Faculty of Computer Science, Dalhousie University *30-Oct-2025*

CSCI 4152/6509 — Natural Language Processing

Lecture 11: N-gram Model and Markov Chain Model

Location: Studley LSC-Psychology P5260 Instructor: Vlado Keselj
Time: 14:35 – 15:55

Previous Lectures

- Fully Independent Model (continued)
- Naïve Bayes classification model
  - Assumption, definition
  - Graphical representation
  - Spam detection example
  - Computational tasks
  - Number of parameters
  - pros and cons, additional notes
  - Bernoulli and Multinomial Naïve Bayes

---

13 N-gram Model and Markov Chain Model

Before we introduce this model, let us first introduce *Language Modeling.*

To understand better the significance of the N-gram Model, let us first introduce the *language modeling*—an important task in the probabilistic NLP. *Language Modeling* can be described as the task of building a probabilistic model that can estimate the probability of an arbitrary natural language sentence; i.e., of the probability P(sentence).

One application of this problem is in speech recognition. In speech recognition, we are interested in

$$\arg\max_{\text{sentence}} \mathrm{P}(\text{sentence}|\text{sound})$$

This is equal to:

$$\begin{aligned}\arg\max_{\text{sentence}} \mathrm{P}(\text{sentence}|\text{sound}) &= \arg\max_{\text{sentence}} \frac{\mathrm{P}(\text{sentence},\text{sound})}{\mathrm{P}(\text{sound})} \\ &= \arg\max_{\text{sentence}} \mathrm{P}(\text{sentence},\text{sound}) \\ &= \arg\max_{\text{sentence}} \mathrm{P}(\text{sound}|\text{sentence})\mathrm{P}(\text{sentence})\end{aligned}$$

It is easier to estimate $\mathrm{P}(\text{sound}|\text{sentence})$ than $\mathrm{P}(\text{sentence},\text{sound})$, and it is done by an *acoustic model,* while $\mathrm{P}(\text{sentence})$ is estimated by a *language model.*

Slide notes:

Language Modeling

- Task of estimating probability of arbitrary utterance in a language
- Alternative task: Predicting the next token in a sequence: e.g., the next word or words, in a sentence, or next character or characters
- N-gram model: a "natural" model for this task

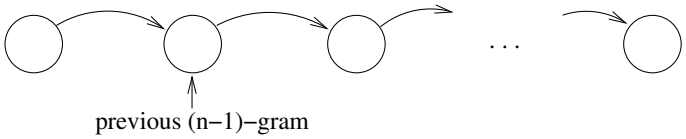
N-gram model is very simple and it is among the most successful models for language modelling; trigram ($n = 3$) word model in particular. In an n-gram model, we calculate joint distribution for all n-tuples of consecutive words (or characters). For example, in the trigram model, we count the number of occurrences of each triple of consecutive words from a corpus. Using this statistics, we can estimate the probability of arbitrary word w_3 following two given words w_1 and w_2: $\mathrm{P}(w_3|w_1w_2)$. It is useful to assign some small probability to unseen triples as well (using a technique called *smoothing*). If we use two "dummy" words '$\cdot$' at the beginning of each sentence, then the probability of arbitrary sentence can be calculated as:

$$\mathrm{P}(w_1w_2 \dots w_n) = \mathrm{P}(w_1|\cdot\cdot)\mathrm{P}(w_2|w_1\cdot)\mathrm{P}(w_3|w_2w_1) \dots \mathrm{P}(w_n|w_{n-1}w_{n-2})$$

N-gram Model: Notes

- Reading: Chapter 4 of [JM]
- Use of log probabilities
  - similarly as in the Naïve Bayes model for text
- Graphical representation

Graphical Representation



Use of log probabilities

Multiplying a large number of probability values, which are typically small, gives a very small result (close to zero), so in order to avoid floating-point underflow, we should use addition of logarithms of the probabilities in the model.

Slide notes:

N-gram Model as a Markov Chain

- N-gram Model is very similar to Markov Chain Model
- Markov Chain consists of
  - sequence of variables V_1, V_2, ...
  - probability of V_1 is independent
  - each next variable is dependent only on the previous variable: V_2 on V_1, V_3 on V_2, etc.
  - Conditional Probability Tables: $P(V_1)$, $P(V_2|V_1)$, ...
- Markov Chain is identical to bi-gram model, but higher-order n-gram models are very similar as well

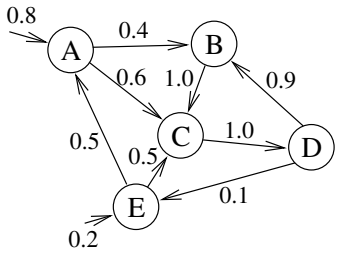
Markov Chain: Formal Definition

A *stochastic process* in general is a family of random variables $\{V_i\}$, where i is an index from a set I. A stochastic process is also denoted as $\{V_i, i \in I\}$, or $\{V_t, t \in T\}$, with intuition coming from time index. The index set I can be an arbitrary ordered set, but we will usually assume they it is either finite or countably infinite (i.e., enumerable), and then process can be denoted as $\{V_i\}_{i=1}^{\infty}$. A process is called a *Markov process* if given the value of V_t, for some index t, the values of V_s, where $s > t$, do not depend on values of V_u, where $u < t$. In case of a finite or countably infinite index set, this means that the value of V_i depends only on the value of the previous variable V_{i-1}. In this case, the Markov process is called a *Markov Chain.*

A Markov Chain can be described similarly to a Deterministic Finite automaton (DFA), but instead of reading input, we assume that we start in a random state based on a probability distribution, and change states in sequence based

on a probability distribution of the next state given the previous state.

For example, a Markov chain could be illustrated in the following way.



This model could generate the sequence $\{A, C, D, B, C\}$ of length 5 with probability:

$$0.8 \cdot 0.6 \cdot 1.0 \cdot 0.9 \cdot 1.0 = 0.432$$

assuming that we are modelling sequences of this length.

If we want to model sequences of arbitrary length, we would also need a stopping probability.

