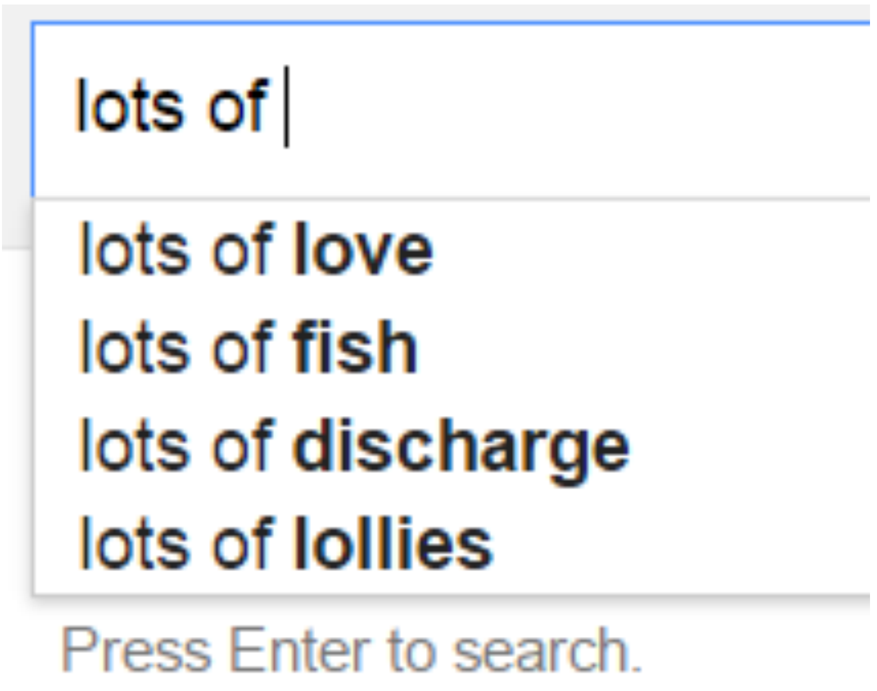
N-gram language models

COMP90042 Lecture 8

Language models

- Assign a probability to a sequence of words
- Useful for
  - Speech recognition
  - Spelling correction
  - Machine translation
  - Query completion
  - Optical character recog.
- Other generation tasks
  - Summarisation
  - Dialogue systems



lots of |

lots of love
lots of fish
lots of discharge
lots of lollies

Press Enter to search.

Outline

- Deriving *n*-gram language models
  - Easy: Markov models
- Smoothing to deal with sparsity
  - Hard: add-1 smoothing does not really work here
- Evaluating language models

Probabilities: Joint to conditional

Our goal is to get a probability for an arbitrary sequence of *m* words

$$P(w_1, w_2, \ldots, w_m)$$

First step is to apply the chain rule to convert joint probabilities to conditional ones

$$P(w_1,w_2,\ldots,w_m) = P(w_1)P(w_2|\, w_1)P(w_3|\, w_1,w_2)\ldots$$
$$P(w_m|w_1 \ \ldots w_{m-1})$$

The Markov Assumption

Still intractable, so make a simplifying assumption:

$$P(w_i|w_1 \;\; \ldots w_{i-1}) \approx P(w_i|w_{i-n+1} \ldots w_{i-1})$$

For some small *n*

When *n* = 1, a unigram model

$$P(w_1, w_2, .. \, w_m) = \prod_{i=1}^{m} P(w_i)$$

When *n* = 2, a bigram model

$$P(w_1, w_2, .. \, w_m) = \prod_{i=1}^{m} P(w_i|w_{i-1})$$

When *n* = 3, a trigram model

$$P(w_1, w_2, .. \, w_m) = \prod_{i=1}^{m} P(w_i|w_{i-2}w_{i-1})$$

Maximum Likelihood estimation

How do we calculate the probabilities? Estimate based on counts in our corpus:

For unigram models,

$$P(w_i) = \frac{C(w_i)}{M}$$

For bigram models,

$$P(w_i | w_{i-1}) = \frac{C(w_{i-1} w_i)}{C(w_{i-1})}$$

For *n*-gram models generally,

$$P(w_i | w_{i-n+1} \dots w_{i-1}) = \frac{C(w_{i-n+1} \dots w_i)}{C(w_{i-n+1} \dots w_{i-1})}$$

