

C
A
4
0
1
2

Statistical Machine Translation



Week 6 : Language Modelling

Lecturer: Mohammed Hasanuzzaman

E-mail: mohammed.hasanuzzaman@dcu.ie

Lab Tutors: Eva Vanmassenhove and Alberto Poncelas

Recap and Quiz

- Given a reference translation and an MT system output, how can we calculate the WER score?
- What is the main reason for using the Breivety Penalty in the BLEU score equation?

Content

1. **What is a Language Model?**
2. N-gram Language Model
3. Smoothing
4. Evaluation
5. Management of Large Language Models
6. Other Approaches

What is a language?

Can we define a language mathematically?

Deterministic Definition:

A language is the set of all the sentences we can say.

Probabilistic Definition:

A language is the probabilistic distribution of all possible sentences.

Formal Language

- Languages may contain infinite sentences
 - they cannot be defined by enumeration
- One way to define a formal language
 - define the grammar to generate all the sentences in that language
- Another way
 - build a machine to recognise if a sentence is valid in that language

Deterministic Definition

- Given a vocabulary $V = \{v_1, \dots, v_n\}$,
- We use V^* to denote the set of any sequence of words of V ,

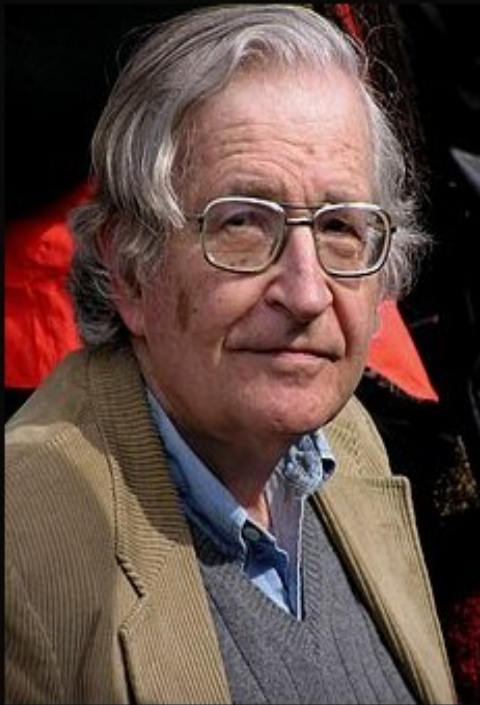
$$V^* = \{W \mid W = w_1, \dots, w_l\}, l \geq 1, w_i \in V$$

- Then a language can be defined as a subset of sentences:

$$L \subseteq V^*$$

- Any element of L is called a *sentence* in the language.

Chomsky Hierarchy



Language is a process of free creation; its laws and principles are fixed, but the manner in which the principles of generation are used is free and infinitely varied. Even the interpretation and use of words involves a process of free creation.

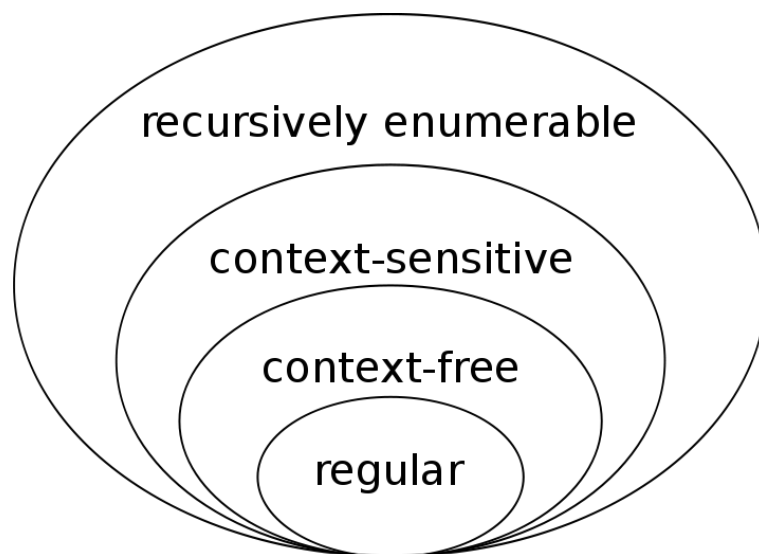
(Noam Chomsky)

izquotes.com

Chomsky Hierarchy

Grammar	Languages	Automaton	Production rules (constraints)
Type-0	Recursively enumerable	Turing machine (semi decidable)	$\alpha \rightarrow \beta$ (no restrictions)
Type-1	Context-sensitive	Linear-bounded non-deterministic Turing machine	$\alpha A \beta \rightarrow \alpha \gamma \beta$
Type-2	Context-free	Non-deterministic pushdown automaton (programming languages)	$A \rightarrow \alpha$
Type-3	Regular	Finite state automaton (regular expressions)	$A \rightarrow a$ and $A \rightarrow aB$

Chomsky Hierarchy



Set inclusions of formal languages described
by
the Chomsky hierarchy

What is a language?

Can we define a language mathematically?

Deterministic Definition:

A language is the set of all the sentences we can say.

Probabilistic Definition:

A language is the probabilistic distribution of all possible sentences

Hard vs soft decision

Probabilistic Definition

- Deterministic definition is not suitable for natural language in most cases
- For some sentences, not easy to say if it is a legal sentence in a certain language:
 - Child languages
 - Tweets
 - Slang
 - ...

