CSC520 - Artificial Intelligence

Lecture 23

Dr. Scott N. Gerard

North Carolina State University

Apr 10, 2025

Agenda

- NLP Overview
- Bag-of-words model for classification
- N-gram language models
- Part-of-speech tagging
- Grammar and parsing
- Word vectors

Natural Language Processing

- NLP's goal is for computers to understand natural languages used by humans for communication
- Natural languages lack clearly defined syntax and semantics
  - Bob saw the man with the telescope.
  - Alice saw her duck.
- Application of NLP include:
  - Text classification
  - Text summarization
  - Sentence completion
  - Machine translation
  - Question answering
  - ...

Bag-of-words Model for Classification

- Suppose we want to classify sentences into classes
  - Positive or negative sentiment
  - Spam or ham email
  - Business, finance, sports

- Assume that each sentence is just a bag of words (unigrams) and use Naive Bayes

$$P(Class|w_1, w_2, \ldots, w_n) = \alpha P(w_1|Class)P(w_2|Class)\ldots P(w_n|Class)P(Class)$$

- Example:

$$P(Spam|w_1, w_2, \ldots, w_n) = \alpha P(w_1|Spam)P(w_2|Spam)\ldots P(w_n|Spam)P(Spam)$$
$$P(Ham|w_1, w_2, \ldots, w_n) = \alpha P(w_1|Ham)P(w_2|Ham)\ldots P(w_n|Ham)P(Ham)$$
$$R = \frac{P(Spam|w_1, w_2, \ldots, w_n)}{P(Ham|w_1, w_2, \ldots, w_n)} = \frac{P(Spam)}{P(Ham)} \prod_i \frac{P(w_i|Spam)}{P(w_i|Ham)}$$

If $R > 1$, then classify the message as *Spam*.

Language Model

- Since natural languages are vague, probabilistic approaches are employed
- Language model is a probability distribution over a sequence of words
- Applications of language model
    - Sentence auto-completion
      "Bob drinks ..." {coffee:0.6, tea:0.1, sprite:0.05, ...}
    - Speech-to-text
      P("eyes awe of an") < P("I saw a van")
    - Word correction
      P("Alice boarded a *ship* at the airport") < P("Alice boarded a *plane* at the airport")

N-gram Language Model

- N-gram is a sequence of N words
- Consider the sentence: "I am excited to learn NLP"
  - Unigrams: {I, am, excited, to, learn, NLP}
  - Bigrams: {I am, am excited, excited to, to learn, learn NLP}
  - Trigrams: {I am excited, am excited to, excited to learn, to learn NLP}

N-gram Language Model

- We can write the probability of the sentence using the product rule

$$P(w_1, w_2, \ldots, w_n) = P(w_1) * P(w_2|w_1) * P(w_3|w_1, w_2) * \ldots * P(w_n|w_1, w_2, \ldots, w_{n-1})$$

- Problem: Training corpus may not contain long sentences like: $w_1, w_2, w_3, \ldots, w_n$
- Solution: Use Markov chain approximation: a word is conditionally independent of all other words given the previous N words
- Suppose $N = 1$, then: $P(w_n|w_1, w_2, \ldots, w_{n-1}) = P(w_n|w_{n-1})$

$$P(w_1, w_2, \ldots, w_n) = P(w_1) * P(w_2|w_1) * P(w_3|w_2) * \ldots * P(w_n|w_{n-1})$$

- Suppose $N = 2$, then: $P(w_n|w_1, w_2, \ldots, w_{n-1}) = P(w_n|w_{n-2}, w_{n-1})$

$$P(w_1, w_2, \ldots, w_n) = P(w_1) * P(w_2|w_1) * P(w_3|w_1, w_2) * \ldots * P(w_n|w_{n-2}, w_{n-1})$$

N-gram Language Model

- Create a vocabulary from the training corpus
  - Keep words with frequency greater than some threshold
  - Limit the size of the vocabulary
- For words that are outside the vocabulary, use unknown token `<unk>`
- Use of tokens for start `<s>` and end of sentence `</s>`

