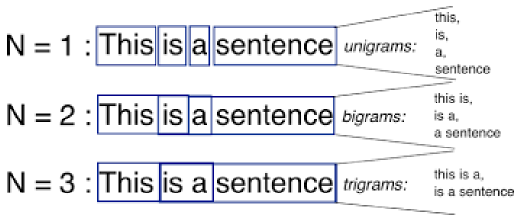
Processamento de Linguagem Natural para Mineração de Textos



Applications of the N-Gram Model

Prof. Rinaldo Lima
rinaldo.ufrpe@gmail.com
UFRPE Universidade Federal Rural de Pernambuco
DEINFO Departamento de Estatística e Informática

14-abr-18

Applications of N-Gram

- Being able to predict the next word (or any linguistic unit) in a sequence is very useful
- It lies at the core of the following applications
  - Automatic speech recognition
  - Handwriting and character recognition
  - Machine translation
  - Augmentative communication
  - Word similarity, generation, POS tagging, etc.
  - Author Identification (stylometry)
  - Spam Detection
  - **Word Prediction**
  - **Spelling correction**
  - **Language Identification**
  - **Criptoanalysis (code breaking)**

2

Contents

Applications of the N-Gram model

- Word Prediction
- Spell Checking
  - Bayes Theorem
  - Edit Distance
- Language Identification
- Keyword Extraction
- Code Breaking

Word Prediction

UFRPE
Universidade Federal Rural de Pernambuco

DEINFO
Departamento de Estatística e Informática

Word Prediction. Why?

- Predictors support writing and are commonly used in combination with assistive devices such as keyboards, virtual keyboards, touchpads and pointing devices.
- Frequently, applications include repetitive tasks such as writing emails in call centers or letters in an administrative environment.
- Applications of word prediction:
  - Spelling Checkers
  - Mobile Phone/PDA Texting
  - Disabled Users
  - Handwriting Recognition
  - Word-sense Disambiguation

Word Prediction - Overview

- *Word Prediction* is the problem of guessing which word is likely to continue a given initial text fragment.
- Word prediction techniques are well-established methods in the field of AAC (Augmentative and Alternative Communication) that are frequently used as communication aids for people with disabilities
  - accelerate the writing;
  - reduce the effort needed to type;
  - suggest the correct word (no misspellings).

Word Prediction - Objectifs

- Ease word insertion in textual software
  - by guessing the next word
  - by giving a list of possible options for the next word
  - by completing a word given a prefix

- General idea:

  guess the next word given the previous ones

  [Input w_1 w_2] → [guess w_3]

7

Example

I s_____ → verb, adverb?

I s_____ → verb

sang? maybe.

singularized? hopefully

I saw a _____

I saw a _____ → noun / adjective

I saw a b____

I saw a b____ → brown? big? bear? barometer?

8

Word Prediction with N-gram Model

- In order to predict the next word (w_N) given the context or history ($w_1, \ldots, w_{N-1}$), we want to estimate this probability function:

$$\mathbb{P}(w_N | w_1, \ldots, w_{N-1})$$

- The language model estimates the values $\mathbb{P}(W)$, where $W = w_1, \ldots, w_N$.

- Markov Assumption: only the prior local content (the last few words) affects the next word.

$(n-1)^{th}$ **Markov Model** or n-**gram**

9

Word Prediction with N-gram Model

Formally, n-gram model is denoted by:

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

- Typical values of n-gram are
  - $n = 1$ (unigram)
    $\mathbb{P}(w_i \mid w_1, \ldots, w_{i-1}) \approx \mathbb{P}(w_i)$
  - $n = 2$ (bigram)
    $\mathbb{P}(w_i \mid w_1, \ldots, w_{i-1}) \approx \mathbb{P}(w_i \mid w_{i-1})$
  - $n = 3$ (trigram)
    $\mathbb{P}(w_i \mid w_1, \ldots, w_{i-1}) \approx \mathbb{P}(w_i \mid w_{i-2}\ w_{i-1})$

10

Word Prediction with N-gram Model (I)

Example:

- W = *Last night I went to the concert*
- Instead of $\mathbb{P}(\textit{concert} \mid \textit{Last night I went to the})$
- we use a bigram $\mathbb{P}(\textit{concert} \mid \textit{the})$
- or a trigram $\mathbb{P}(\textit{concert} \mid \textit{to the})$

Problem with n-grams

- The drawback of these methods is the amount of text needed to train the model. Training corpus has to be large enough to ensure that each valid word sequence appears a relevant number of times.
- A great amount of computational resources is needed especially if the number of words in the lexicon is big.

