N-grams and POS Tagging

Param Ahir

2

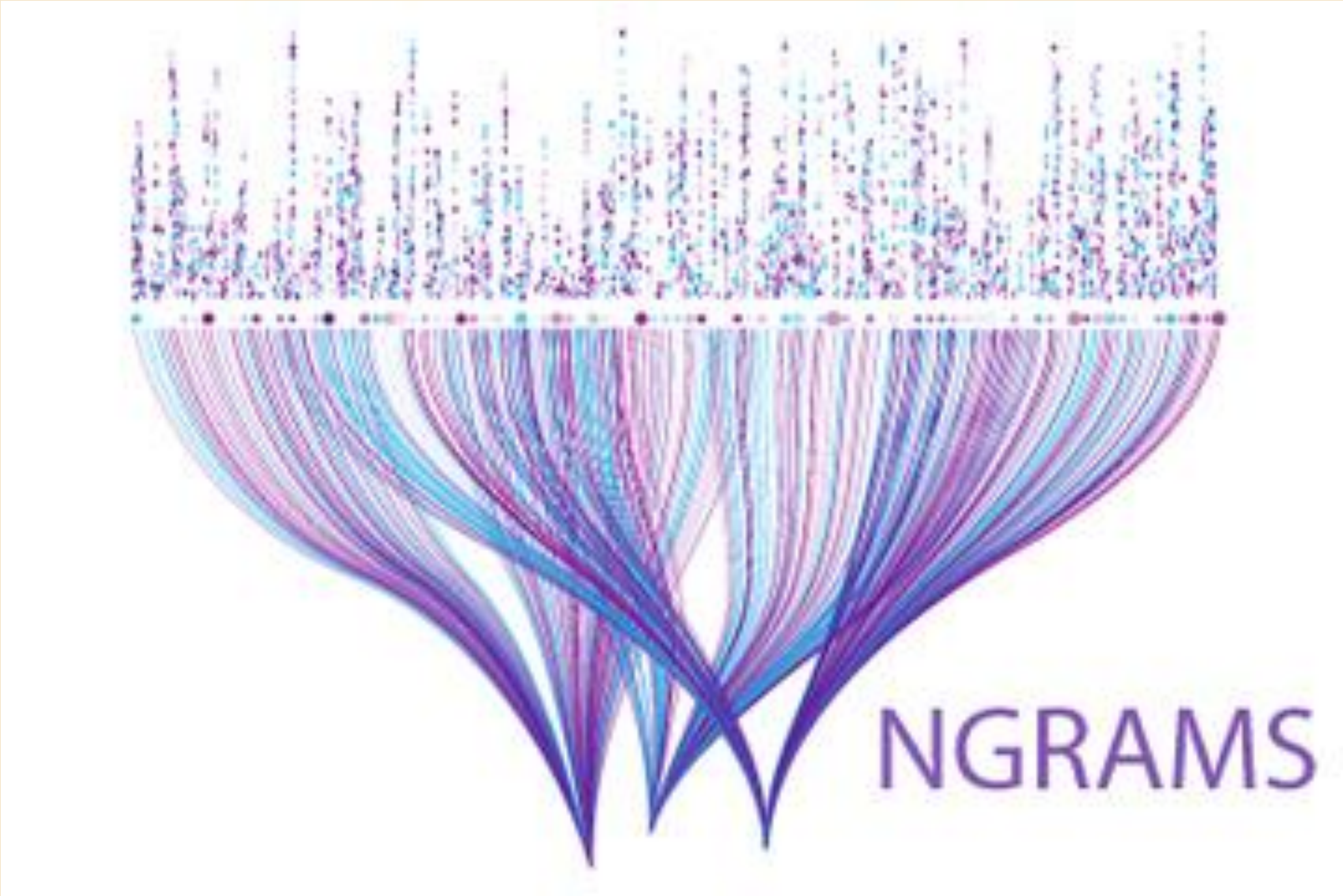
OVERVIEW

1. Introduction
2. N-grams in NLP
3. Unsmoothed vs. Smoothed N-grams
4. N-gram Models
5. Interpolation and Backoff Techniques
6. POS Tagging
7. Methods of POS Tagging
8. HMM vs. MEMM

INTRODUCTION

Natural Language Processing (NLP) explores how computers understand and interact with human language. This presentation focuses on N-grams and POS tagging, which are vital for effective language modeling.

