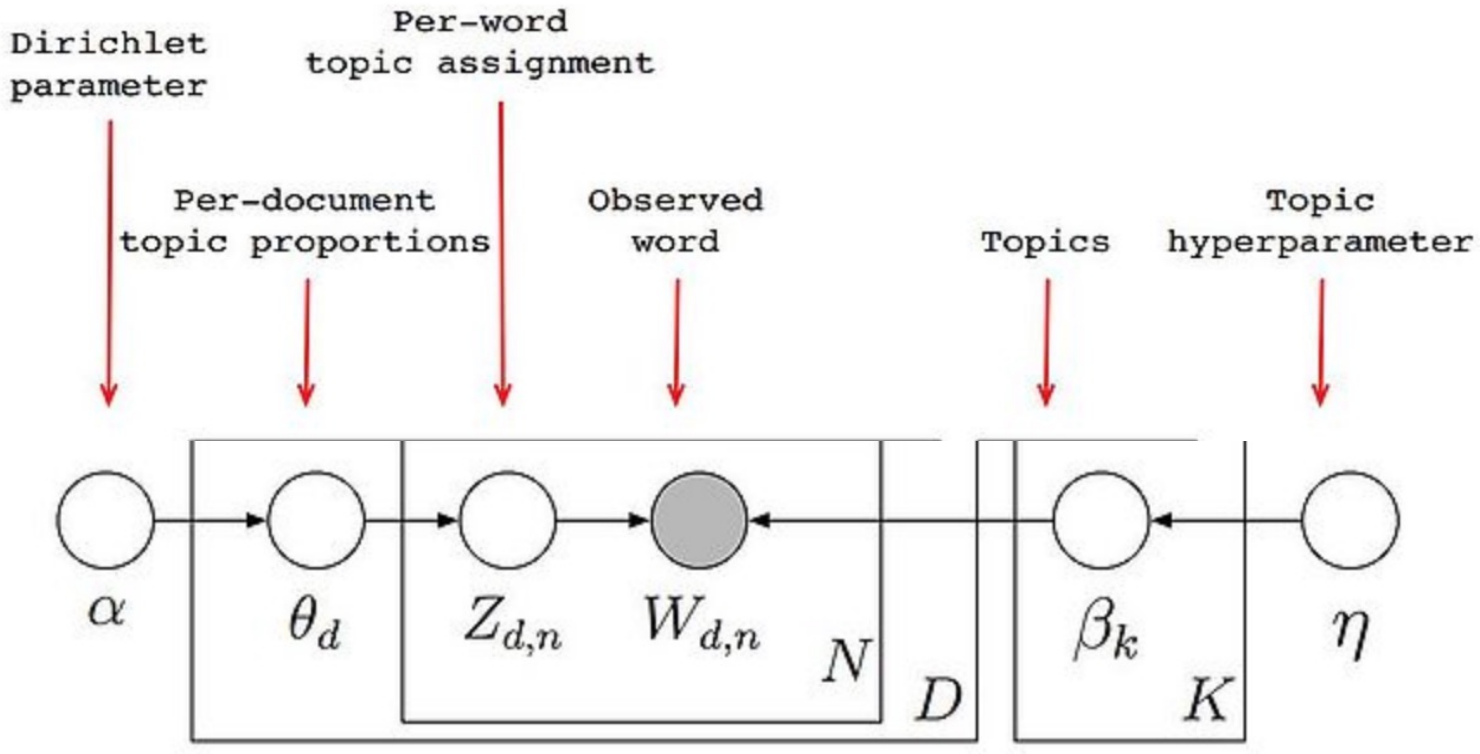
https://www.youtube.com/watch?v=BaM1uiCpj_E

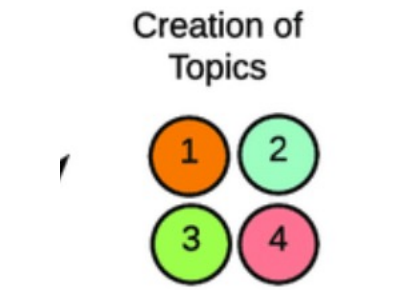
Overall

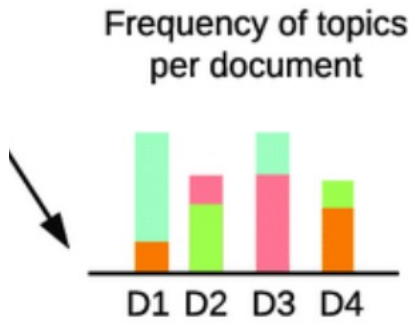

Doc 1
ball
ball
ball
planet
galaxy
Doc 2
referendum
planet
planet
referendum
referendum
Doc 3
planet
planet
galaxy
planet
ball
Doc 4
planet
galaxy
referendum
planet
ball
P(topic | document)
80% Topic 3
20% Topic 1
80% Topic 2
20% Topic 1
80% Topic 1
20% Topic 3
60% Topic 1
20% Topic 2
20% Topic 3
P(word | topic)
Science
Topic 1
planet (7)
galaxy (2)
Politics
Topic 2
referendum (4)
planet (1)
Sports
Topic 3
ball (5)
galaxy (1)

