# Words: Language Modelling By N-Grams

COMP-550

Sept 14, 2017

## Outline

Review of last week's class

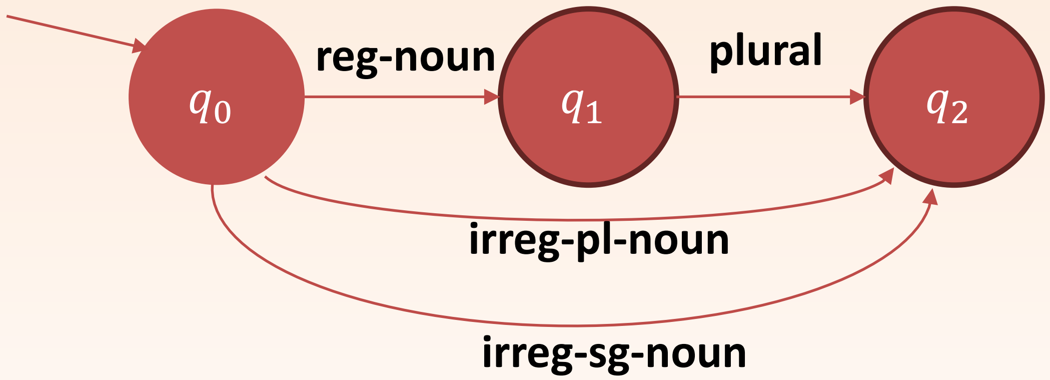
How words are distributed: Zipf's law

Language modelling

- Word sequences: N-grams
- MLE by relative frequencies
- Evaluation by cross entropy and perplexity

# Finite State Machines

FSAs and FSTs for modelling English morphology



# Dealing with Regular Variations

What should be the output of the first FST that maps from the surface form to the intermediate level?

| | |
|---|---|
| *jump* | *jump#* |
| *jumps* | *jump^s#* |
| *jumped* | *jump^ed#* |
| *jumping* | *?* |
| *chat* | *?* |
| *chats* | *?* |
| *chatted* | *?* |
| *chatting* | *?* |
| *hope* | *?* |
| *hopes* | *?* |
| *hoped* | *?* |
| *hoping* | *?* |

## What is a Word?

- Smallest unit that can appear in isolation

Actually not so clear cut:

| | |
|---|---|
| *Football* | One word, or two? |
| *Peanut butter* | One word, or two? |

Languages that don’t separate words with spaces in writing (e.g., Chinese) – even less clear cut

- e.g.,分手信

| | |
|---|---|
| [分][手][信] | 3 words: Distribute + hand + letter ??? |
| [分][手信] | 2 words: Distribute + souvenirs |
| [分手][信] | 2 words: Breakup + letter |
| [分手信] | 1 word: Breakup letter |

- Word segmentation a major problem in Chinese NLP

## Orthographic Word, Types vs. Tokens

Convenient assumption: spaces delimit words

- Exceptions: apostrophe (e.g., *'s*), punctuation

Still ambiguous to ask, "How many words are there?"

*e.g., the cat sat on the mat*

Word **tokens**

6: *cat, mat, on, sat, the, the*

- Instances of occurrences

Word **types**

5: *cat, mat, on, sat, the*

- Kinds of words

## Fuzzy Cases

Do these count as the same word type?

*run, runs*

*happy, happily*

*frágment* (n.), *fragmént* (v.)

*realize, realise*

*We, we*

*srsly, seriously*

**Review**: Which of the above cases would be normalized by **stemming**? By **lemmatization**?

## Word Frequencies

First thing we can do with words? Count them!

**Term frequency**:

$$TF(w, S) = \#w \text{ in corpus } S$$

- e.g., $TF(cat, \text{the cat sat on the mat}) = 1$

**Relative frequency**:

$$RF(w, S) = \frac{TF(w, S)}{|S|}$$

- e.g., $RF(cat, \text{the cat sat on the mat}) = \frac{1}{6}$

## Corpus (n. sing.)

We need a **corpus** (pl.: **corpora**) of text to count.