Statistical Language Model

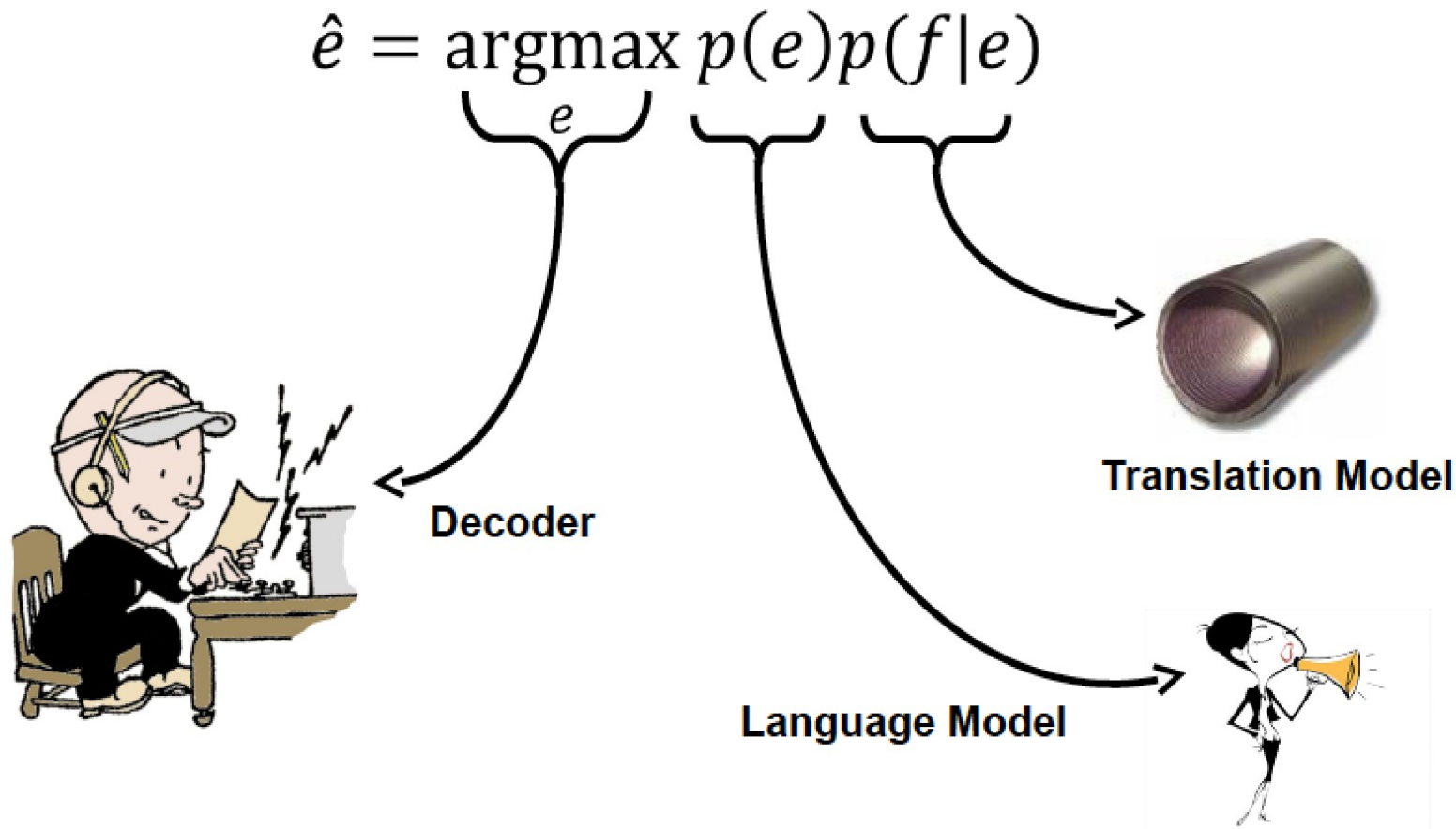
Defined as

$$p(s = w_1 \dots w_n), \forall i: w_i \in V$$

The normalization condition

$$\sum_{s \in V^*} p(s) = 1$$

Noisy Channel Model Revisited



Statistical Language Model

- How can we estimate the probability of a sentence in a specific language?
- Unlike estimating the probability distribution of a dice, we cannot exhaust all the possible sentences in limited samples



Statistical Language Model

- How can we estimate the probability of a sentence in a specific language?
- Unlike estimating the probability distribution of a dice, we cannot exhaust all the possible sentences in limited sample
- Idea
 - break all sentences down to limited substrings (n-grams)
 - Estimate the probability of a sentence by these substrings
 - If a sentence has many plausible substrings then it might be a reasonable sentence



Simplest Language Model

- Simplest way to break down a sentence
 - split it to words
- Thus, the simplest language model

$$p(s = w_1 \dots w_n) = \prod_{i=1}^n p(w_i)$$

- Here the probability of a sentence is just the multiplication of the probability of the words in the sentence
- This model is called **unigram language model**

