CMPT 413
Computational Linguistics

Anoop Sarkar

http://www.cs.sfu.ca/~anoop

n-grams

- A simple model of language
- Computes a probability for observed input
- Probability is likelihood of observation being generated by the same source as the training data
- Such a model is often called a *language model*

An example

- Let’s consider an example we’ve seen before: *spelling correction*

…*was called a “stellar and versatile* ***acress*** *whose combination of sass and glamour has defined her* …

KCG model best guess is **acres**

An example

- A language model can take the context into account:
  - …*was called a "stellar and versatile* ***acress*** *whose combination of sass and glamour has defined her* …
  - …*was called a "stellar and versatile* ***acres*** *whose combination of sass and glamour has defined her* …
  - …*was called a "stellar and versatile* ***actress*** *whose combination of sass and glamour has defined her* …
- Each sentence is a sequence $w_1, \ldots, w_n$. Task is to find $P(w_1, \ldots, w_n)$.

Another example

physical Brainpower not plant is chief , now a 's asset , . firm
, a Brainpower not now chief asset firm 's is . plant physical ,
chief a physical , . firm not , Brainpower plant is asset 's now
not plant Brainpower now physical 's . a chief , asset firm , is
plant Brainpower is now , , not . firm a 's physical asset chief
physical is 's plant firm not chief . Brainpower now asset , , a
Brainpower , not physical plant , is now a firm 's chief asset .

Each sentence is a sequence $w_1, \ldots, w_n$.
Task is to find $P(w_1, \ldots, w_n)$.

How can we compute $P(w_1, \ldots, w_n)$

- Apply the *Chain rule*
- $P(w_1, \ldots, w_n) = P(w_1)\ .\ P(w_2 / w_1)\ .\ P(w_3 / w_1, w_2) \ldots P(w_n / w_1, \ldots, w_{n-1})$
- Each of these probabilities can be estimated (using frequency counts) from *training data*
- **But** we need to apply these probabilities on unseen *test data*
- The curse of dimensionality: **sparse data**

The Markov Assumption

a stellar and versatile ***acres*** *whose combination of*

P(a) . P(stellar | a) . P(and | a, stellar) .

P(versatile | a, stellar, and) .

*P(**acres** | a, stellar, and, versatile) .*

*P(whose | a, stellar, and, versatile, **acres**) …*

a stellar and versatile ***acres*** *whose combination of*

P(a) . P(stellar | a) . P(and | a, stellar) . P(versatile | stellar, and) .

*P(**acres** | and, versatile) . P(whose | versatile, **acres**) …*

n-grams

- *0*th order Markov model: P(w_i) called a ***unigram*** model
- *1*st order Markov model: P(w_i | w_{i-1}) called a ***bigram*** model
- 2nd order Markov model: P(w_i | w_{i-2}, w_{i-1}) called a ***trigram*** model

Parameter size

Corpus: <s> said the joker to the thief

N (tokens) = 7 |V| = 6

$$p(joker|the) = \frac{p(the, joker)}{p(the)}$$

$$= \frac{\frac{f(the, joker)}{\text{num of bigrams}}}{\frac{f(the)}{\text{num of unigrams}}} = \frac{f(the, joker)}{f(the)}$$

n-grams

- How many possible distinct probabilities will be needed?, i.e. **parameter values**
- Total number of **word tokens** in our training data
- Total number of unique words: **word types** is our vocabulary size

n-gram Parameter Sizes

- Let V be the vocabulary, size of V is $|V|$
- $P(W_i = x)$, how many different values for W_i
  - $|V| = 3x10^3$
- $P(W_i = x \mid W_j = y)$, how many different values for W_i, W_j
  - $|V|^2 = 9x10^6$
- $P(W_i = x \mid W_k = z, W_j = y)$, how many different values for W_i, W_j, W_k
  - $|V|^3 = 27x10^9$

Parameter size

Corpus: <s> said the joker to the thief

$|V| = 6$

Bigrams: max num of parameters = $|V|^2 = 36$

said | <s>
the | said
joker | the
to | joker
the | to
thief | the

observed = $W_T = 6 \quad << 36$

n-gram model of Jane Austen

- Three novels by Jane Austen: *Emma*, *Sense and Sensibility*, *Pride and Prejudice*
- Removed punctuation and kept paragraph structure
- Trained a trigram model on this text

n-gram model of Jane Austen

f(3gram)	f(2gram)	f(1gram)	w_0	w_1	w_2
378	518	10381	I	do	not
366	1366	10381	I	am	sure
214	1917	9182	in	the	world
202	572	6917	she	could	not
189	462	2751	would	have	been
174	184	10381	I	dare	say
173	179	5758	as	soon	as
173	357	11135	a	great	deal
171	332	7573	it	would	be
155	945	3017	could	not	be