Some well-known English text corpora:

- Brown corpus
- British National Corpus (BNC)
- Wall Street Journal corpus
- English Gigaword

## Zipf's Law

When counting word frequencies in corpora, this is one striking effect that you'll notice:

$$f \propto \frac{1}{r}$$

Frequency of word type

Rank of word type (by frequency)

## Some Empirical Counts

| Rank | Word | Frequency |
|---|---|---|
| 1 | *the* | 228,257,001 |
| 2 | *to* | 96,247,620 |
| 3 | *of* | 93,917,643 |
| 10 | *for* | 34,180,099 |
| 100 | *most* | 3,499,587 |
| 1,000 | *work* | 1,999,899 |
| 10,000 | *planning* | 299,996 |

Word counts from the English Gigaword corpus

Zipf's Law is (very) roughly true

## Zipf-Mandelbrot Law

To get a better fit to the word counts we see, we can add parameters to the equation:

$f \propto \frac{1}{r}$ means $f = \frac{P}{r}$ for some $P$

Add additional parameters $\rho, B$:

$$f = \frac{P}{(r+\rho)^B}$$

Or equivalently:

$$\log f = \log P - B \log(r + \rho)$$

## “The Long Tail”

Practical implications:

- Most word (types) are very rare!
- A small number of word (types) make up the majority of word (tokens) that you see in any corpus.

- These issues will cause problems for us in terms of designing models and evaluating their performance, as we will see.

## Cross-linguistically Speaking

The parameters in the Zipf-Mandelbrot equation will differ by language

| | |
|---|---|
| English: | top handful of word types will account for most tokens. ~40% of words appear once in a corpus. |
| Hungarian: | same number of word types account for fewer tokens |
| Inuktitut: | ~80% of words appear only once (Langlais and Patry, 2006) |

Disparity caused by the difference in morphological richness between languages

## Why Count Words?

Word frequencies turn out to be very useful:

- Text classification (for genre, sentiment, authorship, …)
- Information retrieval
- Many, many, other applications

Task we will be considering: **language modelling**

## Language Modelling

Predict the next word given some context

*Mary had a little _____*

| | |
|---|---|
| • *lamb* | GOOD |
| • *accident* | GOOD? |
| • *very* | BAD |
| • *up* | BAD |

## Viewed Probabilistically

Learn a probability distribution

- $P(W = w \mid C)$

Random variable $W$ takes on a value which is a word in the lexicon

$w$ represents that value

$C$ is the context that we are conditioning on

e.g.,

$$P(W = \text{"lamb"} \mid C = \text{“Mary had a little”}) = 0.6$$

People are often lazy:

$$P(\text{"lamb"} \mid \text{“Mary had a little”})$$

If any of this notation is not obvious, go review basics of probability theory now! (As in, right after class.)

## Equivalently

Learn probability distribution over sequences of words

Let the context be all of the previous words. Then,

$P(w_1 w_2 \dots w_k)$

$= P(w_k | w_1 \dots w_{k-1}) P(w_1 \dots w_{k-1})$ By the **chain rule**

$= P(w_k | w_1 \dots w_{k-1}) P(w_{k-1} | w_1 \dots w_{k-2}) P(w_1 \dots w_{k-2})$

Keep decomposing further…

$= P(w_k | w_1 \dots w_{k-1}) \dots P(w_2 | w_1) P(w_1)$

## Example

A good language model should assign:

- higher probability to a grammatical string of English

  *You are wearing a fancy hat.*

- lower probability to ungrammatical strings

  **Fancy you are hat a wearing.*

  **Your waring a fency haat.*

## Note

The absolute probability from a language model isn't a good indicator of grammaticality.

- e.g., P(*artichokes intimidate zippers*)
- Likely low probability, but grammatical

Also, the length of the sentence and the rarity of the words in the sentences affect the probability

- e.g., P(*I ate the*) > P(*I ate the cake*) in most language models, but the former is clearly not a well formed sentence!

## What Do Language Models Capture?

- Some linguistic knowledge
- Even facts about the world

eg., Consider just the previous word as context:

P(English|want) = 0.0011

P(Chinese|want) = 0.0065

World knowledge: culinary preferences?