Evaluating Language Models: Perplexity

- Evaluation of language model: extrinsic and intrinsic
- Extrinsic: model embedded in application
- Intrinsic: direct evaluation using a measure
  In extrinsic evaluation, the language model is embedded in a wider application, and the performance of the model is measured through the performance of the application. For example, we can evaluate performance of a language model by measuring improvement in a speech recognition application in which it is embedded.
  In intrinsic evaluation, we directly evaluate the language model using some measure, such as perplexity.
- Perplexity, W — text, $L = |W|$,

$$\text{PP}(W) = \sqrt[L]{\frac{1}{P(W)}} = \sqrt[L]{\prod_i \frac{1}{P(w_i|w_{i-n+1} \ldots w_{i-1})}}$$

- Weighted average branching factor

Perplexity

Use of Language Modeling in Classification

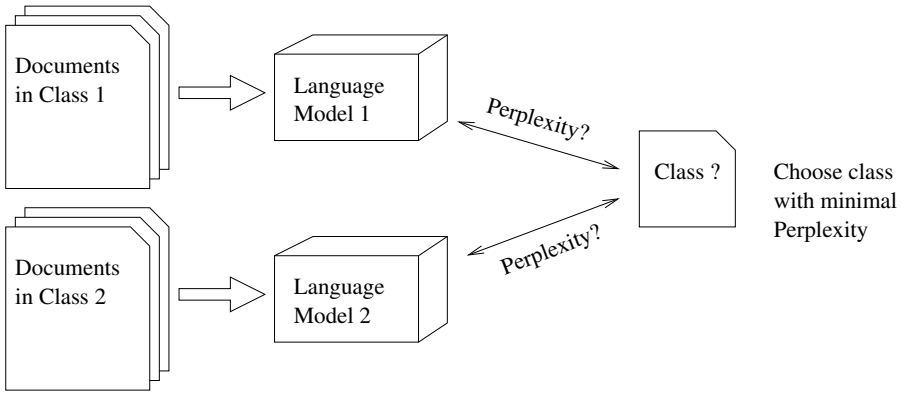
- Perplexity, W — text, $L = |W|$,

$$\text{PP}(W) = \sqrt[L]{\frac{1}{P(W)}} = \sqrt[L]{\prod_i \frac{1}{P(w_i|w_{i-n+1} \ldots w_{i-1})}}$$

- Text classification using language models

The perplexity measure tells us how well a language model predicts an existing text. It is also called a weighted branching factor. For example, if we generate a text using words from a 100,000-word vocabulary, and we have not better way to predict words than randomly choosing them using a uniform distribution, then the perplexity measure for any text will be about 100,000. A lower perplexity measure means that the model is "predicting" a text better. For example, a perplexity of 4 means that the model predicts the next word with odds of 1 to 4, which would be quite good.

We can do text classification using language models and perplexity in the following way: We build language models for different classes by training them using training data for each class. Then, we measure perplexity of each model on a test document, and we choose the class whose model has lowest perplexity, as shown in the following figure:



Slide notes:

Unigram Model and Multinomial Naïve Bayes

- It is interesting that classification using Unigram Language Model is same as Multinomial Naïve Bayes with all words

13.1 N-gram Model Smoothing

Smoothing is any technique in probabilistic modeling used to avoid zero probabilities in a model trained from training data. Namely, due to the sparse data problem, we may easily have one of the probabilities estimated to zero by the MLE process. Since these probabilities are typically used as factors in products, this easily leads to a zero probability being assigned to a full configuration that happen not to be seen in the training data. To avoid this situation, we use smoothing techniques. Generally speaking, the smoothing techniques take some probability from seen and predictable events and assign it to the unseen events. There are several well-known smoothing techniques used in the n-grams model: add-one smoothing (or Laplace smoothing), Witten-Bell smoothing, Good-Turing smoothing, Kneser-Ney smoothing (described in the new edition of the Jurafsky and Martin textbook), etc. We will now take a closer look at the Laplace (add-one) and the Witten-Bell smoothing. These techniques can be generalized to other models as well.

Example: Character Unigram Probabilities

- Training example: `mississippi`
- What are letter unigram probabilities?
- What would be probability of the word '`river`' based on this model?

The letter unigram probabilities without smoothing:

Letter	Counts	Estimated frequency
i	4	$4/11 \approx 0.363636363636364$
m	1	$1/11 \approx 0.0909090909090909$
p	2	$2/11 \approx 0.181818181818182$
s	4	$4/11 \approx 0.363636363636364$
other letters	0	0
Total:	11	1.0

The probability of the word '`river`' would be 0 in this model, since it contains letters with the probability 0, so the

product of those letter probabilities would be 0:

$$\mathrm{P}(\text{`river'}) = \mathrm{P}(\mathtt{r})\mathrm{P}(\mathtt{i})\mathrm{P}(\mathtt{v})\mathrm{P}(\mathtt{e})\mathrm{P}(\mathtt{r}) = 0 \cdot \frac{4}{11} \cdot 0 \cdot 0 \cdot 0 = 0.0$$

13.1.1 Add-one Smoothing (Laplace Smoothing)

The *add-one smoothing* is also known the *Laplace smoothing*. We start with the count of 1 for all events, and thus prevent any events to end up with the probability 0. In a unigram example, it would mean that all tokens $w \in V$, where V is the vocabulary, start with count 1. If $|V|$ is the vocabulary size, and n is the number of tokens in a text, the smoothed unigram probabilities end up to be

$$P(w) = \frac{\#(w) + 1}{n + |V|}$$

where $\#(w)$ denotes the number of occurrences of the token w in the training text.

Similarly, for bigram smoothing for example, the estimated probability would be:

$$P(a|b) = \frac{\#(ba) + 1}{\#(b) + |V|}$$

If the vocabulary size is very large compared to $\#(b)$, which happens with words, for example, or if b is relatively rare, then this kind of smoothing takes too much of the probability distribution from the seen events and assigns it to the unseen events.