n-gram model of Jane Austen

3gram $\frac{f(w_0,w_1,w_3)}{f(w_0,w_1)}$	2gram $\frac{f(w_0,w_1)}{f(w_0)}$	1gram $\frac{f(w_0)}{N}$	w_0	w_1	w_2
0.72	0.04	0.016	I	do	not
0.26	0.13	0.016	I	am	sure
0.11	0.20	0.014	in	the	world
0.35	0.08	0.011	she	could	not
0.40	0.16	0.004	would	have	been
0.94	0.01	0.016	I	dare	say
0.96	0.03	0.009	as	soon	as
0.48	0.03	0.018	a	great	deal
0.51	0.04	0.012	it	would	be
0.16	0.31	0.004	could	not	be

n-gram model of Jane Austen

f(3gram)	f(2gram)	f(1gram)	w_0	w_1	w_2
1	1	1	favor	of	your
1	1	1	peerage	his	wealth
1	1	1	stagnation	Mrs	Elton's
1	1	1	genteelly	and	paid
1	1	1	adept	in	the
1	1	1	deckers	now	in
1	1	1	oracle	Fanny's	explanations
1	1	1	Ashworth	is	too
1	1	1	puddles	with	impatient
1	1	1	Harringtons	to	come
1	1	1	roasted	No	coffee
1	1	1	coherent	Dearest	Lizzy

n-gram model of Jane Austen

3gram	2gram	1gram	w_0	w_1	w_2
0.00039	0.13	0.029	of	the	Middletons
0.00039	0.13	0.029	of	the	Meryton
0.00039	0.13	0.029	of	the	Lucases
0.00039	0.13	0.029	of	the	London
0.00039	0.13	0.029	of	the	Irish
0.00039	0.13	0.029	of	the	History
0.00039	0.13	0.029	of	the	First
0.00039	0.13	0.029	of	the	Esquire
0.00039	0.13	0.029	of	the	Elegant
0.00039	0.13	0.029	of	the	Dashwoods
0.00039	0.13	0.029	of	the	Crown

Applications of n-gram models

- N-gram models are used to provide a probabilty $P(w_1, \ldots, w_n)$
- Note that this is simplest such model of the probability of some NL input
- The probability can be exploited in many applications that require some ranking of plausible output in a NL
- Applications include speech recognition, machine translation, generation, language identification, spelling correction, re-capitalization of text, …

Speech Recognition

- **Speech recognition**: find word sequence **w** that maximizes P(**w** | **o**), where **o** is a sequence of time dependent acoustic features (output of signal processing on speech signal)
- P(**w** | **o**) = P(**o** | **w**) * P(**w**) using Bayes Rule
- Decomposition of P(**o** | **w**) into a cascade of models (each one can be implemented as a Hidden Markov Model):

  Acoustic Model (e.g. model trained on the TIMIT corpus):

  P(**o** | **p**) predict observation sequence **o** given phone sequence **p**

  Pronunciation Model (e.g. using TIMIT and the CMU pronunciation dictionary):

  P(**p** | **w**) predict phone sequence **p** given a word sequence **w**
- **Language Model**: P(**w**) predict word sequence **w**

cf. *Fundamentals of Speech Recognition*, Rabiner and Juang

Statistical machine translation

- Statistical machine translation: find word sequence **e** that maximizes P(**e** | **f**), where **e** is a sentence in the target language and **f** is the input sentences in the source language (IBM models: Brown et al, 1993)
- P(**e** | **f**) = P(**f** | **e**) * P(**e**) using Bayes Rule
- Decomposition of P(**f** | **e**) into a cascade of generative models:
- **Translation Model**:

  P(**f** | **e**) predict observed source **f** given candidate target **e**

  $P(\mathbf{f} \mid \mathbf{e}) = \Sigma_{\mathbf{a}} P(\mathbf{a}, \mathbf{f} \mid \mathbf{e})$

