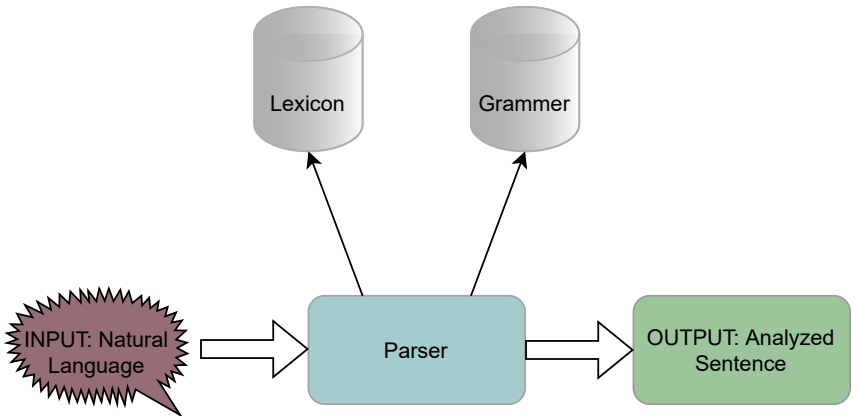
Chapter NLP:IV

IV. Text Representation Models

- Introduction to Text Representation Models
- Bag of Words / Vector Space Model
- Similarity Measures in Natural Language Processing
- Statistical Language Models / Sequence Modelling
- Topic Models
- Topic Models – Evaluation and Variants

Introduction to Statistical Language Models

Classic processing model for language:



Statistical aspects of language

- The lexical entries are not used equally often
- The grammatical rules are not used equally often
- The expected value of certain word forms or word form combinations depends on the technical language used (Sub Language)

Introduction to Statistical Language Models

Question: Is a sentence ***S***, given by a sequence of word forms, a correct sentence of a language ***L***?

How can a computer answer this question?

1. Listing of all sentences of language ***L*** and search for sentence ***S*** in this list. $\rightarrow$ Not possible for Natural Language

2. Create construction rule (grammar) for ***L*** and check sentence ***S***
   - Natural language is very complex (inclusion of dialects, linguistic features)
   - Checking of ***S*** is a complex operation
   - What about semantic correctness?

3. **Probabilistic Language Model:** Calculation of the probability that ***S*** is a correct sentence of ***L***. (probability that ***S*** occurs in a text of language ***L***).

Introduction to Statistical Language Models

Basis of calculation is the distribution of word forms in texts of language ***L***.

- Counting the occurrence of word form combinations
- Determination of the probabilities for the succession of word forms
- Calculation of the probability of the sentence ***S*** from the individual probabilities of the word form sequence of ***S***

In texts of the language ***L*** occur "meaningful" word form combinations

- Consideration of both syntactic and semantic aspects

Statistical Foundations

Notation: (See Probability theory for details and more background)

- Let X be a random variable with a finite set $V(X)$ of m events.
- Let $|X = x|$ be the number of events where X has the value x (i.e. $x \in V(X)$).
- The probability of occurrence of x_i (abbreviation $P(x_i)$ is:

$$P(X = x_i) = \frac{|x_i|}{\sum_{j=1}^{m} |x_j|}$$

Let W be the occurrence of a particular word form w_i from the set of m word forms of a text. The probability of occurrence of the i-th word form w_i is then:

$$P(W = w_i) = \frac{|w_i|}{\sum_{j=1}^{m} |w_j|}$$

Statistical Foundations

Conditional probability: The probability for the occurrence of an event A under the condition that the event B has already occurred is called conditional probability $P(A|B)$.

$$P(A|B) = \frac{P(A,B)}{P(B)}$$

If A and B are independent of each other, then:

$$P(A,B) = P(A) \cdot P(B)$$

The conditional probability of independent events is:

$$P(A|B) = P(A)$$

Example: The conditional probability of the succession of two word forms is:

$$P(W_2 = w_j|W_1 = w_i) = \frac{|W_1 = w_i, W_2 = w_j|}{|W_1 = w_i|}$$

Statistical models of language

Intuition: Assign a probability to all sequences of word forms of length n, i.e.

$$P(W_{1,n} = w_{1,n}) \text{ For all sequences } w_{1,n}.$$

$W_{1,n}$ is a sequence of n random variables $w_1, w_2, \ldots, w_n$, each of which can take any word form as a value, and $w_{1,n}$ is a concrete sequence of word forms.