LDA: Estimating *P(word | topic), P(topic | document)*

Stands for **Latent Dirichlet Allocation**
It's a Bayesian Network







LDA: Estimating *P(word | topic), P(topic | document)*

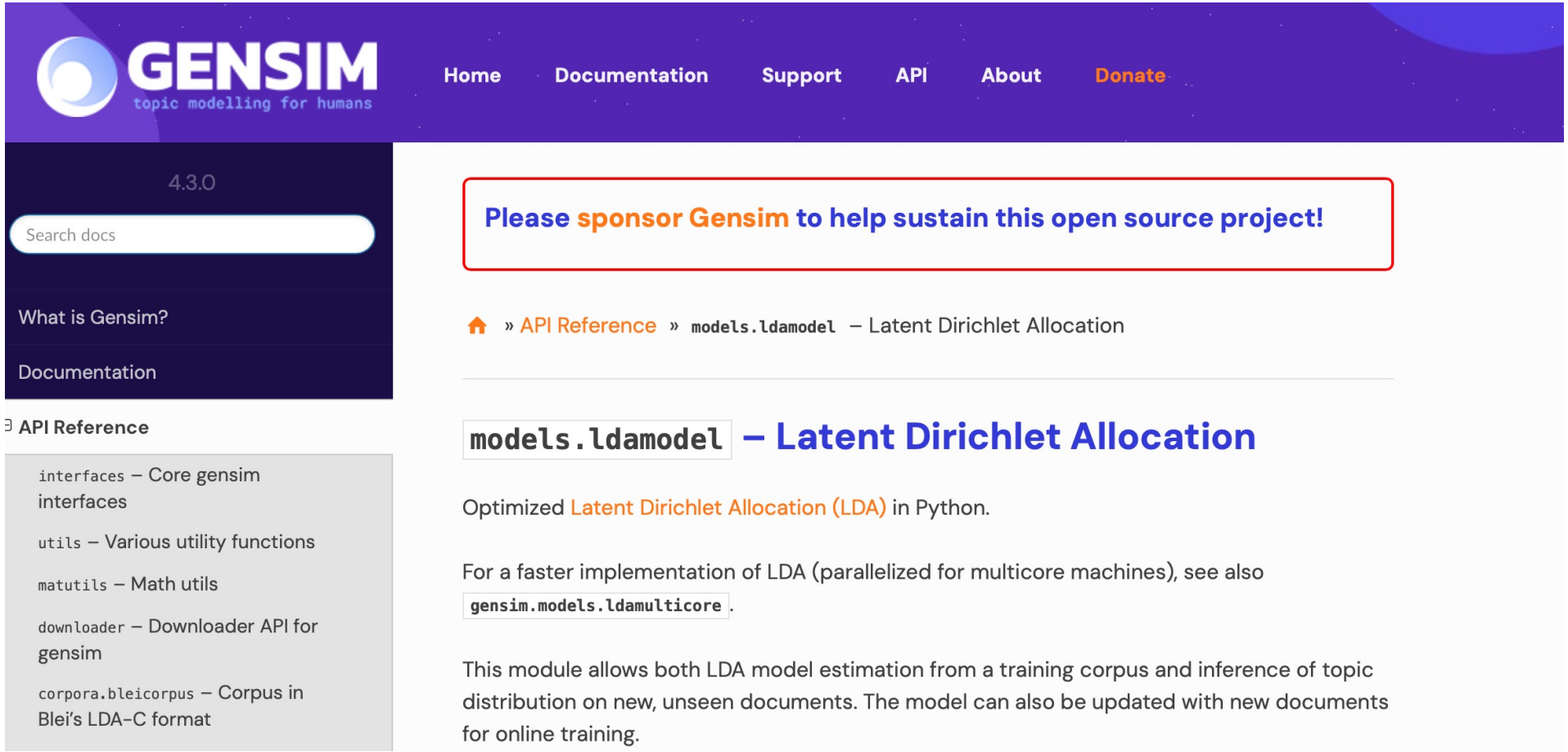
Solving estimation based on a directed graphical model (a.k.a Bayesian Networks)
Sampling techniques (such as Gibbs sampling to reduce complexity).

$$p(\vec{\theta}_{1:D}, z_{1:D,1:N}, \vec{\beta}_{1:K} \mid w_{1:D,1:N}, \alpha, \eta) = \frac{p(\vec{\theta}_{1:D}, \vec{z}_{1:D}, \vec{\beta}_{1:K} ; \vec{w}_{1:D}, \alpha, \eta)}{\int_{\vec{\beta}_{1:K}} \int_{\vec{\theta}_{1:D}} \sum_{\vec{z}} p(\vec{\theta}_{1:D}, \vec{z}_{1:D}, \vec{\beta}_{1:K} ; \vec{w}_{1:D}, \alpha, \eta)}.$$

Evaluation of Topics

- Mostly Qualitative:
  - In unsupervised settings, we do not have labels like Sports, Politics ***etc***
  - Humans will have to assess the topic quality based on manual assessment
  - Sometimes, quantitative metrics such for topic coherence, perplexity etc are used.

In Python


GENSIM
topic modelling for humans
Home
Documentation
Support
API
About
Donate
4.3.0
Search docs
What is Gensim?
Documentation
API Reference
interfaces – Core gensim interfaces
utils – Various utility functions
matutils – Math utils
downloader – Downloader API for gensim
corpora.bleicorpus – Corpus in Blei's LDA-C format
Please sponsor Gensim to help sustain this open source project!
» API Reference » models.ldamodel – Latent Dirichlet Allocation
models.ldamodel – Latent Dirichlet Allocation
Optimized Latent Dirichlet Allocation (LDA) in Python.
For a faster implementation of LDA (parallelized for multicore machines), see also gensim.models.ldamulticore.
This module allows both LDA model estimation from a training corpus and inference of topic distribution on new, unseen documents. The model can also be updated with new documents for online training.

Now…

- **Tutorial:**
  - Unsupervised ML and document clustering
  - Topic Extraction from Text