Mississippi Example: Add-one Smoothing

- Let us again consider the example trained on the word: `mississippi`
- What are letter unigram probabilities with add-one smoothing?
- What is the probability of: `river`

With the Add-one smoothing, we would start with count 1 for each letter in the vocabulary. If we assume that our vocabulary consists of all lower-case letters in the English alphabet, the total alphabet size would be 26. Since each letter count would start with 1, with 11 letters in the word 'mississippi', we would have a total count of 37. This leads to the following table of smoothed probabilities:

Letter	Modified counts	Estimated frequency
i	5	5/37 $\approx$ 0.135135135135135
m	2	2/37 $\approx$ 0.0540540540540541
p	3	3/37 $\approx$ 0.0810810810810811
s	5	5/37 $\approx$ 0.135135135135135
other letters($\times$22)	1($\times$22)	1/37 $\approx$ 0.027027027027027 ($\times$22)
Total:	37	1.0

The probability of the word '`river`' in this model would be:

$$\mathrm{P}(\text{`river'}) = \mathrm{P}(\mathtt{r})\mathrm{P}(\mathtt{i})\mathrm{P}(\mathtt{v})\mathrm{P}(\mathtt{e})\mathrm{P}(\mathtt{r}) = \frac{1}{37} \cdot \frac{5}{37} \cdot \frac{1}{37} \cdot \frac{1}{37} \cdot \frac{1}{37} \approx 7.21043363591149 \cdot 10^{-8}$$

13.1.2 Witten-Bell Discounting

In the context of data compression, Witten and Bell (1991) analyzed several smoothing methods, under the title "The Zero-Frequency Problem: Estimating the Probabilities of Novel Events in Adaptive Text Compression". They

considered three methods A, B, and C, and then, based on a Poisson process modelling, the methods P, X, and XC. It is interesting to note that the method X uses the same, or very similar, idea as in the Good-Turing smoothing.

The method C is what is usually referred to as the Witten-Bell smoothing. It uses an intuitive idea from data compression. Let us assume that we use a data compression method, which uses a dictionary of tokens w_1, w_2, ..., w_r, so far. As long as the new tokens are from this set, we can encode them in some way, but whenever a new token appears, we need a special 'escape' code to introduce this token to the vocabulary. In this way, we can think of new tokens appearing as a new event, beside the events of seeing existing tokens. This is supported in practice by seeing approximately constant rate of new words appearing in a text after some initial reading. We can use the frequency of such 'escape' code as an estimate of the probability of seeing previously unseen events, and make sure that we allocate that much probability distribution mass for the smoothing purposes.

Example: Let us consider again using the example of training data '`mississippi`' to train a unigram model, and then use it to estimate probability of the string '`river`'.

We will consider again estimating probability of 26 lower-case letters from the word '`mississippi`'. As in the case of unsmoothed n-grams, we will count letters: '`m`' 1 time, '`i`' 4 times, '`s`' 4 times, and '`p`' 2 times. However, we also note that we saw 4 different letters, which is equivalent to seeing 'escape' character 4 times, so we will reserve count 4 for unseen events in future as well. This is how we get the following probabilities using the Witten-Bell discounting:

Letter	Modified counts	Estimated frequency
i	4	4/15 $\approx$ 0.266666666666667
m	1	1/15 $\approx$ 0.0666666666666667
p	2	2/15 $\approx$ 0.133333333333333
s	4	4/15 $\approx$ 0.266666666666667
new letters total	4	4/15 $\approx$ 0.266666666666667
Total:	15	

When we split the probability reserved for new letters equally among the remaining $26 - 4 = 22$ letters, we obtain the final estimated frequency:

Letter	Estimated frequency
i	4/15 $\approx$ 0.266666666666667
m	1/15 $\approx$ 0.0666666666666667
p	2/15 $\approx$ 0.133333333333333
s	4/15 $\approx$ 0.266666666666667
other letters	$\frac{4}{15 \cdot 22} = 2/165 \approx 0.0121212121212121$

The probability of the word '`river`' in this model would be:

$$\mathrm{P}(\text{`river'}) = \mathrm{P}(\mathtt{r})\mathrm{P}(\mathtt{i})\mathrm{P}(\mathtt{v})\mathrm{P}(\mathtt{e})\mathrm{P}(\mathtt{r}) = \frac{2}{165} \cdot \frac{4}{15} \cdot \frac{2}{165} \cdot \frac{2}{165} \cdot \frac{2}{165} \approx 5.75642615879697 \cdot 10^{-9}$$

Formulae for Witten-Bell Discounting If we want to express this in terms of formulae, we will denote that we saw r distinct tokens in a text of length n, or we can say that we saw n events and r 'escape' events, so the probability of seeing new tokens is $\frac{r}{n+r}$. Hence the unigram probability for seen tokens:

$$P(w) = \frac{\#(w)}{n+r}$$

and the total probability for unseen tokens is:

$$\frac{r}{n+r}$$

It is convenient that in the previous formulae we did not need to know the vocabulary size. If we do know the vocabulary size, we can now divide the probability for unseen tokens equally, and obtain:

$$P(w) = \frac{r}{(n+r)(|V| - r)}$$

for unseen tokens w.

Bigrams and Higher-order n-grams

The probabilities for bigrams and higher-order n-grams are smoothed in a similar way:

$$P(a|b) = \frac{\#(ba)}{\#(b) + r_b}$$

for seen bigrams ba, where r_b is the number of different tokens following b. The number $\#(b)$ does not represent necessarily the exact number of occurrences of b in this case. More precisely, it is the number of occurrences of b except at the end of text; i.e., the number of occurrences of b where b is followed by another token. The remaining probability mass for unseen events:

$$\frac{r_b}{\#(b) + r_b}$$

is used for unseen bigrams that start with b, and is usually divided according to lower-order n-grams; which would be unigrams in this case. If N_b is the set of all tokens that never follow b in the training data, then:

$$P(a|b) = \frac{r_b}{\#(b) + r_b} \cdot P(a)/\Sigma_{x \in N_b} P(x)$$

for unseen bigrams ba.

The next model: HMM. Our next probabilistic model is the Hidden Markov Model (HMM), and it is applicable to the task of labelling tokens of a sequence, such as the task of Part-of-Speech tagging (POS Tagging). Before that, we will make a review of the parts of speech in English, which are quite applicable with some changes to other natural languages as well.

Slide notes:

The Next Model: HMM

- HMM — Hidden Markov Model
- Typically used to annotate sequences of tokens
- Most common annotation: Part-of-Speech Tags (POS Tags)
- First, we will make a review of parts of speech in English

14 Part-of-Speech Tags (POS Tags)

Slide notes:

Part-of-Speech Tags (POS Tags)

- Reading: Sections 5.1–5.2 (Ch. 8 in new edition)
- Word classes called **Part-of-Speech (POS) classes**
  - also known as **syntactic categories**, **grammatical categories**, or **lexical categories**
- Ambiguous example: Time flies like an arrow.