This sequence can be calculated based on the **generalized Bayesian rule**.

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

Statistical models of language

A sequence of wordforms can be considered a conditional probability.

For two wordforms we have

$$P(w_i, w_j) = P(w_i) \cdot P(w_j|w_i)$$

For three wordforms we have

$$\begin{aligned} P(w_i, w_j, w_k) &= P((w_i, w_j), w_k) && (1)\\ &= P(w_i, w_j) \cdot P(w_k|w_i, w_j) && (2)\\ &= P(w_i) \cdot P(w_j|w_i) \cdot P(w_k|w_i, w_j) && (3) \end{aligned}$$

Statistical models of language

Generalising (3), the probability of any sentence of length n can be stated as:

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

Using the ***maximum likelihood estimate*** we can compute this probability by counting the occurrences of wordforms:

$$P(w_1, w_2, w_3, \ldots, w_n) = \frac{|w_1|}{\sum_{k=1}^{m} |w_k|} \cdot \frac{|w_1, w_2|}{|w_1|} \cdot \frac{|w_1, w_2, w_3|}{|w_1, w_2|} \ldots \frac{|w_1, w_2, w_3, \ldots, w_{n-1}, w_n|}{|w_1, w_2, w_3, \ldots, w_{n-1}|}$$

$$= \frac{|w_1, w_2, w_3, \ldots, w_{n-1}, w_n|}{\sum_{k=1}^{m} |w_k|}$$

Statistical models of language

Example: Text of Language ***L*** = Training Data (For preparation of probabilities)

- Mr. Schuhmann reads a book.
- Mr. Schuhmann eats a loaf of bread.
- Mrs. Mueller eats chicken.

Do the following sentences belong to the language ***L***?

- Mr. Schuhmann eats a loaf of bread.
- Mr. Schuhmann eats chicken.
- Mr. Schuhmann reads a loaf of bread.
- Mrs. Mueller reads a book.

Statistical models of language

For conditional probability of combination of two word forms we defined:

$$P(w_j|w_i) = \frac{|w_i, w_j|}{|w_i|}$$

Examples:

- P(Schumann|Mr.) = |Mr., Schuhmann|/|Mr.|
- P(reads|Schumann) = |Schuhmann, reads|/|Schuhmann|
- P(a|reads) = |a, reads|/|reads|
- P(reads|Mueller) = |Mueller, reads|/|Mueller|

Statistical models of language

For the succession of three word forms we get:

$$P(w_i, w_j, w_k) = P((w_i, w_j), w_k) = P(w_i, w_j) \cdot P(w_k | w_i, w_j) = P(w_i) \cdot P(w_j | w_i) \cdot P(w_k | w_i, w_j)$$

- P(Mr,Schuhmann, reads) = P(Mr.) · P(Schumann|Mr.) · P(reads|Mr., Schuhmann)
- P(Schumann, reads, a) = P(Schuhmann) · P(reads|Schuhmann) · P(a|Schuhmann, reads)
- . . .

Statistical models of language

For the entire set, we obtain according to the ***maximum likelihood estimate***:

$$P(w_1, w_2, w_3, \ldots, w_n) = \frac{|w_1|}{\sum_{k=1}^{m} |w_k|} \cdot \frac{|w_1, w_2|}{|w_1|} \cdot \frac{|w_1, w_2, w_3|}{|w_1, w_2|} \ldots \frac{|w_1, w_2, w_3, \ldots, w_{n-1}, w_n|}{|w_1, w_2, w_3, \ldots, w_{n-1}|}$$

$$= \frac{|w_1, w_2, w_3, \ldots, w_{n-1}, w_n|}{\sum_{k=1}^{m} |w_k|}$$

- P(Mr., Schuhmann, reads, a, book) =
  |Mr., Schuhmann, reads, a, book|/Total word count
- P(Mr., Schuhmann, reads, a, loaf, of, bread) =
  |Mr., Schuhmann, reads, a, loaf, of, bread|/Total word count
- P(Mrs., Mueller, reads, a, book) =
  |Mrs., Mueller, reads, a, book|/Total word count