5



Understanding N- grams

N-grams are essential for understanding language patterns and structures. They help in predicting the next item in a sequence, which is vital for various NLP applications.

6

N-grams Comparison

Unsmoothed N-grams can yield zero probabilities for unseen events, limiting their effectiveness. Smoothed N-grams improve this by redistributing probabilities, making models more robust.

Visual representation of N-grams illustrating the difference between unsmoothed and smoothed approaches, highlighting robustness.

N-Gram Models

Estimate probability of each word given prior context.

P(phone | Please turn off your cell)

Number of parameters required grows exponentially with the number of words of prior context.

An N-gram model uses only N–1 words of prior context.

Unigram: P(phone)

Bigram: P(phone | cell)

Trigram: P(phone | your cell)

The *Markov assumption* is the presumption that the future behavior of a dynamical system only depends on its recent history. In particular, in a *kth-order Markov model*, the next state only depends on the *k* most recent states, therefore an N-gram model is a (N–1)-order Markov model.

N-Gram Model Formulas

Word sequences $w_1^n = w_1 \ldots w_n$

Chain rule of probability $P(w_1^n) = P(w_1)P(w_2 \mid w_1)P(w_3 \mid w_1^2) \ldots P(w_n \mid w_1^{n-1}) = \prod_{k=1}^{n} P(w_k \mid w_1^{k-1})$

Bigram approximation $P(w_1^n) = \prod_{k=1}^{n} P(w_k \mid w_{k-1})$

N-gram approximation $P(w_1^n) = \prod_{k=1}^{n} P(w_k \mid w_{k-N+1}^{k-1})$

Estimating Probabilities

N-gram conditional probabilities can be estimated from raw text based on the *relative frequency* of word sequences.

Bigram: $$P(w_n \mid w_{n-1}) = \frac{C(w_{n-1} w_n)}{C(w_{n-1})}$$

To have a consistent probabilistic model, append a unique start (<s>) and end (</s>) symbol to every sentence and treat these as additional words.

N-gram: $$P(w_n \mid w_{n-N+1}^{n-1}) = \frac{C(w_{n-N+1}^{n-1} w_n)}{C(w_{n-N+1}^{n-1})}$$

Generative Model & MLE

An N-gram model can be seen as a probabilistic automata for generating sentences.

Initialize sentence with N–1 <s> symbols
Until </s> is generated do:
    Stochastically pick the next word based on the conditional probability of each word given the previous N –1 words.

Relative frequency estimates can be proven to be *maximum likelihood estimates* (MLE) since they maximize the probability that the model *M* will generate the training corpus *T*.

$$\hat{\lambda} = \underset{\lambda}{\operatorname{argmax}}\, P(T \mid M(\lambda))$$

Example from Textbook

P(<s> i want english food </s>)
= P(i | <s>) P(want | i) P(english | want)
P(food | english) P(</s> | food)
= .25 x .33 x .0011 x .5 x .68 = .000031

- P(<s> i want chinese food </s>)

  = P(i | <s>) P(want | i) P(chinese | want)

  P(food | chinese) P(</s> | food)

  = .25 x .33 x .0065 x .52 x .68 = .00019

Train and Test Corpora

- A language model must be trained on a large corpus of text to estimate good parameter values.
- Model can be evaluated based on its ability to predict a high probability for a disjoint (held-out) test corpus (testing on the training corpus would give an optimistically biased estimate).
- Ideally, the training (and test) corpus should be representative of the actual application data.
- May need to *adapt* a general model to a small amount of new (*in-domain*) data by adding highly weighted small corpus to original training data.

Unknown Words

How to handle words in the test corpus that did not occur in the training data, i.e. *out of vocabulary* (OOV) words?

Train a model that includes an explicit symbol for an unknown word (<UNK>).

- Choose a vocabulary in advance and replace other words in the training corpus with <UNK>.
- Replace the first occurrence of each word in the training data with <UNK>.

Evaluation of Language Models

Ideally, evaluate use of model in end application

- Realistic
- Expensive

Evaluate on ability to model test corpus.

- Less realistic
- Cheaper

Verify at least once that intrinsic evaluation correlates with an extrinsic one.

Perplexity

- **Measure of how well a model “fits” the test data.**
- **Uses the probability that the model assigns to the test corpus.**
- **Normalizes for the number of words in the test corpus and takes the inverse.**

$$PP(W) = \sqrt[N]{\frac{1}{P(w_1 w_2 \ldots w_N)}}$$

- Measures the weighted average branching factor in predicting the next word (lower is better).