Bigram Language Model

Example:

Alice eats apple
Bob eats pizza
Alice drinks tea
Apple is red

<s> Alice eats apple </s>
<s> Bob eats pizza </s>
<s> Alice drinks tea </s>
<s> Apple is <UNK> </s>

$V = \{\text{Alice, Bob, eats, apple, pizza, tea, is}\}$

$$P(\text{Alice}|\text{<s>}) = \frac{count(\text{<s> Alice})}{count(\text{<s>})} = \frac{2}{4}$$

$$P(\text{eats}|\text{Alice}) = \frac{count(\text{Alice eats})}{count(\text{Alice})} = \frac{1}{2}$$

$$P(\text{apple}|\text{eats}) = \frac{count(\text{eats apple})}{count(\text{eats})} = \frac{1}{2}$$

$$P(\text{</s>}|\text{apple}) = \frac{count(\text{apple </s>})}{count(\text{apple})} = \frac{1}{2}$$

$$\begin{aligned} P(\text{Alice eats apple}) &= P(\text{Alice}|\text{<s>}) * P(\text{eats}|\text{Alice}) * P(\text{apple}|\text{eats}) * P(\text{</s>}|\text{apple}) \\ &= \frac{2}{4} * \frac{1}{2} * \frac{1}{2} * \frac{1}{2} = \frac{1}{16} \end{aligned}$$

$$P(\text{Bob drinks sprite}) = P(\text{Bob}|\text{<s>}) * P(\text{drinks}|\text{Bob}) * P(\text{<UNK>}|\text{drinks}) * P(\text{</s>}|\text{<UNK>})$$

Smoothing N-gram Models

- Problem: N-grams of interest may be missing in the training corpus

$$P(w_n|w_{n-1}) = \frac{count(w_{n-1}, w_n)}{count(w_{n-1})}$$

- Solution: Add-one smoothing

$$P(w_n|w_{n-1}) = \frac{count(w_{n-1}, w_n) + 1}{count(w_{n-1}) + |V|}$$

- In general, add-k smoothing

$$P(w_n|w_{n-1}) = \frac{count(w_{n-1}, w_n) + k}{count(w_{n-1}) + k * |V|}$$

- Other smoothing techniques: backoff, linear interpolation, etc.

Part of Speech Tagging

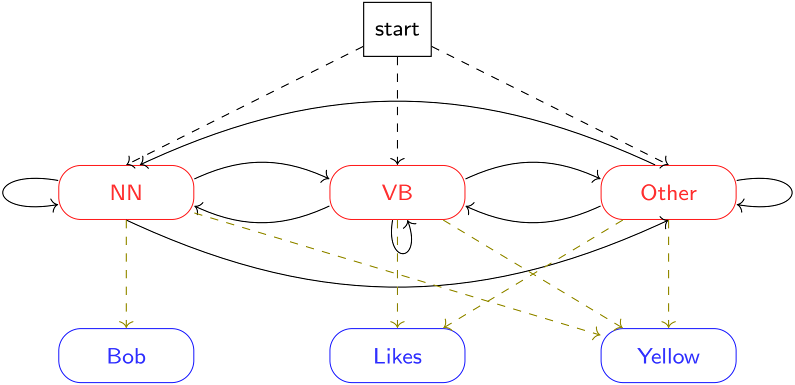
- Assign a part of speech (POS) tag to each word in a sentence
- Example:

Bob	likes	the	yellow	bicycle	with	wide	tires.
NNP	VBZ	DT	JJ	NN	IN	JJ	NNS

Tag	Description
NNP	Proper noun, singular
VBZ	Verb, 3rd-singular present tense
DT	Determiner
JJ	Adjective
NN	Noun, singular
IN	Preposition
NNS	Noun, plural

- Applications include: entity recognition, text-to-speech, question answering, etc.