Book-ending Sequences

- Special tags used to denote start and end of sequence
  - <s> = sentence start (□ in E18)
  - </s> = sentence end (■ in E18)
- Include (n-1) start tokens for n-gram model, to provide context to generate first word
  - never generate <s>
  - generating </s> terminates the sequence

Trigram example

Corpus:

<s> <s> *yes no no no no yes* </s>
<s> <s> *no no no yes yes yes no* </s>

What is the probability of

<s> <s> *yes no no yes* </s>

under a trigram language model?

P(sent=yes no no yes)
= P(*yes* |*<s> <s>*) * *P(no | <s> yes)* * *P(no | yes no)*
* *P(yes | no no)* * *P(</s> | no yes)*
= *½ * 1 * ½ ** $\frac{2}{5}$ ** 1 = 0.1*

Several problems

- Language has long distance effects — need large *n*
  - *The lecture/s that took place last week was/were on retrieval.*
- Resulting probabilities are often very small
  - Use log probability to avoid numerical underflow
- No probabilities for unseen words
- Words in new contexts
  - By default, zero count for any *n*-gram we've never seen before, zero probability for the sentence
  - Need to smooth the LM!

Smoothing (or discounting)

- Basic idea: give events you've never seen before some probability
- Have to take away probability from events you have seen
- Must be the case that *P*(everything) = 1
- Many different kinds of smoothing
  - Laplacian (add-one) smoothing
  - Add-*k* smoothing
  - Jelinek-Mercer interpolation
  - Katz backoff
  - Absolute discounting
  - Kneser-Ney
  - And others…

Laplacian (Add-one) smoothing

- Simple idea: pretend we've seen each *n*-gram once more than we did.

For unigram models (**V**= the vocabulary),

$$P_{add1}(w_i) = \frac{C(w_i) + 1}{M + |\boldsymbol{V}|}$$

For bigram models,

$$P_{add1}(w_i|w_{i-1}) = \frac{C(w_{i-1}w_i) + 1}{C(w_{i-1}) + |\boldsymbol{V}|}$$

Add-one Example

<s> *the rat ate the cheese* </s>

What's the bigram probability *P(ate|rat*) under add-one smoothing?

$= \frac{C(rat\ ate)+1}{C(rat)+|\boldsymbol{V}|} = \frac{2}{5}$

V = { the, rat, ate, cheese, </s> }

What's the bigram probability *P(ate|cheese*) under add-one smoothing?

$= \frac{C(cheese\ ate)+1}{C(cheese)+|\boldsymbol{V}|} = \frac{1}{5}$

Add-*k* smoothing

- Adding one is often too much
- Instead, add a fraction *k*

$$P_{addk}(w_i|w_{i-1},w_{i-2}) = \frac{C(w_{i-2},w_{i-1},w_i)+k}{C(w_{i-2},w_{i-1})+k|\boldsymbol{V}|}$$

- Have to choose *k*
  - number of “classes” is huge (n-grams), so can have a big effect

Kneser-Ney smoothing

- State-of-the-art method for n-gram language models.
- A fairly complex method, combining three ideas:
  - Interpolation (or alternatively, backoff)
  - Absolute discounting
  - Continuation counts
- Let’s see each of these steps in detail.

Backoff and Interpolation

- Smooth using lower-order probabilities (less context)
- Backoff: fall back to *n*-1-gram counts only when *n*-gram counts are zero

$$P_{BO}(w_i|w_{i-2},w_{i-1})$$
$$= \begin{cases} P^*\,(w_i|w_{i-2},w_{i-1}) & if\ C(w_{i-2},w_{i-1},w_i) > 0 \\ \alpha(w_{i-2},w_{i-1}) * P_{BO}(w_i|w_{i-1}) & otherwise \end{cases}$$

P* and α must preserve “sum to 1” property.