Sample Perplexity Evaluation

- **Models trained on 38 million words from the Wall Street Journal (WSJ) using a 19,979 word vocabulary.**
- **Evaluate on a disjoint set of 1.5 million WSJ words.**

	Unigram	Bigram	Trigram
Perplexity	962	170	109

Smoothing

- **Since there are a combinatorial number of possible word sequences, many rare (but not impossible) combinations never occur in training, so MLE incorrectly assigns zero to many parameters (a.k.a. *sparse data*).**
- **If a new combination occurs during testing, it is given a probability of zero and the entire sequence gets a probability of zero (i.e. infinite perplexity).**
- **In practice, parameters are *smoothed* (a.k.a. *regularized*) to reassign some probability mass to unseen events.**
  - Adding probability mass to unseen events requires removing it from seen ones (***discounting***) in order to maintain a joint distribution that sums to 1.

Laplace (Add-One) Smoothing

"Hallucinate" additional training data in which each possible N-gram occurs exactly once and adjust estimates accordingly.

Bigram: $$P(w_n \mid w_{n-1}) = \frac{C(w_{n-1}w_n)+1}{C(w_{n-1})+V}$$

N-gram: $$P(w_n \mid w_{n-N+1}^{n-1}) = \frac{C(w_{n-N+1}^{n-1}w_n)+1}{C(w_{n-N+1}^{n-1})+V}$$

where *V* is the total number of possible (N–1)-grams (i.e. the vocabulary size for a bigram model).

- Tends to reassign too much mass to unseen events, so can be adjusted to add $0<\delta<1$ (normalized by δV instead of V).

Advanced Smoothing

Many advanced techniques have been developed to improve smoothing for language models.

- Good-Turing
- Interpolation
- Backoff
- Kneser-Ney
- Class-based (cluster) N-grams

Model Combination

- **As N increases, the power (expressiveness) of an N-gram model increases, *but* the ability to estimate accurate parameters from sparse data decreases (i.e. the smoothing problem gets worse).**
- **A general approach is to combine the results of multiple N-gram models of increasing complexity (i.e. increasing N).**

Interpolation

Linearly combine estimates of N-gram models of increasing order.

Interpolated Trigram Model:

$$\hat{P}(w_n \mid w_{n-2,} w_{n-1}) = \lambda_1 P(w_n \mid w_{n-2,} w_{n-1}) + \lambda_2 P(w_n \mid w_{n-1}) + \lambda_3 P(w_n)$$

Where: $\sum_i \lambda_i = 1$

- Learn proper values for λ_i by training to (approximately) maximize the likelihood of an independent ***development*** (a.k.a. ***tuning***) corpus.

Backoff

- **Only use lower-order model when data for higher-order model is unavailable (i.e. count is zero).**
- **Recursively back-off to weaker models until data is available.**

$$P_{katz}(w_n \mid w_{n-N+1}^{n-1}) = \begin{cases} P^*(w_n \mid w_{n-N+1}^{n-1}) & \text{if } C(w_{n-N+1}^{n}) > 1 \\ \alpha(w_{n-N+1}^{n-1}) P_{katz}(w_n \mid w_{n-N+2}^{n-1}) & \text{otherwise} \end{cases}$$

Where P* is a discounted probability estimate to reserve mass for unseen events and α's are back-off weights (see text for details).

8

Interpolation & Backoff

Estimates

Both techniques enhance probability estimates.

Techniques

Interpolation merges probabilities from various N-grams.

Models

Backoff utilizes lower-order models for sparse data.

9

POS Tagging Overview

Definition

Part-of-Speech (POS) tagging is a process that assigns word classes to tokens, which is essential for syntactic analysis.

Tagging Schemes

Common tagging schemes include Penn Treebank and Universal Dependencies, widely used in linguistic studies.

10

Methods of POS Tagging

Rule-Based Approach

Utilizes predefined rules to assign tags based on grammar.

Stochastic Approach

Employs probabilistic models to predict tags based on context.

Transformation-Based

Combines rules and statistics for improved accuracy.

Part of Speech

- Each word belongs to a word class. The word class of a word is known as **part-of-speech (POS)** of that word.
- Most POS tags implicitly encode fine-grained specializations of eight basic parts of speech:
  - *noun, verb, pronoun, preposition, adjective, adverb, conjunction, article*
- These categories are based on morphological and distributional similarities (not semantic similarities).
- Part of speech is also known as:
  - *word classes*
  - *morphological classes*
  - *lexical tags*

Part of Speech (cont.)