For a vocabulary $\mathcal{V}$ of 20,000 words

- $|\mathcal{V}|^2 = 400$ million of bigrams;
- $|\mathcal{V}|^3 = 8$ trillion of trigrams;
- $|\mathcal{V}|^4 = 1.6 \times 10^{17}$ of four-grams.

- Since the number of possible words is very large, there is a need to focus attention on a smaller subset of these.

POS N-Gram Model (2)

- One proposed solution consists in generalizing the n-gram model, by grouping the words in *category* according to the context.
- A mapping φ is defined to approximate a context by means of the equivalence class it belongs to: $\mathbb{P}(w_i|\varphi[w_{i-n+1}, \ldots, w_{i-1}])$.
- Usually, Part-of-Speech (POS) tags are used as mapping function, replacing each word with the corresponding POS tag (i.e. classification).
- POS tags have the potential of allowing generalization over similar words, as well as reducing the size of the language model.

Hybrid Approach to Word Prediction (3)

- Prediction can either be based on text statistics or linguistic rules.
- Two Markov models can be included: one for word classes (POS tag unigrams, bigrams and trigrams) and one for words (word unigrams and bigrams). A linear combination algorithm may combine these two models.
- Incorporating morpho-syntactic information to enforce prediction accuracy.

Syntactic Methods

- Syntactic knowledge
  - Consider sequences of part of speech tags
    [Article] [Noun] → predict [Verb]
  - Phrase structure
    [Noun Phrase] → predict [Verb]
  - Syntactic knowledge can be statistical or based on hand-coded rules

Semantic Methods

- Semantic knowledge
  - Assign semantic categories to words
  - Find a set of rules which constrain the possible candidates for the next word
    - [*eat* verb] → *predict* [word of category *food*]
  - Not widely used in word prediction, mostly because it requires complex hand coding and is too inefficient for real-time operation

Word Prediction Knowledge Sources

- Corpora: texts and frequencies
- Vocabularies (Can be domain specific)
- Lexicons with syntactic and/or semantic knowledge
- User's history
- Morphological analyzers
- Unknown words models

17

Spell Checking

(but, before that, let´s see the Bayes Theorem)

(other slides)

UFRPE
Universidade Federal Rural de Pernambuco

DEINFO
Departamento de Estatística e Informática

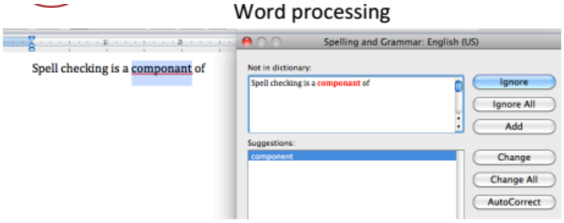
14-abr-18

Edit Distance
(other slides)

Spell Checking

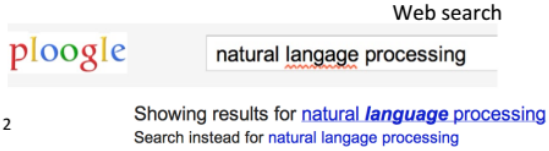
Spelling Checkers

Word processing



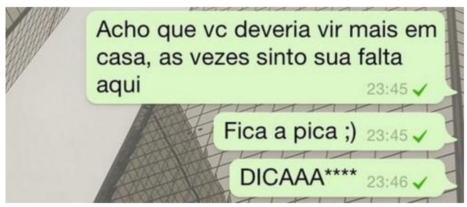
Phones

Web search



2

Spell Checkers and Word Prediction in real world ...