Backoff and Interpolation

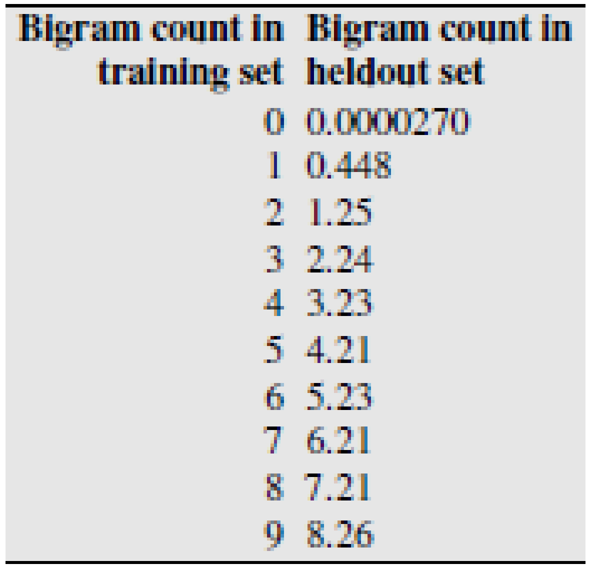
- Interpolation involves taking a linear combination of all relevant probabilities
- Defined recursively:

$$P_{interp}(w_i|w_{i-2},w_{i-1}) =$$
$$\lambda(w_{i-2},w_{i-1})\,P(w_i|w_{i-2},w_{i-1})$$
$$+\,(1\; - \lambda\big(w_{i-2},w_{i-1}\big)\big)P_{interp}(w_i|w_{i-1})$$

- Interpolation of probabilities preserves “sum to 1” property
- λs can be constant across all contexts
  - But better if sensitive to n-grams
- Parameters need to be trained on held out data

Absolute discounting

- What if we estimate the counts from a heldout corpus?
- Turns out a single absolute discounting works for almost all *n*-grams
  - Most mass taken from low counts
  - Doesn’t effect high counts much



Bigram count in training set	Bigram count in heldout set
0	0.0000270
1	0.448
2	1.25
3	2.24
4	3.23
5	4.21
6	5.23
7	6.21
8	7.21
9	8.26

- $P_{Abs}(w_i|w_{i-1}) = \frac{C(w_{i-1},w_i)-d}{C(w_{i-1})} + \lambda(w_{i-1})P(w_i)$

Continuation counts

- When backing-off or interpolating, raw counts can be fairly unreliable
  - $P(Zealand|Old) = ?$ will interpolate with $P(Zealand)$
  - *Zealand* has high counts, but only appears after *New*
    - Don’t want to assign it much probability when *New* not present
- Instead, use frequency **of contexts** in which word appears
  - For many words, closely related to total count
  - But just 1 for *Zealand*

$$continuation\_count(w_1) = |\{v: count(v, w_1) > 0\}|$$

Throwing it all together

$$P_{KN}(w_i|w_{i-2},w_{i-1}) = \frac{\max(0, C_{KN}(w_{i-2},w_{i-1},w_i) - d)}{C_{KN}(w_{i-2},w_{i-1})}$$

$$+ \lambda(w_{i-2},w_{i-1})P_{KN}(w_i|w_{i-1})$$

$$\lambda(w_{i-2},w_{i-1}) = \frac{d}{C_{KN}(w_{i-2},w_{i-1})} |\{w: C_{KN}(w_{i-2},w_{i-1},w) > 0\}|$$

C_{KN} is a continuation count, **except for the highest n-gram order:** we use a regular count instead.

In practice

- Best Kneser-Ney version uses different discount values for each n-gram order.
- Most used LMs use 5-grams as the max order but higher order sometimes can be used if large amounts of data are available.