Statistical models of language

Summary: Approach up till now *not really convincing*

- To compute the probability of any sentence we would have to compute the occurrence of any combination of n wordforms of the total vocabulary N of some language ($n \in N$)
  - Huge list
  - Only possible for some specific corpus, not for language **L** in general
- Insufficient treatment of sentences that do not appear in the trainings corpus
  - Sentences not contained in the trainings corpus are being assigned probability 0 this happens, even if we only change wordorder
  - However, any trainings corpus can only contain a finite number of sentences
  - We do not adequately compute the probability of sentences that do not occur in the trainings corpus (in fact, this is the majority of sentences)
  - We call this: **The model fails to generalize the problem**

Statistical models of language

Evolution 1: Usage of Bi- and Trigrams

- Observation: Relationships between word forms in a sentence are strongly localized. Word forms are grouped into phrases.

- The probability of occurrence of word form w_i is **strongly influenced by remaining word forms of same phrase** and less strongly influenced by word forms of other phrases. However, this is not exploited by the n-gram model.

- It is enough to approximate the probability of occurrence of word forms. Only a few predecessors have to be considered.
  - **Sufficient:** use of only 2 preceding word forms
  - More predecessors hardly bring more accuracy, however, increase the computational effort enormously

Statistical models of language

The *n-gram* model

Assumption that only the preceding $n-1$ word forms are of influence on the probability of the next word form, where $n = 3$ (hence tri-gram).

$$
\begin{aligned}
P(w_n|w_1,\ldots,w_{n-1}) &= P(w_n|w_{n-2},w_{n-1}) \\
P(w_{1,n}) &= P(w_1)\cdot P(w_2|w_1)\cdot P(w_3|w_{1,2})\cdot\ldots\cdot P(w_n|w_{n-2},w_{n-1}) \\
&= P(w_1)\cdot P(w_2|w_1)\cdot\prod_{i=3}^{n} P(w_i|w_{i-2,i-1}) \\
&= \prod_{i=1}^{n} P(w_i|w_{i-2,i-1})
\end{aligned}
$$

Statistical models of language

The *n-gram* model

- Considering at most triple combinations of word forms allows a strong reduction of complexity (regarding possible combinations).

- Probability of sentence ***S*** not immediately 0, if ***S*** does not occur in the training corpus. The main thing is that all combinations of 2 and 3 consecutive word forms in ***S*** occur in the training corpus.

- Nevertheless, difficulty with extremely rare word forms: If such a word form occurs in ***S***, the probability is high that one of the trigrams does not occur in the corpus, therefore **P(S) = 0**.

Statistical models of language

Evolution 2: Smoothing – Generalization and zeros

- Handle problem of extremely rare word forms or word forms that do not occur in the training corpus.