  The model sums over all possible alignments **a** between **f** and **e**

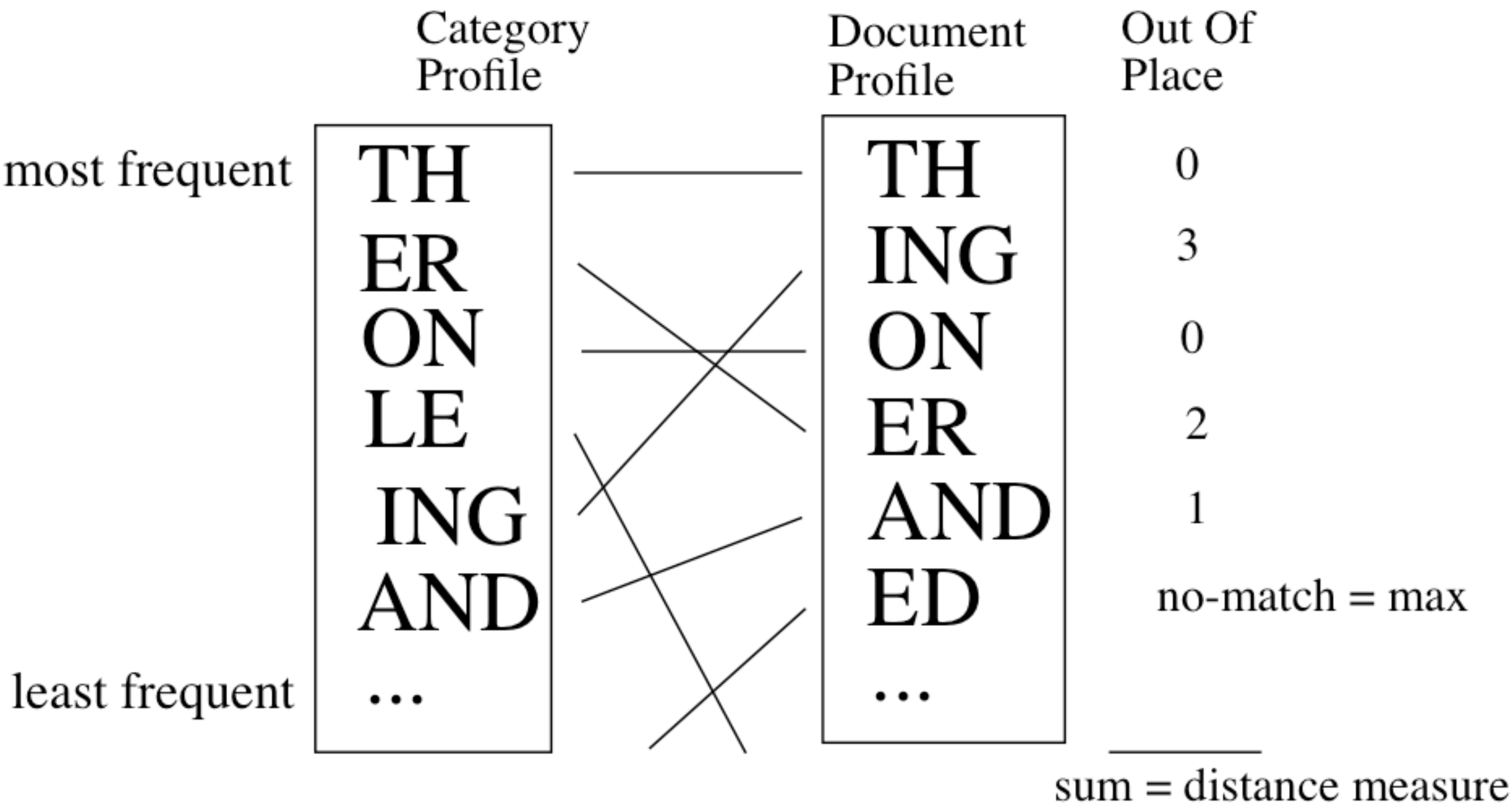
  Think of each alignment as being an edit-distance alignment between **f** and **e**
- **Language Model**:

  P(**e**) predict word sequence in target language **e**

Language Identification using n-grams

- N-gram based algorithm (Cavnar and Trenkle, 1994) for language identification
- Collect a set of documents in different languages, e.g. a list of 77 languages: http://www.let.rug.nl/~vannoord/TextCat/list.html
- Compute n-grams for each language (n=5) and keep the K most frequent n-grams (K=400): call this the **language profile**
- For an input document, compute n-grams and then find the out-of-place score for the n-gram ranks between the input document and all the language profiles
- Sum all the out-of-place scores for the input and pick the language with the minimum score

Language Identification using n-grams



Language Id on Twitter

Language	Number of Tweets
EN	59660690
PT-BR	7986562
ES	6244053
ID	3475389
NL	3150534
DE	2216601
MS	1624710
JP	7134916
IT	240035
PT-PT	169643
Total dataset	111852004

Summary

- *n*-grams define a probability model over sequences
  - we have seen examples of sequences of words, but you can define *n*-grams over sequences of characters or other sequences
- *n*-grams deal with sparse data by using the Markov assumption
- The number of parameters increase rapidly when the value of *n* is increased for *n*-grams but the data cannot keep up with the parameter size.
- Google English n-gram corpus: 1TB word tokens and 13M word types
- Open source software for n-grams: SRI language modeling toolkit (SRILM); CMU LM toolkit