	Time	flies	like	an	arrow.
1.	N	V	P	D	N
2.	N	N	V	D	N
⋮					

- **POS tags:** labels to indicate POS class
- **POS tagging:** task of assigning POS tags

Note about reading material: Some reading material for the topics in this section can be found in the JM textbook in Sections 5.1–5.2 ("5.1 (Mostly) English Word Classes" and "5.2 Tagsets for English"), or in Chapter 8 of the upcoming edition 3 of the book.

The words in text are divided into classes according to their function, and these classes are called **Part-of-Speech classes** or **POS classes** for short. The POS classes are also sometimes called **syntactic categories**, **grammatical categories**, or **lexical categories**.

We can take a look at the ambiguous example of the sentence "Time flies like an arrow." that we used before. The sentence can be interpreted in two ways, one is that the time goes very fast, and another one is that a species of flies, called "time flies", like the arrow fruit. We can even think of a third meaning, which is a command to go and time the files immediately. If we label the words in this sentence according to their part-of-speech classes, we get three different sequences of labels for the the three different meanings as follows:

	Time	flies	like	an	arrow.
1.	N	V	P	D	N
2.	N	N	V	D	N
3.	V	N	P	D	N

The labels used below the words denote the following well-known POS classes: 'N' for nouns, 'V' for verbs, 'P' for prepositions, and 'D' for determiners.

The task of determining the part-of-speech label for each word in a sentence, or a text in general is called **POS tagging**. From the above example, we can see that POS tagging is ambiguous, i.e., it may depend on the text interpretation by a reader.

POS Tag Sets

The concept of parts of speech as types of words used in language is known in linguistics for a long time. It was mentioned by several Antient Greek writers and it is well described in the work "The Art of Grammar", which is believed to be written by Dionysius Thrax in the 2nd century BC. This work distinguishes the following eight parts of speech: nouns, verbs, pronouns, prepositions, adverbs, conjunctions, participles, and articles.

Slide notes:

POS Tag Sets

- Traditionally based on Ancient Greece source: eight parts of speech:
  - nouns, verbs, pronouns, prepositions, adverbs, conjunctions, participle, and articles
- Computer processing introduced a need for a large set of categories
- Useful in NLP, e.g.: named entity recognition, information extraction
- Various POS tag sets (in NLP):
  Brown Corpus, Penn Treebank, CLAWS, C5, C7, ...
- We will use the Penn Treebank system of tags

WSJ Dataset

- WSJ — Wall Street Journal data set
- Most commonly used to train and test POS taggers
- Consists of 25 sections, about 1.2 million words
- Example:

```
Pierre NNP Vinken NNP , , 61 CD years NNS old JJ , ,
  will MD join VB the DT board NN as IN a DT
  nonexecutive JJ director NN Nov. NNP 29 CD . .
Mr. NNP Vinken NNP is VBZ chairman NN of IN
  Elsevier NNP N.V. NNP , , the DT Dutch NNP
  publishing VBG group NN . .
Rudolph NNP Agnew NNP , , 55 CD years NNS old JJ
  and CC former JJ chairman NN of IN Consolidated NNP
  Gold NNP Fields NNP PLC NNP , , was VBD named VBN
```

Open and Closed Categories

- Word POS categories are divided into two sets: *open* and *closed* categories:
- **open categories**
  - dynamic set
  - content words
  - larger set
  - e.g.: nouns, verbs, adjectives
- **closed categories** or **functional categories**:
  - fixed set
  - small set
  - frequent words
  - e.g.: articles, auxiliaries, prepositions

The words of a language, and generally POS categories, can be divided into two sets: *open and closed categories*.

Open POS categories are lexical categories that are dynamic in sense that their content changes over time, or depending on dialect or domain of usage. These sets are larger and the words bear most of the information content in a text. Some examples of open word categories are nouns, verbs, and adjectives.

Closed or **functional POS categories** are lexical categories consisting of fixed sets of words, which are used frequently in text and they are typically used in a functional way, i.e., as syntactic fillers and with less information content. Examples of such categories are articles, auxiliaries, and prepositions.

14.1 Open Word Categories

The open word categories are: *nouns, adjectives, verbs,* and *adverbs*. There are also groups of adverbs that belong to the closed word categories. Generally, like many other rules in natural language, this division is not strict and there are many exceptions.

Open Word Categories

- nouns (NN, NNS, NNP, NNPS)
  - concepts, objects, people, and similar
- adjectives (JJ, JJR, JJS)
  - modify (describe) nouns
- verbs (VB, VBP, VBZ, VBG, VBD, VBN)
  - actions
- adverbs (RB, RBR, RBS)
  - modify verbs, but other words too

Nouns (NN, NNS, NNP, NNPS)

Nouns refer to people, animals, objects, concepts, and similar.

Features:

- number: singular, plural
- case: subject (nominative), object (accusative)
- Some languages have more cases, and more number values
- Some languages have grammatical gender

Nouns have a number of linguistic properties called *features*, which vary across languages. The features are important in creating larger phrases and sentences in a linguistically correct way. The most common feature is *number* and the main values for it are *singular* and *plural*, expressing whether we are talking about one instance of an object or multiple instances. Some languages distinguish a more finer-grained set of values for number. The number, as other features, is typically expressed by modifying a suffix of a word; for example, the suffix *-s* is added in English for plural nouns.

Case is another common feature for nouns, which is not used much in English. The case indicates the syntactic and semantic role a noun plays in a phrase or a sentence. For example, the *nominative* case is used for nouns in the subject position in a sentence, while *accusative* case is used in the direct object position.

Noun Tags and Examples

Slide notes:

Noun Tags and Examples
NN for common singular nouns; e.g., company, year, market
NNS for common plural nouns; e.g., shares, years, sales, prices, companies
NNP for proper nouns (names); e.g., Bush, Japan, Federal, New York, Corp, Mr., Friday, James A. Talcott ("James NNP A. NNP Talcott NNP")
NNPS for proper plural nouns; e.g., Canadians, Americans, Securities, Systems, Soviets, Democrats

The noun tags in the Penn tag set are: NN, NNS, NNP, and NNPS.