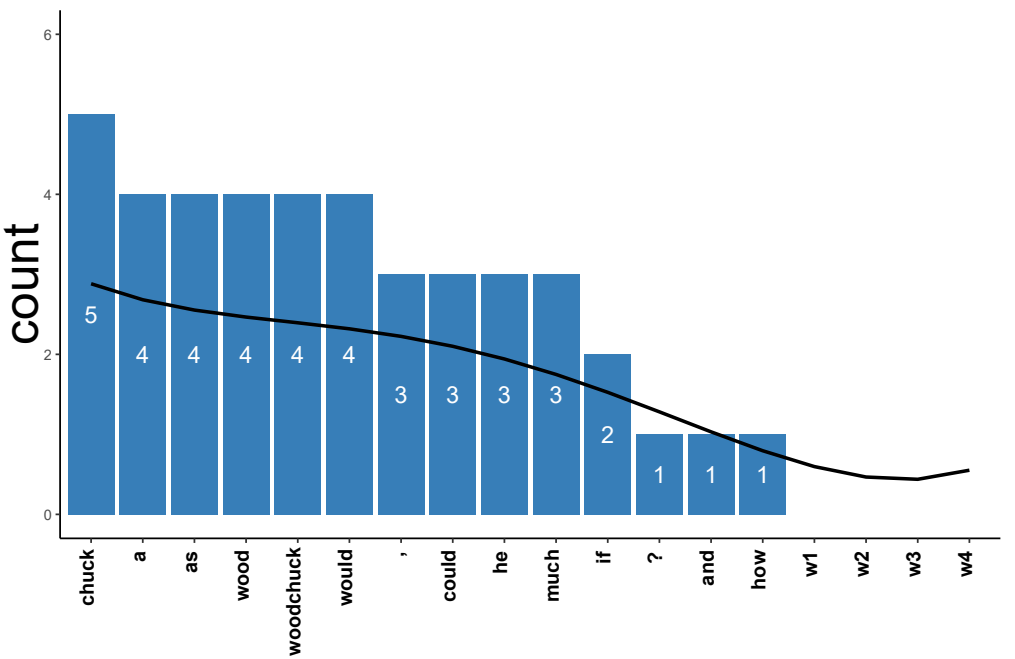
- Evolution
  - Backing Off Smoothing: Falling back to n-grams of lower level in case of missing information
  - Discounting: Subtracting a small amount from the probabilities computed using the training corpus and distributing them to word form combinations that did not occur
    - Good Turing Smoothing
    - Laplace Smoothing
    - Kneser-Ney Smoothing

Statistical models of language

Add-one (Laplace) smoothing

Example: Imagine the following text as as training input for a n-gram language model. Additionally image a situation where you want to obtain the probability of a sentence including the words w_1, w_2, w_3, w_4.

```
How much wood would a woodchuck chuck if a woodchuck could chuck wood?
He would chuck, he would, as much as he could, and chuck as much wood As
a woodchuck would if a woodchuck could chuck wood
```



- We normally get the probability of a word like:

$$P(W = w_i) = \frac{|w_i|}{\sum_{j=1}^{m} |w_j|}$$

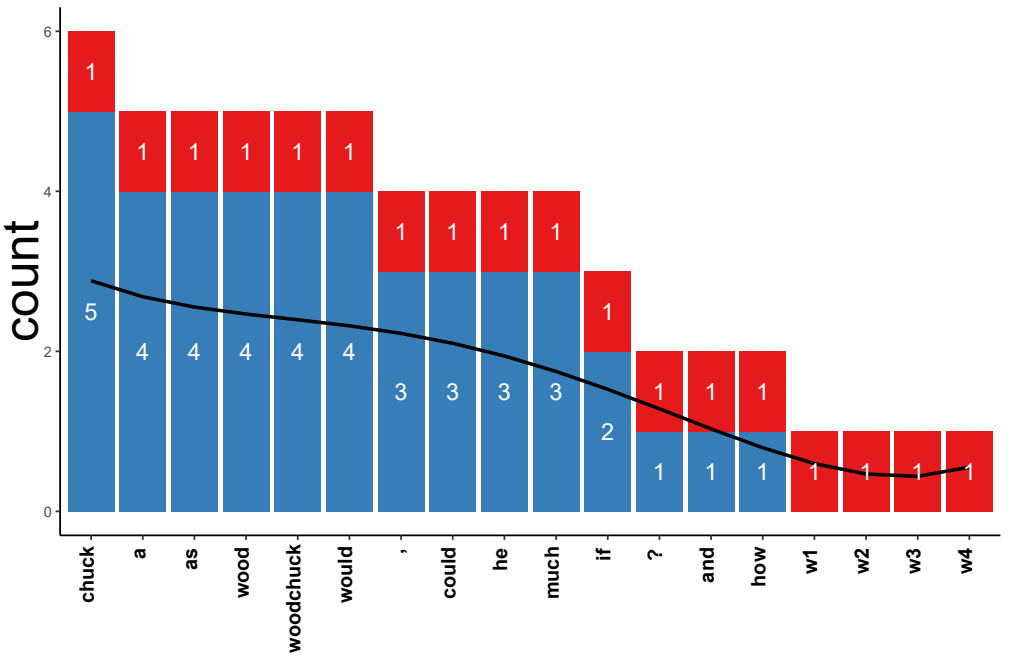
- In this model we assign 0 probability to w_1, w_2, w_3, w_4.