P(to|want) = 0.66

P(eat|to) = 0.28

P(food|to) = 0

Syntax

P(I|<start-of-sentence>) = 0.25

Discourse: people like to talk about themselves?

## Applications

- Text prediction for mobile devices
- Automatic speech recognition (ASR)
- Machine translation

Typically, find the solution that maximizes a combination of:

1. Task-specific quality

   ASR: acoustic model quality

   MT: word/phrase alignment probability
2. Language model probability

## Building Models

Given lots of data from the real world, we can build a **model**, which is a set of **parameters** that describes the data, and can be used to **predict** or **infer** future or unseen data.

e.g.,

*Task*: language modelling

*Model*: a probability distribution, $P(W = w \mid C)$

*Parameters*: the parameters to this probability distribution

*Application*: tell us how likely it is to observe $w_N$ given its context

## Steps

1. Gather a large, representative training corpus
2. Learn the parameters from the corpus to build the model
3. Once the model is fixed, use the model to evaluate on testing data

## Steps

1. Gather a large, representative training corpus
2. **Learn the parameters from the corpus to build the model**
3. Once the model is fixed, use the model to evaluate on testing data

## Learning the Model

How do we actually learn the parameters to $P(W = w | C)$ given training data?

Need to:

- Specify exactly what the context of a word is
- Use corpus counts to derive the parameter values

## N-grams

Make a **conditional independence assumption** to make the job of learning the probability distribution easier.

- Context = the previous N-1 words

Common choices: N is between 1 and 3

**Unigram** model

$P(w_N|C) = P(w_N)$

**Bigram** model

$P(w_N|C) = P(w_N|w_{N-1})$

**Trigram** model

$P(w_N|C) = P(w_N|w_{N-1}, w_{N-2})$

## Deriving Parameters from Counts

Simplest method: count N-gram frequencies, then divide by the total count

e.g.,

**Unigram**: P(*cats*) = Count(*cats*) / Count(all words in corpus)

**Bigram**: P(*cats* | *the*) = Count(*the cats*) / Count(*the*)

**Trigram**: P(*cats* | *feed the*) = ?

These are the **maximum likelihood estimates** (**MLE**).

## Exercise

Come up with the MLE estimate of a unigram and a bigram language model using the following sentence as training data:

*that that is is that that is not is not is that it it is*

## Steps

1. Gather a large, representative training corpus
2. Learn the parameters from the corpus to build the model
3. **Once the model is fixed, use the model to evaluate on testing data**

## Training and Testing Data

After training a model, we need to evaluate it on unseen data that the model has not been exposed to.

- We are testing the model's ability to generalize.
- More on this topic next class

Given a corpus, how is the data usually split?

**Training data**: often 60-90% of the available data

**Testing data**: often 10-20% of the available data

There is often also a **development** or **validation** data set, for deciding between different versions of a model.

# Cross Validation

**k-fold cross validation**: splitting data into k partitions or folds; iteratively test on each after training on the rest

e.g., 3-fold CV: split dataset into 3 folds

| | **Fold 1** | **Fold 2** | **Fold 3** |
|---|---|---|---|
| Exp. 1 | test | train | train |
| Exp. 2 | train | test | train |
| Exp. 3 | train | train | test |

Average results from above experiments

- CV is often used if the corpus is small

## Evaluation Measures

Likelihood of generating the test corpus

- i.e., P(test_corpus; $\theta$), where $\theta$ represents the parameters learned by training our LM on the training data
- **Intuition**: a good language model should give a high probability of generating some new, valid English text.
- Absolute number is not very meaningful—this can only be used to compare the quality of different language models!

Unwieldy because of small values, so not actually used in the literature. Alternatives to likelihood:

- Cross-entropy
- Perplexity

## Basic Information Theory

Consider some random variable X, distributed according to some probability distribution.

We can define information in terms of how much certainty we gain from knowing the value of X.