NN is used for common singular nouns, such as *company, year,* and *market.*
NNS is used for common plural nouns, such as *shares, years, sales, prices, and companies.*
NNP is used for proper singular nouns (names), which are the names of people, geographical entities, countries, institutions, and similar, such as *Bush, Japan, Federal, New York, Corp, Mr. Friday, James A. Talcott.* The proper naes consisting of several words are all tagged with the NNP tag; for example: "James NNP A. NNP Talcott NNP" The token "Mr." comes with a person's name and it would be tagged as NNP as well. The words "Federal" and "Corp." are proper nouns as parts of institution or organization names. Somewhat specific to English, the names of days in a week and months, such as "Friday" and "January" are also proper nouns, so tagged as NNP.
NNPS is used for proper plural nouns, such as *Canadians, Americans, Securities, Systems, Soviets, and Democrats.*

Adjectives (JJ, JJR, JJS)

Adjectives describe properties of nouns; for example: red rose, long journey, etc.

Adjectives have three forms and each of them is tagged separately:

Form	Example	Tag
positive	rich	JJ
comparative	richer	JJR
superlative	richest	JJS

In the Brown corpus, the corresponding tags were JJ, JJR, and JJT; while the JJS tag was reserved for the **semantic superlative forms,** such as: chief, main, top, etc. These forms are tagged as JJ in the Penn Treebank corpus.

Comparatives and superlatives of longer adjectives in English are formed as multi-word sequences, such as "more intelligent" and "the most intelligent" for the adjective "intelligent". These sequences are called the **periphrastic adjective forms**. They are tagged as follows:
`more JJR intelligent JJ`
and
`the DT most JJS intelligent JJ`

Verbs (VB, VBP, VBZ, VBG, VBD, VBN)

Verbs are used to describe:

- actions; e.g., throw the stone
- activities; e.g., walked along the river
- or states; e.g., have $50

Verbs can have different forms and they are tagged accordingly:

Tag	Form name	Example
VB	base	eat, be, have, walk, do
VBD	past	ate, said, was, were, had
VBG	present participle	eating, including, according, being
VBN	past participle	eaten, been, expected
VBP	present non-3sg	eat, are, have, do, say, 're, 'm
VBZ	present 3sg	eats, is, has, 's, says

Gerund is a noun which has the same form as the present participle; e.g., 'Walking is fun.'

Verb Features:

- number: singular, plural
- person: 1st, 2nd, 3rd
- tense: present, past, future
- aspect: progressive, perfect
- mood: possibility, subjunctive (e.g. 'They requested that he be banned from driving.')
- participles: present participle, past participle
- voice: passive, active: "He wrote a book." vs. "A book was written by him."

Verb Tenses:

- present: I walk
- infinitive: to walk
- progressive: I am walking
- present perfect I have walked

– past perfect: I have walked

Adverbs (RB, RBR, RBS)

Adverbs modify verbs, as their name suggests, but also other lexical classes, such as adjectives and adverbs. In this way their function is quite heterogeneous. For example, some typical adverbs that can be used to modify verbs are *allegedly* and *quickly*, because they obviously describe whether something happened or how something happened.

Not all adverbs belong to the open group of categories: a group of adverbs called **qualifiers** or **degree adverbs** belong to the closed group. Example of such adverbs are *very* and *not*.

Here are a few examples of adverb usage. An example of an adverbs modifying a verb:

She *often* travels to Las Vegas.

an example of adverbs modifying verbs and adverbs:

Unfortunately, John walked *home extremely slowly yesterday*.

and two examples of adverbs modifying adjectives:

a *very* unlikely event
a *shockingly* frank exchange

Adverb Inflections

Adverbs can have three forms, similarly to adjectives;

Tag	Form	Examples
RB	positive	late, often, quickly
RBR	comparative	later, better, less
RBS	superlative	most, best

The superlative adverbs are tagged as RBT in the Brown corpus.

Adverbial Nouns are nouns that also behave as adverbs. Such nouns are 'home' and 'tomorrow.' For example, we can say

I am going home.

but not

* I am going room.

In the Brown corpus these nouns were tagged as NNR, but in the Penn Treebank corpus they are tagged as NN.

Another noun tag in the Brown corpus that cannot be found in the Penn Treebank corpus is NN$, which was used to denote possessives, like 'people's'; in the Penn Treebank this would be tagged as 'people NNP 's POS'.

14.2 Closed Word Categories

– small, fixed, frequent, functional group
– typically no morphological transformations
– include:
  – determiners, pronouns, prepositions, particles, auxiliaries and modal verbs, qualifiers, conjunctions, numbers, interjections

Determiners (DT)

- articles: the, a, an
- demonstratives:
  - this, that, those
  - some, any
  - either, neither
- quantifiers: all, some

Interrogative Determiners (WDT)

Interrogative determiners are tagged as a separate class. Some examples are: 'what', 'which', 'whatever', and 'whichever'.

Predeterminers (PDT)

- Examples: both, quite, many, all such, half
- Examples in context:
  "half the debt", "all the negative campaign"
- Interesting classifications of determiners (Bond 2001)
  - by linear order: pre-determiners, central determiners, post-determiners
  - by meaning: quantifiers, possessives, determinatives

A Side Note:

Two interesting classifications of determiners were given by Francis Bond in his dissertation "Determiners and Numbers in English contrasted with Japanese, as exemplified in Machine Translation."These classifications are a classification of determiners by linear order, and a classification by meaning. This classification is not in accordance with the Penn tag set; e.g., the numerals are also included in the set of determiners.

Determiners grouped by linear order:

- pre-determiners
  - quantifiers: *all, both*
  - fractions: *half, one-third, ...*
  - multiples: *double, twice, three times, ...*
  - *what* (exclamative: *What a great party!*)
- central determiners
  - articles: *the, a(an), some, any*
  - demonstratives: *this/these*, *that/those*
  - possessive pronouns: *my, your, his, her, its, their, our*
  - possessive phrases: *the king's*, *his friend's*, ...
  - quantifiers *no, some, any, either, neither, another, each, enough, much, more, most, less, a few, a little, many a, several*
  - *which, what* (interrogative: *What sound is that?*)
  - pronouns: *we*, *us*, *you*
- post-determiners
  - cardinal numerals: *one, two, three, ...*
  - fixed-numbers: *dozen, score, ...*
  - quantifiers: *every, many, few, little*
  - emphatic possessive: *own*
  - *such*

Determiners grouped by meaning:

- quantifiers
  - cardinal numerals *one, two, three, ...*
  - other quantifiers *all, both, no, some, any, much, many, few, a few, little, a little, either, neither, another, enough, more, most, less, many a, several*
  - fractions: *half, one-third, ...*
  - multiples: *double, twice, three times, ...*
- possessives
  - possessive pronouns: *my, your, his, her, its, their, our*
  - possessive phrases: *the king's*, *his friend's*, ...
  - emphatic possessive: *own*
- determinatives
  - articles: *the, a/an, some/any*
  - demonstratives: *this/these*, *that/those*
  - *which, what* (interrogative)
  - *what* (exclamative)
  - *such*
  - pronouns: *we*, *us*, *you*

Pronouns (PRP, PRP$)

- PRP for personal pronouns
  - examples: I, you, he, she, it, we, you, they
- PRP tag for accusative case (diff. tag in Brown):
  - examples: me, him, her, us, them
- PRP tag for reflexive pronouns (diff. in Brown):
  - examples: myself, ourselves, ...
- PRP$ tag for possessive pronouns:
  - examples: your, my, her, his, our, their, its
- PRP for second possessives (diff. in Brown):
  - examples: ours, mine, yours, ...

The personal pronouns are tagged with PRP in the Penn tagset. The following are some of the features of pronouns:

- number: singular (sg), plural (pl)
- person: first (1st), second (2nd), third (3rd)
- case: nominative (subject), accusative (object)
- gender: masculine (he), feminine (she), neuter (it)

The singular personal pronouns used to be tagged with PP in the Brown corpus, and the plural personal pronouns were tagged with PPS (we, you, they).

The personal pronouns in accusative case (me, you, him, her, it, us, you, them) have the same PRP tag, while in the Brown corpus they had tag PPO. The reflexive pronouns (myself, ourselves, ...) have the same tag PRP, while they used to be tagged PPL and PPLS in Brown.

Tha tag for **possessive pronouns** is PRP$; e.g., for your, my, her, our, his, their, its.

The **second possessives** (ours, mine, yours, ...) are tagged PRP (they used to be tagged PP$$ in Brown).

Wh-pronouns (WP) and Wh-possessive (WP$)

- wh-pronouns (WP): who, what, whom, whoever, ...
- wh-possessive pronoun (WP$): whose

Prepositions (IN)

Prepositions reflect spatial or time relationships.

Examples: of, in, for, on, at, by, concerning, ...

Particles (RP)

- frequently ambiguous and confused with prepositions
- used to create compound verbs
- examples: put off, take off, give in, take on, "went on for days," "put it off"

Possessive ending (POS)

- possessive clitic: 's
- Example: John's book
- tagged as: John NNP 's POS book NN

Modal Verbs (MD)

- the examples of modal verbs: can, may, could, might, should, will
- and their abbreviations: 'd, 'll
- tag for modal verbs: MD
- negative forms are separated into a modal verb and an adverb 'not' (will be covered); e.g.: 'couldn't' is tagged as "could MD n't RB"
- *Auxiliary verbs* are: be, have, and do; and their different forms
- in Brown: each auxiliary verb has a separate tag
- in Penn Treebank: they are tagged in the same way as common verbs (we will see that later)

Infinitive word 'to' (TO)

- used to denote an infinitive: e.g., to call
- 'na' is marked as TO in 'gonna', 'wanna' and similar; e.g.: "gonna call" is tagged "gon VB na TO call VB"

Qualifiers (RB)

- qualifiers are closed adverbs, and they are tagged as adverbs (RB) (covered later)
- example: not, n't, very
- postqualifiers: enough, indeed

Wh-adverbs (WRB)

Examples: how, when, where, whichever, whenever,...

Conjunctions (CC)

- words that connect phrases
- coordinate conjunctions (tag: CC) connect coordinate phrases:
- examples; and, or, but, yet, plus, versus, ...
- subordinate conjunctions connect phrases where one is subordinate to another
- examples: if, although, that, because, ...

- subordinate conjunctions are tagged as prepositions (IN) in Penn Treebank
- in Brown corpus, they used to be tagged CS

Numbers (CD)

Numbers behave in a similar way to adjectives: they also modify nouns. Here, we distinguish two kinds of numbers: **cardinal numbers** or **cardinals,** and **ordinal numbers** or **ordinals.**

Examples:

- cardinals: 1, 0, 100.34, hundred, etc.
- ordinals: first, second, 3rd, 4th, etc.

Cardinal numbers are tagged as CD.

Ordinal numbers have a separate tag in the Brown corpus—OD. In the Penn Treebank corpus, they are tagged as adjectives—JJ.

Interjections (UH)

Examples: yes, no, well, oh, quack, OK, please, indeed, hello, Congratulations, ...

14.3 Remaining POS Classes

Existential 'there' (EX)

Belongs to closed word category; example: "There/EX are/VBP three/CD classes/NNS per/IN week/NN"

Foreign Words (FW)

Examples: de (tour de France), perestroika, pro, des

List Items (LS)

Examples: 1, 2, 3, 4, a., b., c., first, second, etc.

Punctuation

Examples	Tag	Description
,	,	comma
; : ... – --	:	mid-sentence separator
. ! ?	.	sentence end
({ [<	(	open parenthesis
) }] >	)	closed parenthesis
` `` non-``	``	open quote
' ''	''	closed quote
$ c HK$ CAN$	$	dollar sign
#	#	pound sign
– + & @ * ** ffr	**SYM**	everything else

14.4 Overview of POS Tags

Penn Treebank Part-of-Speech Tags (adapted from [JM])

Tag	Description	example	Tag	Description	Example
CC	coordinating conjunction	*and, but, or*	SYM	symbol	+, %, &
			TO	infinitive 'to'	*to*
CD	cardinal number	*one, two, three*	UH	interjections	*uh, oops*
DT	determiner	*a, the*	VB	verb, base form	*eat*
EX	existential 'there'	*there*	VBD	verb, past tense	*ate*
FW	foreign word	it mea culpa	VBG	verb, present participle	eating
IN	preposition or subordinating conjunction	*of, in, by*	VBN	verb, past participle	eaten
JJ	adjective	*rich*	VBP	verb, non-3sg pres	eat
JJR	adj., comparative	*richer*	VBZ	verb, 3sg pres	eats
JJS	adj., superlative	*richest*	WDT	wh-determiner	*which*
LS	list item marker	*1, 2, a*	WP	wh-pronoun	*what, who*
MD	modal verb	*can should*	WP$	wh-possessive	*whose*
NN	noun, singular or mass	*llama, snow*	WRB	wh-adverb	*how, where*
NNS	noun, plural	*llamas*	$	dollar sign	*$*
NNP	proper noun, singular	*IBM*	#	pound sign	*#*
NNPS	proper noun, plural	*Canadians*	"	left quote	*', "*
PDT	predeterminer	*all, both*	"	right quite	*', "*
POS	possessive ending	*'s*	(	left parenthesis	*(, [*
PRP	personal pronoun	*I, you, we*	)	right parenthesis	*),]*
PRP$	possessive pronoun	*your, one's*	,	comma	*,*
RB	adverb	*quickly, never*	.	sentence-end punc.	*. ! ?*
RBR	adverb, comparative	*faster*	:	mid-sentence punc.	*: ;*
RBS	adverb, superlative	*fastest*			
RP	particle	*up, off*			