Word Frequency

- $p(w)$ is word frequency

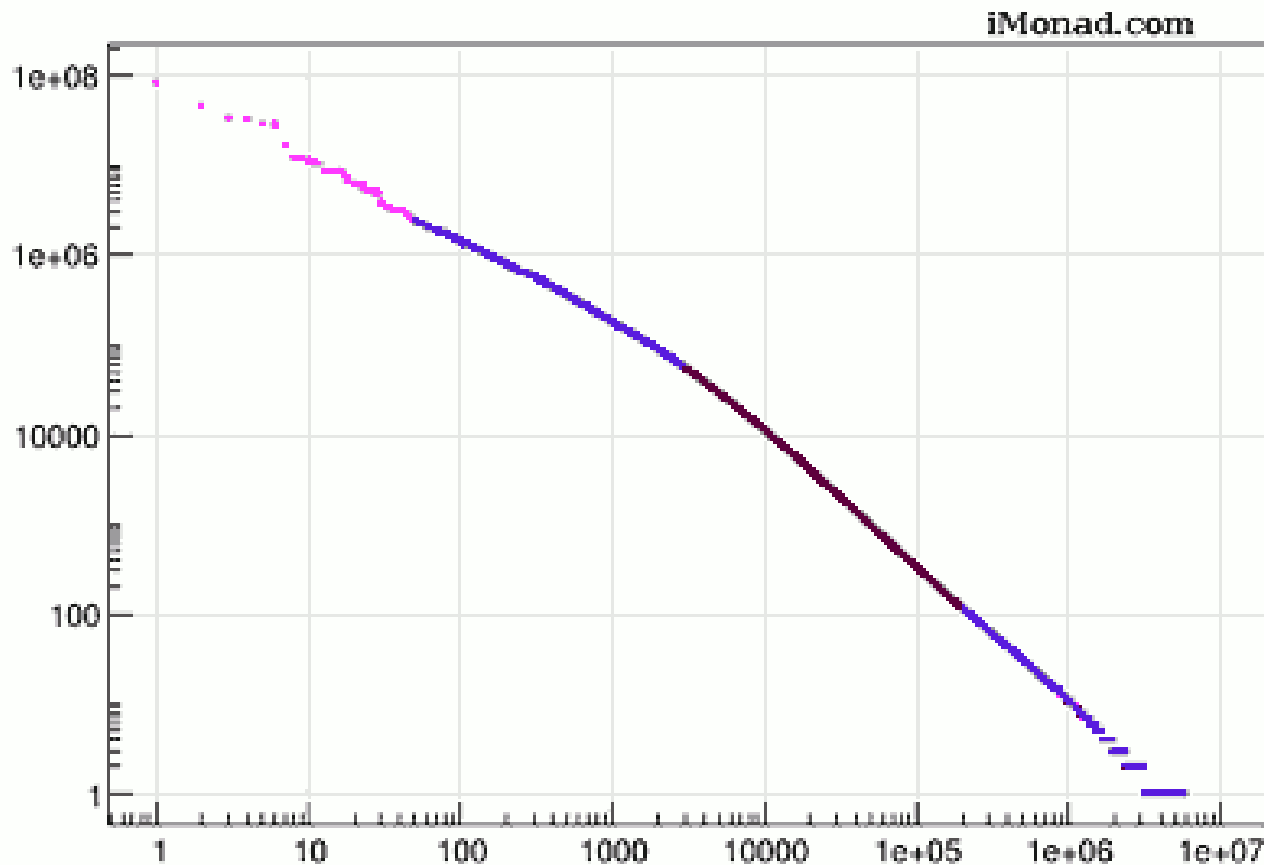
$$p(w) = \frac{\text{occurrences of } w}{\text{number of tokens}}$$

Type	Occurrences	Rank
the	3789654	1st
he	2098762	2nd
[...]		
king	57897	1,356th
boy	56975	1,357th
[...]		
stringyfy	5	34,589th
[...]		
transducionalify	1	123,567th



Word Frequency

Wikipedia Words Frequency List



Zipf's law

The frequency of any word is inversely proportional to its rank in the frequency table

Thus the most frequent word will occur approximately twice as often as the second most frequent word, three times as often as the third most frequent word, etc

Unigram Language Model

$$p(\text{"I am a student."}) \\ = p(\text{"I"}) \times p(\text{"am"}) \times p(\text{"a"}) \times p(\text{"student"}) \times p(\text{"."})$$

- Problems of unigram language model

- Unseen words

$$p(\text{"I am a *zdwi*."}) = ?$$

- No word order

$$p(\text{"I am a student."}) = p(\text{"a I . student am"})$$

Content

1. What is a Language Model?
- 2. N-gram Language Model**
3. Smoothing
4. Evaluation
5. Management of Large Language Models
6. Other Approaches

N-gram Language Model

$$\begin{aligned} p(s = w_1 \dots w_n) &= p(w_1)p(w_2|w_1) \dots p(w_n|w_1 \dots w_{n-1}) \\ &= \prod_{i=1}^n p(w_i|w_1 \dots w_{i-1}) \end{aligned}$$

Markov Assumption

it is assumed that the probability of observing the i^{th} word w_i in the context history of the preceding $i - 1$ words can be approximated by the probability of observing it in the shortened context history of the preceding $n - 1$ words (n^{th} order **Markov property**).

N-gram Language Model

$$p(s = w_1 \dots w_n) = p(w_1)p(w_2|w_1) \dots p(w_n|w_1 \dots w_{n-1})$$

$$= \prod_{i=1}^n p(w_i | \underbrace{w_1 w_2 \dots w_{i-1}}_{\text{The whole history (i - 1 words)}})$$

The whole history ($i - 1$ words)

$$\approx \prod_{i=1}^n p(w_i | \underbrace{w_{i-(N-1)} w_{i-(N-2)} \dots w_{i-1}}_{\text{The shortened history (N - 1 words)}})$$

The shortened history ($N - 1$ words)

N-gram Language Model

1-gram (unigram) Model:

$$p(s = w_1 \dots w_n) = p(w_1) \times p(w_2) \times \dots \times p(w_n)$$

2-gram (bigram) Model:

$$p(s = w_1 \dots w_n) = p(w_1) \times p(w_2|w_1) \times \dots \times p(w_n|w_{n-1})$$