Evaluation

- Extrinsic
  - E.g. spelling correction, machine translation
- Intrinsic
  - Perplexity on held-out test set

Perplexity

- Inverse probability of entire test set
  - Normalized by number of words (including </s>)
- The lower the better

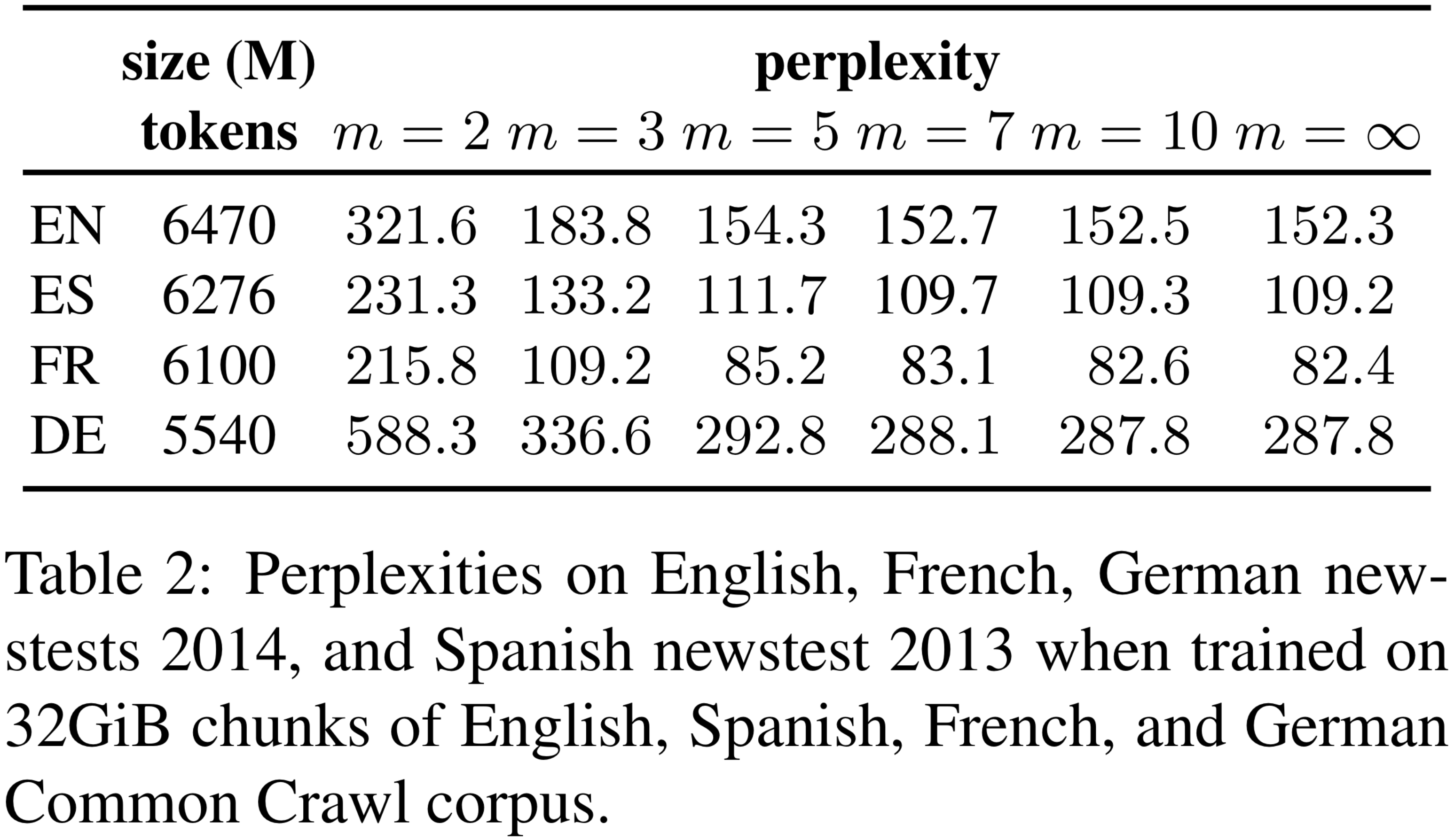
$$PP(w_1, w_2, .. w_m) = \sqrt[m]{\frac{1}{P(w_1, w_2, .. w_m)}}$$

equivalently

$$PP(w_1, w_2, .. w_m) = 2^{-\frac{\log_2 P(w_1, w_2, .. w_m)}{m}}$$

- Unknown (OOV) words a problem (omit)