Statistical models of language

Add-one (Laplace) smoothing

Example: Imagine the following text as as training input for a n-gram language model. Additionally image a situation where you want to obtain the probability of a sentence including the words w_1, w_2, w_3, w_4.

```
How much wood would a woodchuck chuck if a woodchuck could chuck wood?
He would chuck, he would, as much as he could, and chuck as much wood As
a woodchuck would if a woodchuck could chuck wood
```



- We can add 1 to each count:

$$P_{\text{add-1}}(W = w_i) = \frac{|w_i| + 1}{\sum_{j=1}^{m} |w_j| + V}$$

- In this model we assign some probability to w_1, w_2, w_3, w_4.

Statistical models of language

Add-one (Laplace) smoothing

In general we could define the calculation of the probabilities of the n-grams as:

$$P_{\text{add-1}}(W = w_i) = \frac{|w_i| + 1}{\sum_{j=1}^{m} |w_j| + V}$$

$$P_{\text{add-1}}(w_n | w_{n-2}, w_{n-1}) = \frac{|w_{n-2}, w_{n-1}, w_n| + 1}{|w_{n-2}, w_{n-1}| + V}$$

We introduce pseudo counts like:

$$|w_i|^* = (|w_i| + 1) \times \frac{\sum_{j=1}^{m} |w_j|}{\sum_{j=1}^{m} |w_j| + V}$$

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

Example

Bigram Model based on *Berkeley Restaurant Project corpus*:

	i	want	to	eat	chinese	food	lunch	spend
i	5	827	0	9	0	0	0	2
want	2	0	608	1	6	6	5	1
to	2	0	4	686	2	0	6	211
eat	0	0	2	0	16	2	42	0
chinese	1	0	0	0	0	82	1	0
food	15	0	15	0	1	4	0	0
lunch	2	0	0	0	0	1	0	0
spend	1	0	1	0	0	0	0	0

Figure 3.1 Bigram counts for eight of the words (out of $V = 1446$) in the Berkeley Restaurant Project corpus of 9332 sentences. Zero counts are in gray.

i	want	to	eat	chinese	food	lunch	spend
2533	927	2417	746	158	1093	341	278

	i	want	to	eat	chinese	food	lunch	spend
i	0.002	0.33	0	0.0036	0	0	0	0.00079
want	0.0022	0	0.66	0.0011	0.0065	0.0065	0.0054	0.0011
to	0.00083	0	0.0017	0.28	0.00083	0	0.0025	0.087
eat	0	0	0.0027	0	0.021	0.0027	0.056	0
chinese	0.0063	0	0	0	0	0.52	0.0063	0
food	0.014	0	0.014	0	0.00092	0.0037	0	0
lunch	0.0059	0	0	0	0	0.0029	0	0
spend	0.0036	0	0.0036	0	0	0	0	0

Figure 3.2 Bigram probabilities for eight words in the Berkeley Restaurant Project corpus of 9332 sentences. Zero probabilities are in gray.