3-gram (trigram) Model:

$$p(s = w_1 \dots w_n) = p(w_1) \times p(w_2|w_1) \times p(w_3|w_1w_2) \times \\ \dots \times p(w_n|w_{n-2}w_{n-1})$$

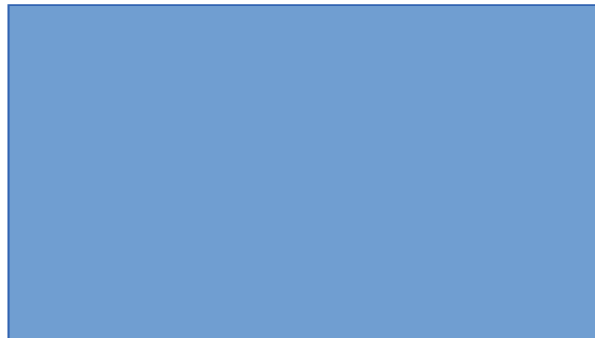
N-gram Language Model

p(I don't like spiders that are poisonous)



bigram

p(I don't like spiders that are poisonous)



trigram

N-gram Language Model

p(I don't like spiders that are poisonous)

$= p(I)$

$\times p(\text{don't}|I)$

$\times p(\text{like}|\text{don't})$

$\times p(\text{spiders}|\text{like})$

$\times p(\text{that}|\text{spiders})$

$\times p(\text{are}|\text{that})$

$\times p(\text{poisonous}|\text{are})$

bigram

p(I don't like spiders that are poisonous)

$= p(I)$

$\times p(\text{don't}|I)$

$\times p(\text{like}|I \text{ don't})$

$\times p(\text{spiders}|\text{don't like})$

$\times p(\text{that}|\text{like spiders})$

$\times p(\text{are}|\text{spiders that})$

$\times p(\text{poisonous}|\text{that are})$

trigram

Deal with Sentence Boundaries

- To better estimate the probabilities of words at sentence boundaries, we usually add `<s>` and `</s>` before and after sentences

N-gram Language Model

$p(<s>I \text{ don't like spiders that are poisonous}</s>)$

$$\begin{aligned} &= p(I|<s>) \\ &\times p(\text{don't}|I) \\ &\times p(\text{like}|\text{don't}) \\ &\times p(\text{spiders}|\text{like}) \\ &\times p(\text{that}|\text{spiders}) \\ &\times p(\text{are}|\text{that}) \\ &\times p(\text{poisonous}|\text{are}) \\ &\times p(</s>|\text{poisonous}) \end{aligned}$$

bigram

$p(<s>I \text{ don't like spiders that are poisonous}</s>)$

$$\begin{aligned} &= p(I|<s>) \\ &\times p(\text{don't}|<s>I) \\ &\times p(\text{like}|I \text{ don't}) \\ &\times p(\text{spiders}|\text{don't like}) \\ &\times p(\text{that}|\text{like spiders}) \\ &\times p(\text{are}|\text{spiders that}) \\ &\times p(\text{poisonous}|\text{that are}) \\ &\times p(</s>|\text{are poisonous}) \\ &\times p(</s>|\text{poisonous}) \end{aligned}$$

trigram

Probability of an n-gram

Suppose we have the phrase “x y” (i.e. word “x” followed by word “y”)
 $p(y|x)$ is the probability that word y follows word x and can be estimated from a corpus as follows

$$p(y|x) = \frac{\text{number-of-occurrences ("x y")}}{\text{number-of-occurrences ("x")}} \quad \text{bigram}$$

Similarly, suppose we have the phrase “x y z”.

$p(z|x y)$ is the probability that word z follows words x and y

$$p(z|x y) = \frac{\text{number-of-occurrences ("x y z")}}{\text{number-of-occurrences ("x y")}} \quad \text{trigram}$$

Estimation with n-gram LM

Trigram LM

History the red (total: 225)

word	c.	prob.
cross	123	
tape	31	
army	9	
card	7	
,	5	

$P(\text{cross}|\text{the,red}) =$

Estimation with n-gram LM

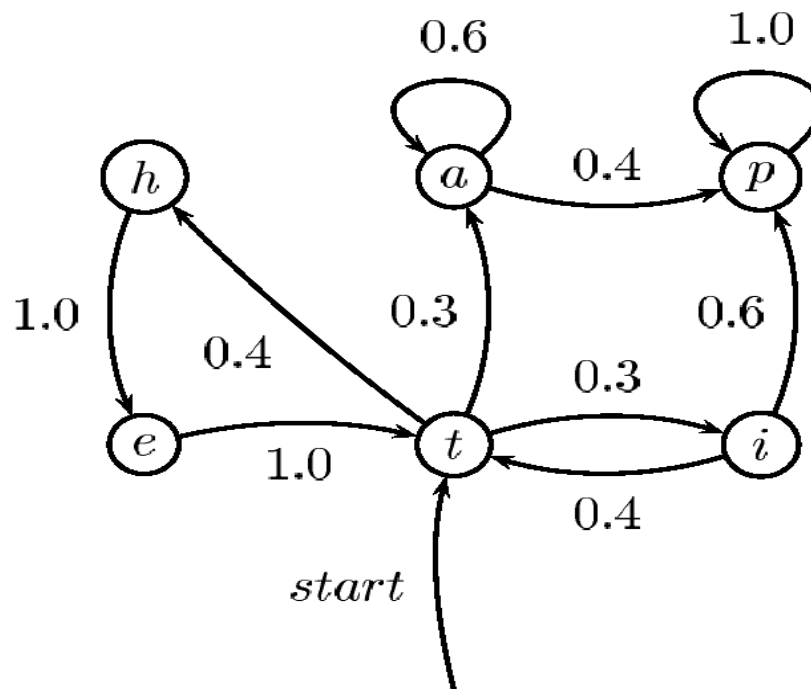
Trigram LM

History the red (total: 225)