Spelling Tasks

- Spelling Error Detection
- Spelling Error Correction:
  - Autocorrect
    - hte→the
  - Suggest a correction
  - Suggestion lists

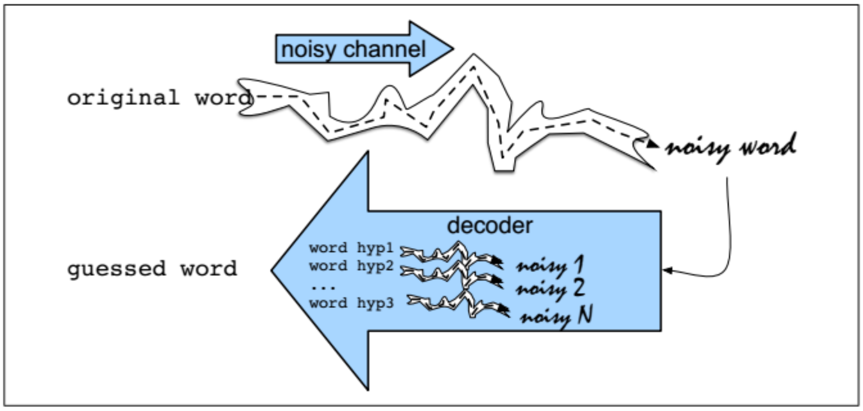
Types of Errors

- Non-word Errors
  - *graffe →giraffe*

Our Focus

- Real-word Errors
  - Typographical errors
    - *three →there*
  - Cognitive Errors (homophones)
    - *piece→peace,*
    - *too → two*

Noisy Channel Intuition



The intuition of the noisy channel is to treat the misspelled word as if a correctly spelled word had been "distorted" by being passed through a noisy communication channel.

The noisy channel model is a kind of **Bayesian Inference.**

26

Noisy Channel

- We see an observation x of a misspelled word
- Find the correct word w

$$\hat{w} = \underset{w \in V}{\operatorname{argmax}}\, P(w \mid x)$$

Out of all words in V, which maximizes P(w|x)

$$= \underset{w \in V}{\operatorname{argmax}} \frac{P(x \mid w)P(w)}{P(x)}$$

Bayes Theorem

$$= \underset{w \in V}{\operatorname{argmax}}\, P(x \mid w)P(w)$$

P(w) is equal to probability of every candidate in the vocabulary V

$$\hat{w} = \underset{w \in C}{\operatorname{argmax}}\, \overbrace{P(x|w)}^{\text{channel model}}\ \overbrace{P(w)}^{\text{prior}}$$

Edit distance Language model

The probability that x would be typed when the user meant w

Spelling Correction Algorithm

Non-word errors are detected by looking for any word not found in a dictionary

For example:

- the typed word "graffe" is not present in the dictionary
- the larger the dictionary the better

To correct non-word spelling errors

1. Generate candidate words: real words having a similar letter sequence to the error
   - Ex.: candidates for the word "graffe" might include giraffe, graf, gaffe, grail

2. Rank the candidates using edit distance between the error word and the candidates
   - First words with edit distance = 1, then edit distance = 2

3. **Choose the candidate word** which maximizes the estimate

$$= \underset{w \in V}{\operatorname{argmax}} P(x \mid w) P(w)$$

28

Spelling Correction Example

Supose the user has typed **acress**

The following algoritm can solve the spell correction problem:

Candidate Generation

Find words with similar spelling → small edit distance to error

Words within 1 of acress

Error	Candidate Correction	Correct Letter	Error Letter	Type
acress	actress	t	–	deletion
acress	cress	–	a	insertion
acress	caress	ca	ac	transposition
acress	access	c	r	substitution
acress	across	o	e	substitution
acress	acres	–	s	insertion
acress	acres	–	s	insertion

- **80% of errors are within edit distance 1**
- **Almost all errors within edit distance 2**

29

Channel Model: Edit Distance

$$\hat{w} = \underset{w \in C}{\operatorname{argmax}} \overbrace{P(x|w)}^{\text{channel model}} \overbrace{P(w)}^{\text{prior}}$$

Edit distance — Language model

Candidate Correction	Correct Letter	Error Letter	x|w	P(x|w)
actress	t	-	c|ct	.000117
cress	-	a	a|#	.00000144
caress	ca	ac	ac|ca	.00000164
access	c	r	r|c	.000000209
across	o	e	e|o	.0000093
acres	-	s	es|e	.0000321
acres	-	s	ss|s	.0000342

Figure 6.4 Channel model for acress; the probabilities are taken from the *del*[], *ins*[], *sub*[], and *trans*[] confusion matrices as shown in Kernighan et al. (1990).