- A POS tag of a word describes the major and minor word classes of that word.
- A POS tag of a word gives a significant amount of information about that word and its neighbours. For example, a possessive pronoun (my, your, her, its) most likely will be followed by a noun, and a personal pronoun (I, you, he, she) most likely will be followed by a verb.
- Most of words have a single POS tag, but some of them have more than one (2,3,4,…)
- For example, book/noun or book/verb
  - I bought a *book*.
  - Please *book* that flight.

Tag Sets

- There are various tag sets to choose.
- The choice of the tag set depends on the nature of the application.
  - We may use small tag set (more general tags) or
  - large tag set (finer tags).
- Some of widely used part-of-speech tag sets:
  - Penn Treebank has 45 tags
  - Brown Corpus has 87 tags
  - C7 tag set has 146 tags
- In a tagged corpus, each word is associated with a tag from the used tag set.

English Word Classes

- Part-of-speech can be divided into two broad categories:
  - **closed class types** -- such as prepositions
  - **open class types** -- such as noun, verb
- Closed class words are generally also **function words**.
  - Function words play important role in grammar
  - Some function words are: *of*, *it*, *and*, *you*
  - Functions words are most of time very short and frequently occur.
- There are four major open classes.
  - noun, verb, adjective, adverb
  - a new word may easily enter into an open class.
- Word classes may change depending on the natural language, but all natural languages have at least two word classes: *noun* and *verb*.

Nouns

- Nouns can be divided as:
  - *proper nouns* -- names for specific entities such as Ankara, John, Ali
  - *common nouns*
- Proper nouns do not take an article but common nouns may take.
- Common nouns can be divided as:
  - *count nouns* -- they can be singular or plural -- chair/chairs
  - *mass nouns* -- they are used when something is conceptualized as a homogenous group -- snow, salt
- Mass nouns cannot take articles *a* and *an*, and they can not be plural.

Verbs

- Verb class includes the words referring actions and processes.
- Verbs can be divided as:
  - main verbs -- open class -- draw, bake
  - auxiliary verbs -- closed class -- can, should
- Auxiliary verbs can be divided as:
  - copula -- be, have
  - modal verbs -- may, can, must, should
- Verbs have different morphological forms:
  - non-3rd-person-sg eat
  - 3rd-person-sg - eats
  - progressive -- eating
  - past -- ate
  - past participle -- eaten

Adjectives

- Adjectives describe properties or qualities
  - for color -- black, white
  - for age -- young, old
- In Turkish, all adjectives can also be used as noun.
  - kırmızı kitap *red book*
  - kırmızıyı *the red one (ACC)*

Adverbs

- Adverbs normally modify verbs.
- Adverb categories:
  - locative adverbs -- home, here, downhill
  - degree adverbs -- very, extremely
  - manner adverbs -- slowly, delicately
  - temporal adverbs -- yesterday, Friday
- Because of the heterogeneous nature of adverbs, some adverbs such as Friday may be tagged as nouns.

Major Closed Classes

- Prepositions -- on, under, over, near, at, from, to, with
- Determiners -- a, an, the
- Pronouns -- I, you, he, she, who, others
- Conjunctions -- and, but, if, when
- Participles -- up, down, on, off, in, out
- Numerals -- one, two, first, second

Prepositions

- Occur before noun phrases
- indicate spatial or temporal relations
- Example:
  - <u>on</u> the table
  - <u>under</u> chair
- They occur so often. For example, some of the frequency counts in a 16 million word corpora (COBUILD).
  - of 540,085
  - in 331,235
  - for 142,421
  - to 125,691
  - with 124,965
  - on 109,129
  - at 100,169

Particles

- A particle combines with a verb to form a larger unit called
- **phrasal verb**.
  - go on
  - turn on
  - turn off
  - shut down

Articles

- A small closed class
- Only three words in the class: a an the
- Marks definite or indefinite
- They occur so often. For example, some of the frequency counts in a 16 million word corpora (COBUILD).
  - the 1,071,676
  - a 413,887
  - an 59,359
- Almost 10% of words are articles in this corpus.