Hidden Markov Model for POS Tagging



- Use most likely explanation to identify the sequence of POS tags that generated the given sentence (sequence of words)

Grammar

- Grammar is a set of rules; sentences in a language follow those rules
- Unlike formal programming languages, grammar for natural languages is not deterministic
- PCFG (probabilistic context free grammar) is a popular model for natural languages
- Example rule: Syntactic category Adjs can be an Adjective with probability 0.8, or an adjective followed by Adjs with probability 0.2

$$
\begin{aligned}
Adjs \rightarrow & Adjective & [0.80] \\
| & Adjective\ Adjs & [0.20]
\end{aligned}
$$

Parsing Sentences

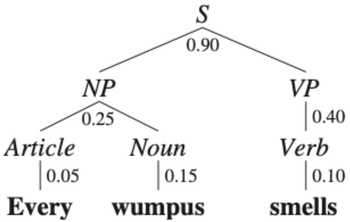
- Process of identifying phrase structure from a sentence following the grammar rules
- Example:

List of items	Rule
S	
$NP\ VP$	$S \rightarrow NP\ VP$
$NP\ VP\ Adjective$	$VP \rightarrow VP\ Adjective$
$NP\ Verb\ Adjective$	$VP \rightarrow Verb$
$NP\ Verb$ **dead**	$Adjective \rightarrow$ **dead**
NP **is dead**	$Verb \rightarrow$ **is**
$Article\ Noun$ **is dead**	$NP \rightarrow Article\ Noun$
$Article$ **wumpus is dead**	$Noun \rightarrow$ **wumpus**
the wumpus is dead	$Article \rightarrow$ **the**

- CYK (Cocke, Younger, Kasami) is an efficient bottom-up probabilistic parsing algorithm
- Grammar must be specified in Chomsky Normal Form
  - Lexical rule: $X \rightarrow \textbf{word}[p]$
  - Syntactic rule: $X \rightarrow Y\ Z[p]$

Parsing Sentences

- Can be formulated as a search problem
  - Start state is list of words
  - Goal state is single item S
  - Actions are the grammar rules
  - Costs are inverse of the probability of rules in the search path
- A* search algorithm can be used with a heuristic, which is usually faster than CYK



Word Vectors

- Words represented in a vector space
  - Vector should capture the word meaning; similar words should be closer in the vector space
  - Word vectors are used as input features in ML models
- Various methods are used to encode words as vectors
  - One-hot vector; does not capture the meaning of the word
  - Word-by-word co-occurrence matrix
  - Word-by-category co-occurrence matrix

Word Vectors

- Word-by-word co-occurrence matrix
  - Number of times a word occurs with another word within distance k; say $k = 2$

Sentence 1: I like to drink coffee in morning.
Sentence 2: Hotel serves coffee in morning.

	i	like	to	drink	coffee	in	morning	hotel	serves
coffee	0	0	1	1	0	2	2	1	1

- Word-by-category co-occurrence matrix
  - Number of times a word occurs in a category

	Finance	Sports	Technology
stocks	800	20	80
investment	1000	200	500
ball	200	1000	100
ai	800	200	1000
gpu	600	20	900

Class Exercise

Alice eats apple
Bob eats pizza
Alice drinks tea
Apple is red

<s> Alice eats apple </s>
<s> Bob eats pizza </s>
<s> Alice drinks tea </s>
<s> Apple is <UNK> </s>

$V = \{\text{Alice, Bob, eats, apple, pizza, tea, is}\}$

$$P(w_n|w_{n-1}) = \frac{count(w_{n-1}, w_n) + 1}{count(w_{n-1}) + |V|}$$

- Compute the following probability using add-one smoothing

$$P(\text{Bob drinks sprite}) = P(\text{Bob}|\texttt{<s>}) * P(\text{drinks}|\text{Bob}) * P(\texttt{<UNK>}|\text{drinks}) * P(\texttt{</s>}|\texttt{<UNK>})$$