Examples taken from Jurafsky & Martin

Example

Bigram Model after Add-One smoothing:

	i	want	to	eat	chinese	food	lunch	spend
i	6	828	1	10	1	1	1	3
want	3	1	609	2	7	7	6	2
to	3	1	5	687	3	1	7	212
eat	1	1	3	1	17	3	43	1
chinese	2	1	1	1	1	83	2	1
food	16	1	16	1	2	5	1	1
lunch	3	1	1	1	1	2	1	1
spend	2	1	2	1	1	1	1	1

Figure 3.6 Add-one smoothed bigram counts for eight of the words (out of $V = 1446$) in the Berkeley Restaurant Project corpus of 9332 sentences. Previously-zero counts are in gray.

	i	want	to	eat	chinese	food	lunch	spend
i	0.0015	0.21	0.00025	0.0025	0.00025	0.00025	0.00025	0.00075
want	0.0013	0.00042	0.26	0.00084	0.0029	0.0029	0.0025	0.00084
to	0.00078	0.00026	0.0013	0.18	0.00078	0.00026	0.0018	0.055
eat	0.00046	0.00046	0.0014	0.00046	0.0078	0.0014	0.02	0.00046
chinese	0.0012	0.00062	0.00062	0.00062	0.00062	0.052	0.0012	0.00062
food	0.0063	0.00039	0.0063	0.00039	0.00079	0.002	0.00039	0.00039
lunch	0.0017	0.00056	0.00056	0.00056	0.00056	0.0011	0.00056	0.00056
spend	0.0012	0.00058	0.0012	0.00058	0.00058	0.00058	0.00058	0.00058

Figure 3.7 Add-one smoothed bigram probabilities for eight of the words (out of $V = 1446$) in the BeRP corpus of 9332 sentences. Previously-zero probabilities are in gray.

Examples taken from Jurafsky & Martin

Example

Effect of Add-One smoothing on reconstitued transition counts:

	i	want	to	eat	chinese	food	lunch	spend
i	5	827	0	9	0	0	0	2
want	2	0	608	1	6	6	5	1
to	2	0	4	686	2	0	6	211
eat	0	0	2	0	16	2	42	0
chinese	1	0	0	0	0	82	1	0
food	15	0	15	0	1	4	0	0
lunch	2	0	0	0	0	1	0	0
spend	1	0	1	0	0	0	0	0

Figure 3.1 Bigram counts for eight of the words (out of $V = 1446$) in the Berkeley Restaurant Project corpus of 9332 sentences. Zero counts are in gray.

	i	want	to	eat	chinese	food	lunch	spend
i	3.8	527	0.64	6.4	0.64	0.64	0.64	1.9
want	1.2	0.39	238	0.78	2.7	2.7	2.3	0.78
to	1.9	0.63	3.1	430	1.9	0.63	4.4	133
eat	0.34	0.34	1	0.34	5.8	1	15	0.34
chinese	0.2	0.098	0.098	0.098	0.098	8.2	0.2	0.098
food	6.9	0.43	6.9	0.43	0.86	2.2	0.43	0.43
lunch	0.57	0.19	0.19	0.19	0.19	0.38	0.19	0.19
spend	0.32	0.16	0.32	0.16	0.16	0.16	0.16	0.16

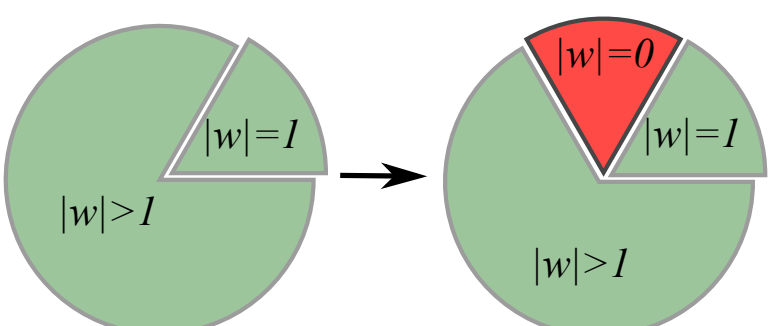
Figure 3.8 Add-one reconstituted counts for eight words (of $V = 1446$) in the BeRP corpus of 9332 sentences. Previously-zero counts are in gray.

Examples taken from Jurafsky & Martin

Statistical models of language

Good-Turing smoothing

- In Add-one too much probability mass is moved! (In fact, we only use it for Naive Bayes with few unknown events)
- **Basic idea:** Use total frequency of events that occur only once to estimate how much mass to shift to unseen events.
- Let N_k be the number of N-grams that occur k times.
  - For bigrams, N_0 is the number of bigrams of count 0, N_1 is the number of bigrams with count 1, etc.