- 80% of errors are within edit distance 1
- Almost all errors within edit distance 2

Channel Model and Language Model

$$\hat{w} = \underset{w \in C}{\operatorname{argmax}} \overbrace{P(x|w)}^{\text{channel model}} \overbrace{P(w)}^{\text{prior}}$$

Edit distance — Language model

Candidate Correction	Correct Letter	Error Letter	x|w	P(x|w)
actress	t	-	c|ct	.000117
cress	-	a	a|#	.00000144
caress	ca	ac	ac|ca	.00000164
access	c	r	r|c	.000000209
across	o	e	e|o	.0000093
acres	-	s	es|e	.0000321
acres	-	s	ss|s	.0000342

Figure 6.4 Channel model for acress; the probabilities are taken from the *del*[], *ins*[], *sub*[], and *trans*[] confusion matrices as shown in Kernighan et al. (1990).

Unigram LM:

w	count(w)	p(w)
actress	9,321	.0000231
cress	220	.000000544
caress	686	.00000170
access	37,038	.0000916
across	120,844	.000299
acres	12,874	.0000318

Use any N-Gram Model (unigram, bigram, trigram)

Channel Model and Language Model

$$\hat{w} = \underset{w \in C}{\operatorname{argmax}} \overbrace{P(x|w)}^{\text{channel model}} \overbrace{P(w)}^{\text{prior}}$$

Edit distance   Language model

Candidate Correction	Correct Letter	Error Letter	x|w	P(x|w)
actress	t	-	c|ct	.000117
cress	-	a	a|#	.00000144
caress	ca	ac	ac|ca	.00000164
access	c	r	r|c	.000000209
across	o	e	e|o	.0000093
acres	-	s	es|e	.0000321
acres	-	s	ss|s	.0000342

Figure 6.4 Channel model for acress; the probabilities are taken from the *del*[], *ins*[], *sub*[], and *trans*[] confusion matrices as shown in Kernighan et al. (1990).

Unigram LM:

w	count(w)	p(w)
actress	9,321	.0000231
cress	220	.000000544
caress	686	.00000170
access	37,038	.0000916
across	120,844	.000299
acres	12,874	.0000318

=> unnorm. posterior:

Candidate Correction	Correct Letter	Error Letter	x|w	P(x|w)	P(w)	10^9*P(x|w)P(w)
actress	t	-	c|ct	.000117	.0000231	2.7
cress	-	a	a|#	.00000144	.000000544	0.00078
caress	ca	ac	ac|ca	.00000164	.00000170	0.0028
access	c	r	r|c	.000000209	.0000916	0.019
across	o	e	e|o	.0000093	.000299	2.8
acres	-	s	es|e	.0000321	.0000318	1.0
acres	-	s	ss|s	.0000342	.0000318	1.0

Figure 6.5 Computation of the ranking for each candidate correction, using the language model shown earlier and the error model from Fig. 6.4. The final score is multiplied by 10^9 for readability.

19

Improving the model using bigram

Unfortunately, "across" is not the intend word

- "a stellar and versatile **acress** whose combination of sass and glamour…"
- Counts from the Corpus of Contemporary American English with add-1 smoothing
- `P(actress|versatile)=.000021 P(whose|actress) = .0010`
- `P(across|versatile) =.000021 P(whose|across) = .000006`

- `P("versatile actress whose") = .000021*.0010 = 210 x10`$^{-10}$
- `P("versatile across whose") = .000021*.000006 = 1 x10`$^{-10}$

28

Improving the model using bigram

- "a stellar and versatile **acress** whose combination of sass and glamour…"
- Counts from the Corpus of Contemporary American English with add-1 smoothing
- P(actress|versatile)=.000021 P(whose|actress) = .0010
- P(across|versatile) =.000021 P(whose|across) = .000006

- **P("versatile actress whose") = .000021*.0010 = 210 x10^{-10}**
- P("versatile across whose") = .000021*.000006 = 1 x10^{-10}

29

The correct word is predicted in this case

34

Language Identification

UFRPE Universidade Federal Rural de Pernambuco

DEINFO Departamento de Estatística e Informática

14-abr-18

Cavnar´s Method [1994]