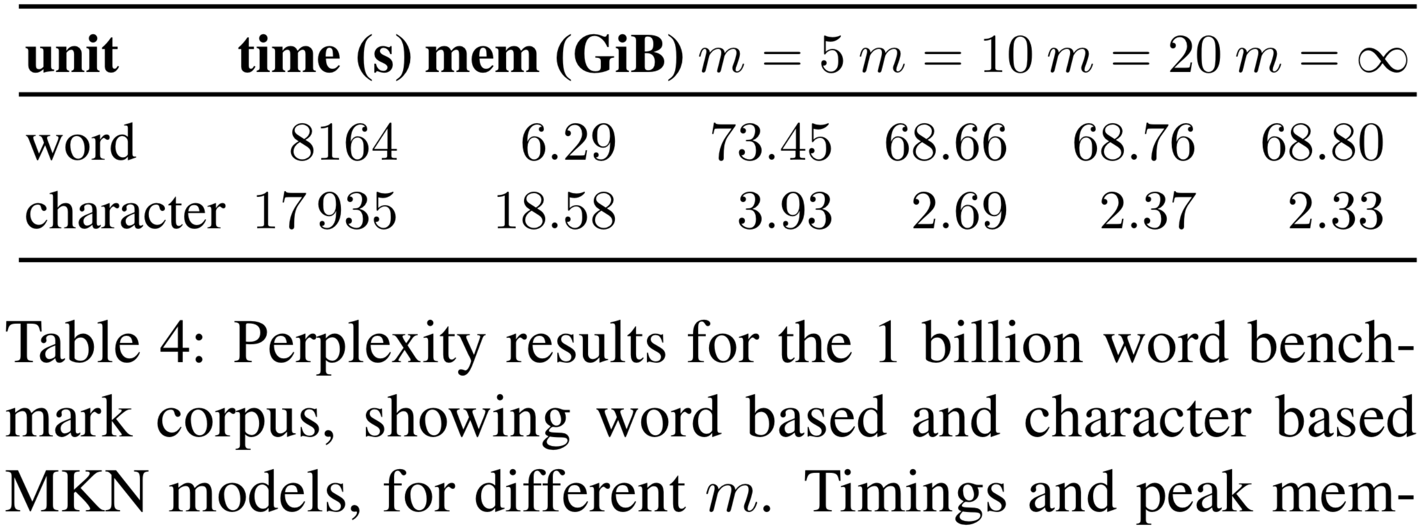
Example perplexity scores



	size (M)	perplexity					
	tokens	$m = 2$	$m = 3$	$m = 5$	$m = 7$	$m = 10$	$m = \infty$
EN	6470	321.6	183.8	154.3	152.7	152.5	152.3
ES	6276	231.3	133.2	111.7	109.7	109.3	109.2
FR	6100	215.8	109.2	85.2	83.1	82.6	82.4
DE	5540	588.3	336.6	292.8	288.1	287.8	287.8

Table 2: Perplexities on English, French, German newstests 2014, and Spanish newstest 2013 when trained on 32GiB chunks of English, Spanish, French, and German Common Crawl corpus.

Shareghi, Petri, Haffari and Cohn. *Transactions of ACL,* 2016.



unit	time (s)	mem (GiB)	$m = 5$	$m = 10$	$m = 20$	$m = \infty$
word	8164	6.29	73.45	68.66	68.76	68.80
character	17 935	18.58	3.93	2.69	2.37	2.33

Table 4: Perplexity results for the 1 billion word benchmark corpus, showing word based and character based MKN models, for different m. Timings and peak mem-

Generated texts

- Language models can also be used to generate texts
- Given a initial word, **sample** the next word according to the probability distribution defined by the language model.

This shall forbid it should be branded, if renown made it empty

They also point to ninety nine point six billion dollars from two hundred four oh three percent of the rates of interest stores as Mexico and Brazil on market conditions

A final word

- *N*-gram language models are a structure-neutral way to capture the predictability of language
- Information can be derived in an unsupervised fashion, scalable to large corpora
- Require smoothing to be effective, due to sparsity
- N-gram models do not (explicitly, at least) encode word similarity: remember the lexical and distributional semantics lectures?

Reading

- E18 Chapter 6 (skip 6.3)