We revisit all counts by:

$$|w|^*_{k-1} = \frac{|w| \cdot N_k}{N_{k-1}}$$

All unseen words get the probability:

$$P(|w|_0) = \frac{N_1}{\sum_{\forall k} N_k}$$

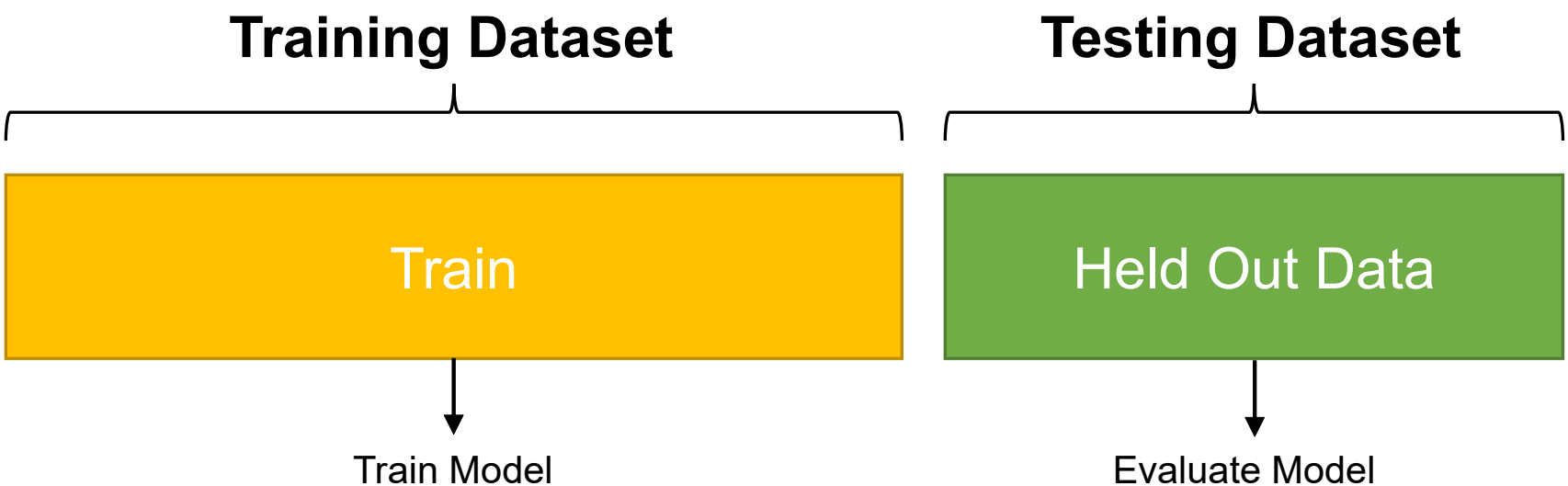
Statistical models of language

Perplexity

- Surprise of the model, w.r.t. new data → a.k.a “Held-out Log Likelihood”
- What is the probability of the words in test data under the pre-trained model?

Assumption:

- The lower the perplexity, the better model captures the prediction of word sequences in the corpus (generalizes to the contents of the corpus)



Statistical models of language

Perplexity

Perplexity is the inverse probability of the Held Out Data, normalized by the number of words:

$$PP(W) = P(w_1, \ldots, w_n)^{-\frac{1}{n}}$$

For trigrams

$$PP(W) = \sqrt[N]{\prod_{i=1}^{N} \frac{1}{P(w_i|w_{i-2}, w_{i-1})}}$$

Minimizing perplexity is the same as maximizing probability

- Perplexity goes down from unigram to trigram models.
- Also smoothing, interpolation etc. raises probability (lowers perplexity) of the model.

Statistical models of language

Linear Interpolation

Idea: Use $(n-1)$-gram probabilities to smooth n-gram probabilities

- **Trigram** combination of the sentence does not occur in the training corpus
  - Exploiting the information about two-word form combination
- **Bigram** combination of the sentence does not occur in the training corpus
  - Exploiting the information about the single word form

Implementation: Utilization of all three sources of information (tri-, bi- and unigrams).