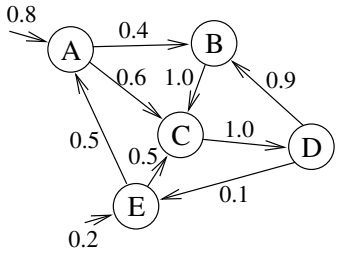
14.5 Some Tagged Examples

```
The/DT grand/JJ jury/NN commented/VBD on/IN
a/DT number/NN of/IN other/JJ topics/NNS ./.

Book/VB that/DT flight/NN ./.

Does/VBZ that/DT flight/NN serve/VB dinner/NN ?/.

It/PRP does/VBZ a/DT first-rate/JJ job/NN ./.

``/`` When/WRB the/DT sell/NN programs/NNS hit/VBP
,/, you/PRP can/MD hear/VB the/DT order/NN
printers/NNS start/VB to/TO go/VB ''/'' on/IN the/DT
Big/NNP Board/NNP trading/NN floor/NN ,/, says/VBZ
one/CD specialist/NN there/RB ./.

``/`` Do/VBP you/PRP make/VB sweatshirts/NNS or/CC
sparkplugs/NNS ?/.
```

15 Hidden Markov Model (HMM)

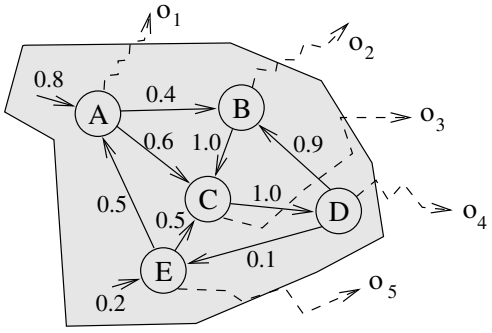
Let us take consider the problem of part-of-speech tagging; i.e., POS tagging. A POS tagger would be an algorithm that takes a tokenized sentence as an input and produces a tagged sentence as the output; i.e., the same sentence in which each token is associated with one of the POS tags. If we want to solve this problem using probabilistic modeling, then it is natural to associate all tokens to probabilistic variables, and their tags as well. There are dependencies between words and their associated tags, but it also seems that tags form some typical sequences, so there are dependencies from each tag to the following tag. This is a motivation for introducing our next model, the Hidden Markov Model.

We will look again at the example of Markov Chain, shown in a section before.



Markov Chain Example

If we assume that the states in such model are not observable, i.e., that they are "hidden," and we can actually observe only an "emitted" symbol, based on a probabilistic distribution for producing observable symbols given a hidden state, we obtain the *Hidden Markov Model* (HMM). An example of such model is represented in the following figure:



Hidden Markov Model Example

15.1 HMM Formal Definition

Slide notes:

HMM Formal Definition

- Five-tuple: (Q, π, a, V, b) (there are other variations)

1. set of states $Q = \{q_1, q_2, \ldots, q_N\}$
2. initial distribution π: $\pi(q)$ for each state q
3. transition probabilities a: $a(q, s)$ for any two states q and s
4. output vocabulary $V = \{o_1, o_2, \ldots, o_m\}$
5. output probability b: $b(q, o)$ for each state q and observable o

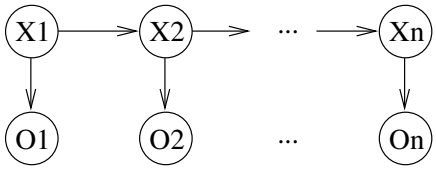
In more precise terms, an HMM (Hidden Markov Model) includes a finite set of hidden states $Q = \{q_1, q_2, \ldots, q_N\}$. A probability distribution $\pi(q)$ ($0 \leq \pi(q) \leq 1$) specifies for each state q the probability that it will be the initial state, where the following constraint $\sum_q \pi(q) = 1$ holds. Instead of making transitions from state to state in a deterministic way, for each pair of states q_i and q_j there is a probability of the transition from state q_i to q_i $a_{ij} = a(q_i, q_j)$, so that $0 \leq a_{ij} \leq 1$ for each parameter a_{ij} and $\sum_s a(q, s) = 1$ for each state q. From each visited state an observable o is generated, where $o \in V$, and V is a finite vocabulary. For an arbitrary state $q \in Q$ and any observable symbol $o \in V$, the output probability $b(q, o)$ that the observable symbol o will be generated is defined, so that for all q $\sum_o b(q, o) = 1$.

We can summarize this into the following definition of a Hidden Markov Model:

Definition 15.1 (Hidden Markov Model) *A Hidden Markov Model is a five-tuple (Q, π, a, V, b), where: Q is a finite set of states $Q = \{q_1, q_2, \ldots, q_N\}$, and $N \geq 1$; $\pi : Q \to [0, 1]$ is the initial probability distribution for the first state, with the constraint $\sum_{q \in Q} \pi(q) = 1$; $a : Q \times Q \to [0, 1]$ is the transition probability, $a(q_i, q_j)$ is also denoted as a_{ij} and it denotes probability of the next state q_j given the current state q_i, so $\sum_{q \in Q} a(s, q) = 1$; V is a finite output vocabulary $V = \{o_1, o_2, \ldots, o_m\}$; and $b : Q \times V \to [0, 1]$ is the output probability, so that $b(q, o)$ is probability of generating the symbol o in the state q, $\sum_{o \in V} b(q, o) = 1$, and we also denote $b(q_i, o_j)$ as b_{ij}.*

HMM Assumption

Given an HMM, we can generate samples by generating an initial state, producing an observable corresponding to that state, and then creating the next state, another observable produced by this state, and so on. For a particular length n, the following graph can be used to illustrate operation of an HMM:



This representation is sometimes called *unrolled* HMM graphical representation, in particular when compared with the DFA-style representation that we saw before. This unrolled representation is similar to the previous graphical representation of the Naïïve Bayes model, and it is called the Belief Network, or Bayesian Network representation. We will later introduce a more general concept of the Bayesian network.