word	c.	prob.
cross	123	0.547
tape	31	0.138
army	9	0.040
card	7	0.031
,	5	0.022

$$P(\text{cross}|\text{the,red}) = 123 / 225$$

A character-based bigram LM



$$\begin{aligned} p(t - i - p) &= p(t) \times p(i|t) \times p(p|i) \\ &= 1.0 \times 0.3 \times 0.6 \\ &= 0.18 \end{aligned}$$

Bag Translation

- Take a sentence, cut it up into words, place the words in a bag, and then try to recover the original sentence given the bag
- Use the n-gram model to rank different arrangements of the words in the bag
- Thus, we consider arrangement S better than S' if
 - $P(S) > Pr(S')$

This example comes from:

Brown P F, Cocke J, Pietra S A D, et al. A statistical approach to machine translation[J]. Computational linguistics, 1990, 16(2): 79-85.

Bag Translation

From a collection of 100 sentences, we considered the 38 sentences with fewer than 11 words each (why?)

Exact reconstruction (24 of 38)

Please give me your response as soon as possible.
⇒ Please give me your response as soon as possible.

Reconstruction preserving meaning (8 of 38)

Now let me mention some of the disadvantages.
⇒ Let me mention some of the disadvantages now.

Garbage reconstruction (6 of 38)

In our organization research has two missions.
⇒ In our missions research organization has two.

Unseen n-grams

- We have seen “*i like*” to in our corpus
- We have never seen “*i like to smooth*” in our corpus

$$p(\textit{smooth}|\textit{i like to}) = 0$$

- Any sentence that includes “*i like to smooth*” will be assigned probability 0
- Why is this a bad thing?

Content

1. What is a Language Model?
2. N-gram Language Model
3. **Smoothing**
4. Evaluation
5. Management of Large Language Models
6. Other Approaches

Language Model Smoothing

- Smoothing in language modelling is the process by which unseen events are assigned a non-zero probability
- This is achieved by some combination of
 - Count adjustment
 - Interpolation
 - Back-off

Count Adjustment

`count(tea) = 10`

`count(tea cup) = 3`

`count(tea drinker) = 3`

`count(tea time) = 4`

$$p(x|tea) = \frac{\text{count}(tea\ x)}{\text{count}(tea)}$$

$$p(cup|tea) = .3$$

$$p(drinker|tea) = .4$$

$$p(time|tea) = .3$$

Count Adjustment

What happens when we want to estimate the probability of the phrase “*tea cosy*”?

Count Adjustment

- What happens when we want to estimate the probability of the phrase “*tea cosy*” ?

$$p(\textit{cosy}|\textit{tea}) = 0/10 = 0$$

Just because we haven't seen an event in our sample, it doesn't mean that it can't occur. The smaller our sample size, the less we trust counts derived from it.

Therefore, we need to assign $p(\textit{cosy}|\textit{tea})$ a non-zero probability.

Count Adjustment

Remember that for our model to be sound

$$\sum_x p(x|tea) = 1$$

Count Adjustment

So, for $p(\textit{cosy}|\textit{tea})$ not to be zero, we have to adjust downwards the probability of our seen events:

- $p(\textit{time}|\textit{tea})$
- $p(\textit{cup}|\textit{tea})$
- $p(\textit{drinker}|\textit{tea})$

How do we do that?

Count Adjustment

So, for $p(\textit{cosy}|\textit{tea})$ not to be zero, we have to adjust downwards the probability of our seen events:

- $p(\textit{time}|\textit{tea})$
- $p(\textit{cup}|\textit{tea})$
- $p(\textit{drinker}|\textit{tea})$

How do we do that?

- Add-one smoothing
- Add- α smoothing
- Good-Turing smoothing

Add-One Smoothing

For all possible events, add the count of one.

$$p = \frac{c+1}{n+v}$$

In case of n-gram language model:

$$p = p(w_n | w_1, \dots, w_{n-1})$$

c = count of n-gram $(w_1, \dots, w_{n-1}, w_n)$ in corpus

n = count of history (w_1, \dots, w_{n-1}) in corpus

v = vocabulary size

Add-One Smoothing

Words in language= $\{tea, time, cup, drinker, cosy\}$

Vocabulary size =5

Observed events =

<i>tea time,</i>	<i>tea time,</i>	<i>tea time,</i>	<i>tea cup,</i>
<i>tea cup,</i>	<i>tea cup,</i>	<i>tea drinker,</i>	<i>tea drinker,</i>
<i>tea drinker,</i>	<i>tea drinker</i>		

	tea	time	cup	drinker	cosy	SUM
tea	0	3	3	4	0	10
time	0	0	0	0	0	0
cup	0	0	0	0	0	0
drinker	0	0	0	0	0	0
cosy	0	0	0	0	0	0