Rank the following in terms of how much information we gain by knowing its value:

- Fair coin flip
- An unfair coin flip where we get tails ¾ of the time
- A very unfair coin that always comes up heads

## Likely vs Unlikely Outcomes

Observing a likely outcome – less information gained

**Intuition**: you kinda knew it would happen anyway

- e.g., observing the word *the*

Observing a rare outcome: more information gained!

**Intuition**: it's a bit surprising to see something unusual!

- e.g., observing the word *armadillo*

Formal definition of information in bits:

$$I(x) = \log_2\left(\frac{1}{P(x)}\right)$$

Minimum number of bits needed to communicate some outcome *x*

## Entropy

The expected amount of information we get from observing a random variable.

Let a discrete random variable be drawn from distribution *p* take on one of k possible values with probabilities $p_1$ … $p_k$

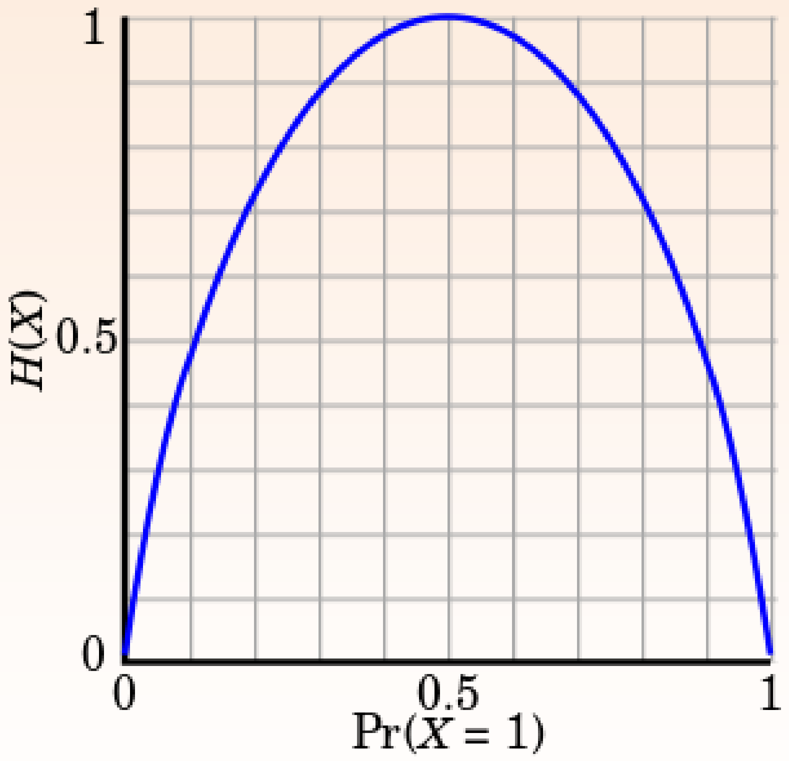
$$H(p) = \sum_{i=1}^{k} p_i I(x_i)$$

$$= \sum_{i=1}^{k} p_i \log_2 \frac{1}{p_i}$$

$$= -\sum_{i=1}^{k} p_i \log_2 p_i$$

# Entropy Example

Plot of entropy vs. coin toss "fairness"



Maximum fairness = maximum entropy

Completely biased = minimum entropy

Image source: Wikipedia, by Brona and Alessio Damato

## Cross Entropy

Entropy is the minimum number of bits needed to communicate some message, *if we know what probability distribution the message is drawn from*.

**Cross entropy** is for when we don't know.

e.g., language is drawn from some true distribution, the language model we train is an approximation of it

$$H(p,q) = -\sum_{i=1}^{k} p_i \log_2 q_i$$

$p$: "true" distribution

$q$: model distribution

## Estimating Cross Entropy

When evaluating our LM, we assume the test data is a good representative of language drawn from *p*.

So, we estimate cross entropy to be:

$$H(p, q) = -\frac{1}{N} \log_2 q(w_1 \dots w_N)$$

True language distribution, which we don't have access to.

Language model under evaluation

Size of test corpus in number of tokens

The words in the test corpus

## Perplexity

Cross entropy gives us a number in bits, which is sometimes hard to read. Perplexity makes this easier.

$$\text{Perplexity}(p, q) \; = \; 2^{H(p,q)}$$