Conjunctions

- Conjunctions are used to combine or join two phrases, clauses or sentences.
- **Coordinating conjunctions** -- and or but
  - join two elements of equal status
  - Example: you and me
- **Subordinating conjunctions** -- that who
  - combines main clause with subordinate clause
  - Example:
    - <u>I thought</u> *that* <u>you might like milk</u>

Pronouns

- Shorthand for referring to some entity or event.
- Pronouns can be divided:
  - **personal** you she I
  - **possessive** my your his
  - **wh-pronouns** who what -- who is the president?

TagSets for English

- There are popular actual tagsets for part-of-speech
- PENN TREEBANK tagset has 45 tags
  - IN preposition/subordinating conj.
  - DT determiner
  - JJ adjective
  - NN noun, singular or mass
  - NNS noun, plural
  - VB verb, base form
  - VBD verb, past tense
- A sentence from Brown corpus which is tagged using Penn Treebank tagset.
  - The/DT grand/JJ jury/NN commented/VBD on/IN a/DT number/NN of/IN other/JJ topics/NNS ./.

Part of Speech Tagging

- Part of speech tagging is simply assigning the correct part of speech for each in an input sentence
- We assume that we have the following:
  - A set of tags (our tag set)
  - A dictionary that tells us the possible tags for each word (including all morphological variants).
  - A text to be tagged.
- There are different algorithms for tagging.
  - Rule Based Tagging
  - Statistical Tagging (Stochastic Tagging)
  - Transformation Based Tagging

How hard is tagging?

- Most words in English are unambiguous. They have only a single tag.
- But many of most common words are ambiguous:
  - can/verb can/auxiliary can/noun
- The number of word types in Brown Corpus
  - unambiguous (one tag) 35,340
  - ambiguous (2-7 tags) 4,100
    - 2 tags 3760
    - 3 tags 264
    - 4 tags 61
    - 5 tags 12
    - 6 tags 2
    - 7 tags 1
- While only 11.5% of word types are ambiguous, over 40% of Brown corpus tokens are ambiguous.

Rule-Based Part-of-Speech Tagging

- The rule-based approach uses handcrafted sets of rules to tag input sentence.
- There are two stages in rule-based taggers:
  - **First Stage**: Uses a dictionary to assign each word a list of potential parts-of-speech.
  - **Second Stage**: Uses a large list of handcrafted rules to window down this list to a single part-of-speech for each word.
- The ENGTWOL is a rule-based tagger
  - In the first stage, uses a two-level lexicon transducer
  - In the second stage, uses hand-crafted rules (about 1100 rules)

After The First Stage

- Example: He had a book.
- After the first stage:
  - he **he/pronoun**
  - had **have/verbpast** have/auxliarypast
  - a **a/article**
  - book **book/noun** book/verb

Tagging Rule

Rule-1:

if (the previous tag is an article)

then eliminate all verb tags

Rule-2:

if (the next tag is verb)

then eliminate all verb tags

Transformation-Based Tagging

- Transformation-based tagging is also known as - Brill Tagging.
- Similar to rule-based taggers but rules are learned from a tagged corpus.
- Then these learned rules are used in tagging.

How TBL Rules are Applied

- Before the rules are applied the tagger labels every word with its most likely tag.
- We get these most likely tags from a tagged corpus.
- Example:
  - He is expected to race tomorrow
  - he/PRN is/VBZ expected/VBN to/TO <u>race/NN</u> tomorrow/NN
- After selecting most-likely tags, we apply transformation rules.
  - Change NN to VB when the previous tag is TO
  - This rule converts race/NN into race/VB
- This may not work for every case
  - ….. According to race

How TBL Rules are Learned

- We will assume that we have a tagged corpus.
- Brill's TBL algorithm has three major steps.
  - Tag the corpus with the most likely tag for each (unigram model)
  - Choose a transformation that deterministically replaces an existing tag with a new tag such that the resulting tagged training corpus has the lowest error rate out of all transformations.
  - Apply the transformation to the training corpus.
- These steps are repeated until a stopping criterion is reached.
- The result (which will be our tagger) will be:
  - First tags using most-likely tags
  - Then apply the learned transformations

Transformations

- A transformation is selected from a small set of templates.

Change tag a to tag b when

- The preceding (following) word is tagged z.
- The word two before (after) is tagged z.
- One of two preceding (following) words is tagged z.
- One of three preceding (following) words is tagged z.
- The preceding word is tagged z and the following word is tagged w.
- The preceding (following) word is tagged z and the word two before (after) is tagged w.

Basic Results

- We get 91% accuracy just picking the most likely tag.
- We should improve the accuracy further.
- Some taggers can perform 99% percent.