- But different weighting of the sources, as trigram is more informative than bigram etc.

Statistical models of language

Linear Interpolation

Idea: Use $(n-1)$-gram probabilities to smooth n-gram probabilities

Instead of:

$$P(w_n|w_1, w_2, \ldots, w_{n-1}) \approx \frac{|w_{n-2}, w_{n-1}, w_n|}{|w_{n-2}, w_{n-1}|}$$

we set:

$$P(w_n|w_1, w_2, \ldots, w_{n-1}) \approx \lambda_1 \frac{|w_{n-2}, w_{n-1}, w_n|}{|w_{n-2}, w_{n-1}|} + \lambda_2 \frac{|w_{n-1}, w_n|}{|w_{n-1}|} + \lambda_3 \frac{|w_n|}{\sum_m^{k=1} |w_k|}$$

with $\lambda \leq \lambda_i \leq 1$ and $\sum_i \lambda_i = 1$.

- λ can be set manually
- Additionally, automatic methods include Expectation Maximization (EM) algorithms, e.g. Hidden Markov Models

Statistical models of language

Back-off models

Idea: Different models are consulted in order depending on their specificity.

- Back-off n-gram models can back off through progressivly shorter histories (3-gram example):

$$P_{BO}(w_n|w_{n-2},w_{n-1}) = \begin{cases} (1-d_{w_{n-2},w_{n-1},w_n})\frac{|w_{n-2},w_{n-1},w_n|}{|w_{n-2},w_{n-1}|} \\ \quad if \quad |w_{n-2},w_{n-1},w_n| > k \\ \alpha_{w_{n-2},w_{n-1},w_n} P_{BO}(w_n|w_{n-1}) \\ \quad otherwise \end{cases}$$

- d is the probability reserved for unseen n-grams.
- α Probability mass left over in the discounting process.
- **Discounting:** Save off a bit of probability mass from some more frequent events and give it to the events we've never seen

Statistical models of language

Kneser-Ney Smoothing

Observation: "New York" is frequent, but "York" only occurs after "New"

- `I want to drive to friends|York ...`
- `York` is more common than `friends` (at least in news corpora)
- The unigram is useful exactly when we haven't seen this bigram!
- We do not use "How likely is w?"
- We introduce $P_{cont}(w)$: "How likely is w to appear as a novel continuation?"
  - For each word, count the number of bigram types it completes
  - Every bigram type was a novel continuation the first time it was seen

Idea: The unigram probability $P(w)$ should not depend on the frequency of w, but on the **number of contexts** in which w appears.

Statistical models of language

Kneser-Ney Smoothing

We calculate how many times w appears as a novel continuation:

$$P_{cont}(w_i)|\{\forall w_{i-1} : |w_{i-1}, w_i| > 0\}|$$

and normalize by the total number of word bigram types:

$$P_{cont}(w_i) = \frac{|\{\forall w_{i-1} : |w_{i-1}, w_i| > 0\}|}{|\{\forall w_{j-1}, w_j : |w_{j-1}, w_j| > 0\}|}$$

Statistical models of language

Kneser-Ney Smoothing

The Kneser-Ney Smoothing P_{KN} is then:

$$P_{KN}(w_i|w_{i-1}) = \frac{max(|w_{i-1}, w_i| - d, 0)}{|w_{i-1}|} + \lambda(w_{i-1}) \cdot P_{cont}(w_i)$$

λ is a normalizing constant for the probability mass which is discounted. Note, $|\{\forall w_n : |w_{n-1}, w_n| > 0\}|$ is the the number of word types that can follow w_{n-1}

$$\lambda(w_{i-1}) = \frac{d}{|w_{i-1}|}|\{\forall w : |w_{i-1}, w| > 0\}|$$

- The parameter d is a constant which denotes the discount value subtracted from the count of each n-gram, usually between 0 and 1.
- There is a recursion as a generalization for higher order ngrams see Jurafsky & Martin