The value of X_1 is the initial state of the HMM, and the value of each consecutive variable X_i is the consecutive state of HMM. The values of variables O_1, O_2, ..., are produced observables.

The HMM assumption formula is:

$$P(X_1, O_1, \ldots, X_n, O_n) = P(X_1) \cdot P(O_1|X_1) \cdot P(X_2|X_1) \cdot P(O_2|X_2) \cdot \ldots \cdot P(X_n|X_{n-1}) \cdot P(O_n|X_n)$$

HMM Application Areas

- Language Modelling
- Acoustic Modelling
- Part-of-Speech tagging (POS tagging)
- Many kinds of sequence tagging (e.g., extracting bio-medical terms)

HMMs are successfully used for acoustic modeling in speech recognition. They are also successfully applied to language modeling and POS tagging, among many applications in NLP.

15.2 POS Tagging using HMM

We will examine now Hidden Markov Model (HMM) in more details, including computational tasks in this moden on the example of POS tagging application.

HMM use in POS Tagging

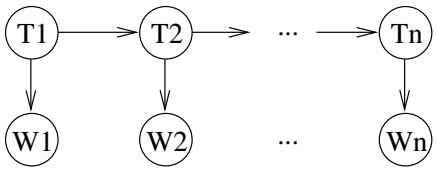
- Hidden states = POS Tags
- Observable variables = words
- In practice: higher-order HMM taggers are used, where the nodes keep a bit longer history (e.g., two previous tags)
- Described in [JM] Sec 5.5 (HMM POS Tagging)

Computational Tasks for HMM

- Evaluation: use HMM assumption formula
- Generation: generate in the order dictated by the "unrolled" graphical representation
- Inference:
  - marginalization, conditioning, completion
  - need for an efficient method (will discuss it)
- Learning: MLE if labeled examples are given

HMM POS Example

To understand better issues involved in efficient HMM inference, we will use a very small, walk-through example in POS tagging. We assume that the hidden internal states of the HMM correspond to correct POS tags of words, while the words correspond to generated observed variables. According to this, a sentence of n words would be associated with the following HMM graph in the unrolled form:



The variables $W_1, \ldots, W_n$ are assigned to words in the sentence, while variables $T_1, \ldots, T_n$ are assigned POS tags. The three probability tables that we mentioned in the definition of HMM are: $\mathrm{P}(T_1)$, $\mathrm{P}(T_{i+1}|T_i)$, and $\mathrm{P}(W_i|T_i)$

Having this in mind, suppose that we are given the following training data:

```
swat V flies N like P ants N
time N flies V like P an D arrow N
```

To accommodate for unseen words, we can assign a special symbol $\star$ to unknown words, and assume that it occurred 0.5 "times" with each tag.

First, we can "learn" the probability of initial states π ($= \mathrm{P}(T_1)$ by counting how many times each state was the

first state in a sequence, and obtain the following counts and the resulting probabilities:

q	counts for $\pi(q)$
D	0
N	1
P	0
V	1

$\Rightarrow$

q	$\pi(q)$
D	0
N	0.5
P	0
V	0.5

We will also denote the initial probability $\pi(q)$ as $\pi(q) = \mathrm{P}(T_1 = q)$.

Similarly, we count the transitions in order to estimate transition probabilities a:

counts for $a(p,q)$	D	N	P	V	sum
D	0	1	0	0	1
N	0	0	1	1	2
P	1	1	0	0	2
V	0	1	1	0	2

$\Rightarrow$

$a(p,q)$	D	N	P	V
D	0	1	0	0
N	0	0	0.5	0.5
P	0.5	0.5	0	0
V	0	0.5	0.5	0

We will also denote the transitional probability $a(p,q)$ as $a(p,q) = PP(T_{i+1} = q | T_i = p)$, which more clearly presents meaning of this probability table.

We will incorporate our smoothing method into learning of the output probabilities by giving a count of 0.5 to any unseen words, marked with '*', to be generated from any tag. This is how we obtain the following counts:

counts for $b(q,o)$	an	ants	arrow	flies	like	swat	time	*	sum
D	1	0	0	0	0	0	0	0.5	1.5
N	0	1	1	1	0	0	1	0.5	4.5
P	0	0	0	0	2	0	0	0.5	2.5
V	0	0	0	1	0	1	0	0.5	2.5

which leads to the following estimated probabilities obtained by dividing numbers in each row with the corresponding sum:

$b(q,o)$	an	ants	arrow	flies	like	swat	time	*
D	$2/3$	0	0	0	0	0	0	$1/3$
N	0	$2/9$	$2/9$	$2/9$	0	0	$2/9$	$1/9$
P	0	0	0	0	$4/5$	0	0	$1/5$
V	0	0	0	$2/5$	0	$2/5$	0	$1/5$

Another way to represent the learned table is as the following conditional probability tables (CPTs):

Generated Tables

T_1	$\mathrm{P}(T_1)$
N	0.5
V	0.5

,

T_{i-1}	T_i	$\mathrm{P}(T_i \mid T_{i-1})$
D	N	1
N	P	0.5
N	V	0.5
P	D	0.5
P	N	0.5
V	N	0.5
V	P	0.5

and

T_i	W_i	$\mathrm{P}(W_i \mid T_i)$
D	an	$2/3 \approx 0.666666667$
D	*	$1/3 \approx 0.333333333$
N	ants	$2/9 \approx 0.222222222$
N	arrow	$2/9 \approx 0.222222222$
N	flies	$2/9 \approx 0.222222222$
N	time	$2/9 \approx 0.222222222$
N	*	$1/9 \approx 0.111111111$
P	like	0.8
P	*	0.2
V	flies	0.4
V	swat	0.4
V	*	0.2

.

In the tables above we did not include zero-probabilities: for example, $\mathrm{P}(T_i = V|T_{i-1} = D)$ is not included since it is equal to 0. The symbol '*' is used to denote any unseen word that may appear in a testing sentence.