Add-One Smoothing

	tea	time	cup	drinker	cosy	SUM
tea	0+1=1	3+1=4	3+1=4	4+1=5	0+1=1	15
time	0+1=1	0+1=1	0+1=1	0+1=1	0+1=1	5
cup	0+1=1	0+1=1	0+1=1	0+1=1	0+1=1	5
drinker	0+1=1	0+1=1	0+1=1	0+1=1	0+1=1	5
cosy	0+1=1	0+1=1	0+1=1	0+1=1	0+1=1	5

$$\begin{aligned}
 p(\text{cup}|\text{tea}) &= (3 + 1)/(10 + 5) = 4/15 \\
 p(\text{time}|\text{tea}) &= (3 + 1)/(10 + 5) = 4/15 \\
 p(\text{drinker}|\text{tea}) &= (4 + 1)/(10 + 5) = 5/15 \\
 p(\text{cosy}|\text{tea}) &= (0 + 1)/(10 + 5) = 1/15 \\
 p(\text{tea}|\text{tea}) &= (0 + 1)/(10 + 5) = 1/15
 \end{aligned}$$

$$\begin{aligned}
 p(\text{cup}|\text{cup}) &= 1/5 \\
 p(\text{time}|\text{cup}) &= 1/5 \\
 p(\text{drinker}|\text{cup}) &= 1/5 \\
 p(\text{cosy}|\text{cup}) &= 1/5 \\
 p(\text{tea}|\text{cup}) &= 1/5
 \end{aligned}$$

Add-One Smoothing

$$p(\textit{tea cozy})$$

$$= p(\textit{tea})p(\textit{cozy}|\textit{tea})$$

$$= \frac{10}{20} \times \frac{1}{15} = \frac{1}{30}$$

Here we use unigram language model to estimate $p(\textit{tea})$.

Deal with Sentence Boundaries

If we add sentence boundaries to observed events:

Observed events =

<s>tea time</s>, <s>tea time</s>, <s>tea time</s>, <s>tea cup</s>, <s>tea cup</s>, <s>tea cup</s>, <s>tea drinker<s>, <s>tea drinker</s>, <s>tea drinker</s>, <s>tea drinker</s>

	tea	time	cup	drinker	cosy	</s>	SUM
<s>	10	0	0	0	0	0	10
tea	0	3	3	4	0	0	10
time	0	0	0	0	0	3	0
cup	0	0	0	0	0	3	0
drinker	0	0	0	0	0	4	0
cosy	0	0	0	0	0	0	0

Deal with Sentence Boundaries

Add-one Smoothing:

	tea	time	cup	drinker	cosy	</s>	SUM
<s>	10+1=11	0+1=1	0+1=1	0+1=1	0+1=1	0+1=1	16
tea	0+1=1	3+1=4	3+1=4	4+1=5	0+1=1	0+1=1	16
time	0+1=1	0+1=1	0+1=1	0+1=1	0+1=1	3+1=4	9
cup	0+1=1	0+1=1	0+1=1	0+1=1	0+1=1	3+1=4	9
drinker	0+1=1	0+1=1	0+1=1	0+1=1	0+1=1	4+1=5	10
cosy	0+1=1	0+1=1	0+1=1	0+1=1	0+1=1	0+1=1	6

$$\begin{aligned}p(\text{cup}|\text{tea}) &= (3 + 1)/(10 + 6) = 4/16 \\p(\text{time}|\text{tea}) &= (3 + 1)/(10 + 6) = 4/16 \\p(\text{drinker}|\text{tea}) &= (4 + 1)/(10 + 6) = 5/16 \\p(\text{cosy}|\text{tea}) &= (0 + 1)/(10 + 6) = 1/16 \\p(\text{tea}|\text{tea}) &= (0 + 1)/(10 + 6) = 1/16\end{aligned}$$

$$\begin{aligned}p(\text{cup}|\text{cup}) &= 1/6 \\p(\text{time}|\text{cup}) &= 1/6 \\p(\text{drinker}|\text{cup}) &= 1/6 \\p(\text{cosy}|\text{cup}) &= 1/6 \\p(\text{tea}|\text{cup}) &= 1/6\end{aligned}$$

Deal with Sentence Boundaries

$$p(<s>tea\ cozy</s>)$$

$$= p(tea|<s>)p(cozy|tea)p(</s>|cozy)$$

$$= \frac{11}{16} \times \frac{1}{16} \times \frac{1}{6} = \frac{11}{1536}$$

Add-One Smoothing

But there are many more unseen n-grams than seen n-grams.

Example: Europarl 2-bigrams:

- 86,700 distinct words
- $86,700^2 = 7,516,890,000$ possible bigrams (v)
- but only about 30,000,000 words (and bigrams) in corpus

Seen n-grams are assigned too low a probability!

Add- α Smoothing

Instead of adding one to the count, we add a lower value α (<1)

$$p = \frac{c + \alpha}{n + \alpha v}$$

How do we determine a *good* value for α ?

Good-Turing Smoothing

Adjust actual counts r to expected counts r^* using the following:

$$r^* = (r + 1) \frac{N_{r+1}}{N_r}$$

where N_r is the number of n-grams with count r in the corpus (count of counts). N_0 is the total number of n-grams in the corpus.