- One of the most successful approaches is the N-gram approach, introduced by (Cavnar et al. 94).
- They were more concerned with **text categorization**, but they found out that their method also performed very well on the task of **language identification**.
- The main idea of using n-grams for language identification is that every language uses certain n-grams more frequently than others, hence providing a clue about the language (**language profile**)

The N-Gram Approach by Cavnar (1994)

Cavnar used N-gram at the character level

Word: GARDEN

bi-grams: _G, GA, AR, RD, DE , EN , N_

tri-grams: _GA, GAR, ARD, RDE, DEN, EN_

quad-grams: _GAR, GARD, ARDE, RDEN, DEN_

Cavnar´s Algorithm

- **Phase 1. Building Language Profiles**
- **Phase 2. Identifying the Document Language**

Cavnar´s Algorithm

Phase 1. (Building Language Profiles)

1. The sample texts in several languages are read one by one and all punctuation marks are deleted. Each word becomes a token delimited by white space before and after.
2. All tokens are scanned and N-grams with **n = 2..5** are produced from these tokens.
3. The N-grams are stored in a hash table and for each occurrence the counter for the N-gram in question is increased.
4. After that, the hash is ordered starting with the most frequent N-grams.
5. This procedure is repeated for each language.

The N-gram hash tables constitute the N-gram profiles for each language.

Cavnar´s Algorithm

Phase 2. Identifying the document Language

The distance is calculated in the following way.

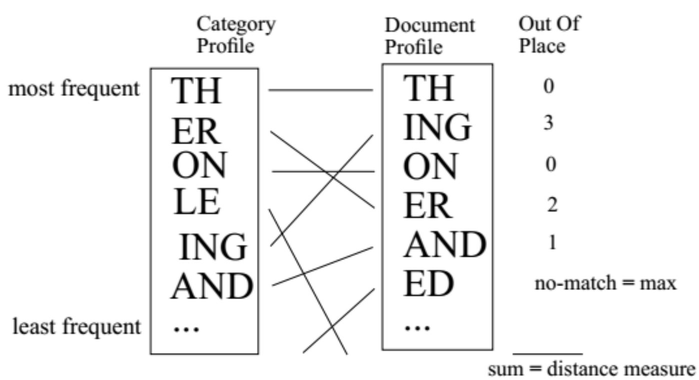
1. For each N-gram in our test document, there can be a corresponding one in the current language profile we are comparing it to.
2. N-grams having the same rank in both profiles receive a zero distance.
3. If the respective ranks for an N-gram vary, they are assigned the number of ranks between the two with a maximum distance of 3
4. Finally all individual N-gram rank distances are added up.
   - This number is now the distance between the sample document and the current language profile.
   - This step is repeated until the sample document has been compared to all language profiles in question.

- The smallest distance among them all identifies the language.

Cavnar´s Algorithm

Phase 2. Identifying the document Language

FIGURE 3. Calculating The Out-Of-Place Measure Between Two Profiles



Note: These profiles are for explanatory purposes only and do not reflect real N-gram frequency statistics.

Keyword Extraction

UFRPE
Universidade Federal Rural de Pernambuco
DEINFO
Departamento de Estatística e Informática

14-abr-18

Key Term Extraction using N-gram IDF

Input: "Alice' s Adventures in Wonderland – Kindle edition by Lewis Carroll"

N-gram	IDF_{N-gram}	N-gram	IDF_{N-gram}
kindle edition	12.043	adventures	7.101
kindle	11.653	kindle edition by	6.739
alice s adventures in wonderland	11.496	lewis	6.192
adventures in wonderland	10.906	edition	4.836
s adventures in wonderland	10.804	adventures in	4.280
wonderland	9.670	s adventures	3.586
lewis carroll	9.498	alice s	3.507
alice s adventures	9.385	s adventures in	2.255
alice s adventures in	9.348	by lewis	1.768
in wonderland	8.762	s	1.030
carroll	8.152	by	0.820
by lewis carroll	7.461	in	0.154
alice	7.234	edition by	-0.875

18/27

Cryptoanalysis
“Code Breaking”
(other slides)

UFRPE
Universidade Federal Rural de Pernambuco

DEINFO
Departamento de Estatística e Informática

14-abr-18