Statistical Part-of-Speech Tagging

- Choosing the best tag sequence $T=t_1,t_2,\ldots,t_n$ for a given word sequence $W = w_1,w_2,\ldots,w_n$ (sentence):

$$\hat{T} = \arg\max_{T \in \tau} P(T \mid W)$$

By Bayes Rule:

$$\hat{T} = \arg\max_{T \in \tau} \frac{P(W \mid T)P(T)}{P(W)}$$

Since P(W) will be same for each tag sequence:

$$\hat{T} = \arg\max_{T \in \tau} P(W \mid T)P(T)$$

Statistical POS Tagging (cont.)

- If we assume a tagged corpus and a trigram language model, then P(T) can be approximated as:

$$P(t_1)P(t_2 \mid t_1)\prod_{i=3}^{n} P(t_i \mid t_{i-2}t_{i-1})$$

To evaluate this formula is simple, we get from simple word counting (and smoothing).

Statistical POS Tagging (cont.)

To evaluate P(W|T), we will make the simplifying assumption that the word depends only on its tag.

$$\prod_{i=1}^{n} P(w_i \mid t_i)$$

So, we want the tag sequence that maximizes the following quantity.

$$P(t_1)P(t_2 \mid t_1)\prod_{i=3}^{n} P(t_i \mid t_{i-2}t_{i-1})\left[\prod_{i=1}^{n} P(w_i \mid t_i)\right]$$

The best tag sequence can be found by Viterbi algorithm.

11

Comparison

01 Hidden Markov Models (HMM) are generative models that focus on modeling joint probabilities, allowing for a comprehensive understanding of the data.

02 In contrast, Maximum Entropy Models (MEMM) are discriminative models that emphasize conditional probabilities, providing a more targeted approach to predictions.

03 Both models serve unique purposes in statistical modeling, with HMMs being useful for sequence data and MEMMs excelling in classification tasks.

Issues in PoS Tagging

1. **Ambiguity**

- **Lexical Ambiguity:** Some words belong to multiple POS categories depending on context (e.g., *"book"* as a noun in *"Read the book"* vs. a verb in *"Book a ticket"*).
- **Syntactic Ambiguity:** The same sequence of words may be tagged differently based on sentence structure.

2. Out-of-Vocabulary (OOV) Words

- POS taggers struggle with new words, rare words, or domain-specific terms (e.g., technical jargon, slang, or newly coined words).

3. Morphological Complexity

- Some languages, such as Turkish, Finnish, or Arabic, have rich morphology where a single word can have multiple affixes, making POS tagging more complex.

4. Lack of Universal Standards

- Different languages and annotation schemes define POS categories differently (e.g., Universal POS (UPOS) vs. Penn Treebank tags).
- The same tagset may be inconsistently applied across different corpora.

Issues in PoS Tagging

5. Dependency on Training Data

- POS taggers rely on labeled datasets, which may introduce biases based on genre, domain, or language.
- Low-resource languages have limited annotated corpora, affecting performance.

6. Handling Code-Switching and Multilingual Text
- POS taggers struggle when sentences contain multiple languages (e.g., *"I went to the tienda to buy milk."*).

7. Errors in Automatic Tagging
- Rule-based and statistical taggers may misclassify words due to incorrect feature selection or training biases.
- Deep learning-based taggers may require large datasets and computational resources.

8. Context-Sensitivity and Long-Distance Dependencies
- Some POS tags depend on broader sentence context (e.g., *"lead"* in *"He will lead the team"* vs. *"This pipe contains lead."*).

9. Domain Adaptation Challenges
- A POS tagger trained on news articles may not perform well on social media text, medical documents, or legal texts due to domain-specific terminology.

Conclusion

01 Understanding N-grams and POS tagging is essential for improving NLP applications. These techniques significantly boost accuracy in processing.

02 By leveraging N-grams, we can analyze language patterns more effectively, leading to better outcomes in various tasks.

03 POS tagging further refines our understanding of language structure, enhancing the overall efficiency of NLP systems.

References

- 1) Jurafsky, D., & Martin, J. H. (2021). Speech and Language Processing. https://web.stanford.edu/~jurafsky/slp3/
- 2) Manning, C. D., & Schütze, H. (1999). Foundations of Statistical Natural Language Processing. https://nlp.stanford.edu/fsnlp/
- 3) Bird, S., Klein, E., & Loper, E. (2009). Natural Language Processing with Python. https://www.nltk.org/book/