Smoothing: the story so far

- The smoothing techniques covered so far work by adjusting (lowering) the counts of seen n-grams so that unseen n-grams get assigned some of the probability mass
- Other techniques
 - Interpolation
 - Back-off
- These are based on the idea of combining language models of different orders

Combining language models

- In given corpus, we may never observe
 - Scottish beer drinkers
 - Scottish beer eaters
- Both have count 0
- The smoothing methods covered so far will assign them the same probability

Combining language models

- In given corpus, we may never observe
 - Scottish beer drinkers
 - Scottish beer eaters
- Both have count 0
- The smoothing methods covered so far will assign them the same probability
- It would be useful to look at bigram frequencies
 - beer drinkers
 - beer eaters

Interpolation

Higher and lower order n-gram models have different strengths and weaknesses

- high-order n-grams
 - sensitive to more context
 - sparse counts
- low-order n-grams
 - very limited context
 - have robust counts

Interpolation

- Combine them

$$\begin{aligned}
 p(w_3 | w_1, w_2) &= \lambda_1 p_1(w_3) \\
 &+ \lambda_2 p_2(w_3 | w_2) \\
 &+ \lambda_3 p_3(w_3 | w_1, w_2)
 \end{aligned}$$

Where: $\lambda_1 + \lambda_2 + \lambda_3 = 1$

- Recursively

$$\begin{aligned}
 p_n^I(w_i | w_{i-n+1}, \dots, w_{i-1}) &= \lambda_{w_{i-n+1}, \dots, w_{i-1}} p_n(w_i | w_{i-n+1}, \dots, w_{i-1}) + \\
 &+ (1 - \lambda_{w_{i-n+1}, \dots, w_{i-1}}) p_{n-1}^I(w_i | w_{i-n+2}, \dots, w_{i-1})
 \end{aligned}$$

Backoff

- Backoff is slightly different
- The basic idea is to trust the highest order language model that contains the n-gram

Backoff

- Given a trigram, bigram and unigram language model
 - **if** trigram count > 0 :
 - Use it
 - **else if** bigram count > 0 :
 - Use it
 - **else**:
 - Use the unigram count

Other Considerations

Diversity of Predicted Words

- Consider the English words *spite* and *constant*
 - both occur 993 times in Europarl corpus
 - only 9 different words follow *spite*
 - almost always followed by *of* (979 times), due to expression *in spite of*
 - 415 different words follow *constant*
 - most frequent: *and* (42 times), *concern* (27 times), *pressure* (26 times), but huge tail of *singletons*: 268 different words

Other Considerations

Diversity of Predicted Words

- More likely to see new bigram that starts with *constant* than *spite*
- Witten-Bell smoothing takes this into account

Witten-Bell smoothing

- Recursive interpolation

$$p_n^I(w_i | w_{i-n+1}, \dots, w_{i-1}) = \lambda_{w_{i-n+1}, \dots, w_{i-1}} p_n(w_i | w_{i-n+1}, \dots, w_{i-1}) + \\ + (1 - \lambda_{w_{i-n+1}, \dots, w_{i-1}}) p_{n-1}^I(w_i | w_{i-n+2}, \dots, w_{i-1})$$

- Number of possible extensions of a history

$$N_{1+}(w_1, \dots, w_{n-1}, \bullet) = |\{w_n : c(w_1, \dots, w_{n-1}, w_n) > 0\}|$$

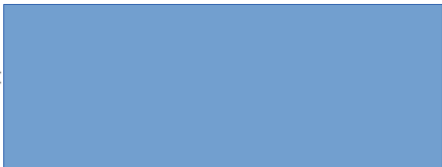
- Lambda parameters

$$1 - \lambda_{w_1, \dots, w_{n-1}} = \frac{N_{1+}(w_1, \dots, w_{n-1}, \bullet)}{N_{1+}(w_1, \dots, w_{n-1}, \bullet) + \sum_{w_n} c(w_1, \dots, w_{n-1}, w_n)}$$


Witten-Bell smoothing

- Example
 - No occurrences spite and constant (993)
 - Words following spite (9) and constant (415)

$$1 - \lambda_{spite} = \frac{N_{1+}(spite, \bullet)}{N_{1+}(spite, \bullet) + \sum_{w_n} c(spite, w_n)}$$

$$=$$


$$1 - \lambda_{constant} = \frac{N_{1+}(constant, \bullet)}{N_{1+}(constant, \bullet) + \sum_{w_n} c(constant, w_n)}$$

$$=$$


Witten-Bell smoothing

- Example
 - No occurrences spite and constant (993)
 - Words following spite (9) and constant (415)

$$\begin{aligned}1 - \lambda_{spite} &= \frac{N_{1+}(spite, \bullet)}{N_{1+}(spite, \bullet) + \sum_{w_n} c(spite, w_n)} \\&= \frac{9}{9 + 993} = 0.00898\end{aligned}$$

$$\begin{aligned}1 - \lambda_{constant} &= \frac{N_{1+}(constant, \bullet)}{N_{1+}(constant, \bullet) + \sum_{w_n} c(constant, w_n)} \\&= \frac{415}{415 + 993} = 0.29474\end{aligned}$$

Other Considerations

- Consider the word *York*
 - fairly frequent word in Europarl corpus, occurs 477 times
 - as frequent as *foods*, *indicates* and *providers*
 - in unigram language model: a respectable probability

Other Considerations

- However, it almost always directly follows *New* (473 times, out of 477)
- Recall: unigram model only used if the bigram model inconclusive
- *York* unlikely second word in unseen bigram
- In backoff unigram model, *York* should have low probability
- Kneser-Ney smoothing takes this into account

Content

1. What is a Language Model?
2. N-gram Language Model
3. Smoothing
- 4. Evaluation**
5. Management of Large Language Models
6. Other Approaches

Evaluating Language Models

The quality of a language model can be estimated using perplexity

$$PP = 2^{H(P_{LM})}$$

which in turn is based on the notion of cross-entropy

$$H(P_{LM}) = -\frac{1}{n} \log P_{LM}(w_1 \dots w_n)$$

Example

- Say we wanted to measure how well a language model p_{LM} predicts the following sentence:

I would like to commend the rapporteur on his work.

A low perplexity indicates the probability distribution is good at predicting the sample.

Prediction	p_{LM}	$-\log_2 p_{LM}$
(I </s><s>)	0.109	3.197
(would <s> I)	0.144	2.791
(like I would)	0.489	1.031
(to would like)	0.905	0.144
(commend like to)	0.002	8.794
(the to recommend)	0.472	1.084
(rapporteur recommend the)	0.147	2.763
(on the rapporteur)	0.056	4.15
(his rapporteur on)	0.194	2.367
(work on his)	0.089	3.498
(. his work)	0.29	1.785
(</s> work .)	0.99999	0.000014
	average	2.634

Here is
how we
would
proceed

N-Gram Comparison

word	unigram	bigram	trigram	4-gram
i	6.684	3.197	3.197	3.197
would	8	2.884	2.791	2.791
like	9	2.026	1.031	1.29
to	5	0.402	0.144	0.113
commend	15	12.335	8.794	8.633
the	4	1.402	1.084	0.88
rapporteur	11	7.319	2.763	2.35
on	7	4.14	4.15	1.862
his	11	7.316	2.367	1.978
work	10	4.816	3.498	2.394
.	5	3.02	1.785	1.51
</s>	4.828	0.005	0	0
average	8.051	4.072	2.634	2.251
perplexity	265.136	16.817	6.206	4.758

Content

1. What is a Language Model?
2. N-gram Language Model
3. Smoothing
4. Evaluation
- 5. Management of Large Language Models**
6. Other Approaches

Managing very large language models

- Millions to billions of words are easy to get (trillions of English words available on the web)
- But: huge language models do not fit into RAM

Delete singletons

Europarl data:

Order	Unique n-grams	Singletons
unigram	86,700	33,447 (38.6%)
bigram	1,948,935	1,132,844 (58.1%)
trigram	8,092,798	6,022,286 (74.4%)
4-gram	15,303,847	13,081,621 (85.5%)
5-gram	19,882,175	18,324,577 (92.2%)

Delete singletons of higher order n-grams

Other Techniques

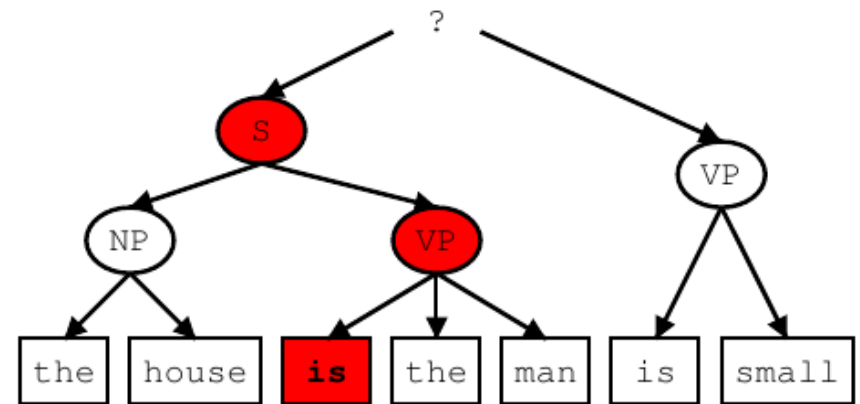
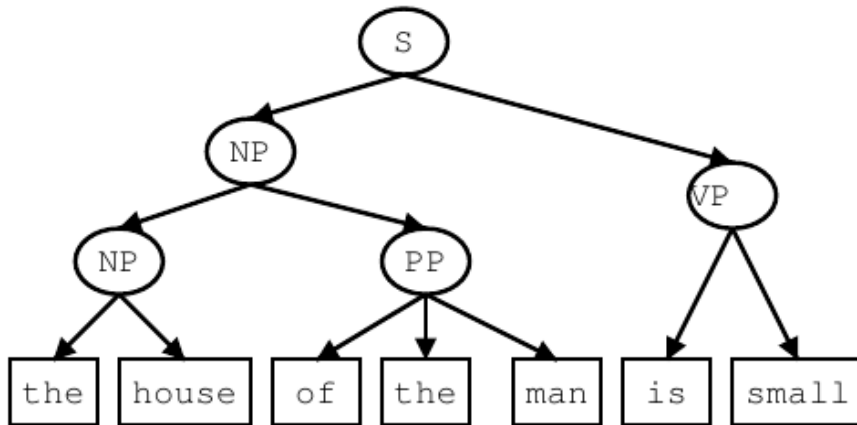
- Reduce vocabulary size – e.g. mapping the numbers to the symbol NUM
- Store on disk and load only what is necessary into memory
- Filter: keep only words that are in the translation options being explored by the decoder
- Quantisation: use fewer bits to store probabilities

Content

1. What is a Language Model?
2. N-gram Language Model
3. Smoothing
4. Evaluation
5. Management of Large Language Models
6. **Other Approaches**

Syntax Language Models

- Take into account syntactic order and tags
 - $p(\text{is}|\text{the,house})$
 - $p(\text{is}|\text{VP,house})$



(P. Koehn)

Continuous Space LMs

- Also referred to as Neural Network LMs
- Words represented as multidimensional vectors (e.g. 300d) in a continuous space
- Neural Network trained to predict a new word (vector) given a sequence of words (vectors)

Will be detailed in Week 9 (Neural Language Modelling)



Discussion

Acknowledgement

Parts of the content of this lecture are taken from previous lectures and presentations given by Antonio Toral, Qun Liu, Jennifer Foster, Declan Groves, Yvette Graham, Kevin Knight, Josef van Genabith and Andy Way