Lexicon Computing
基本词汇计算

Jingbo Xia

Huazhong Agricultural University

xiajingbo.math@gmail.com

2023-02-

Table of contents I

1. Lexicon-centric NLP Computation 6
   - Type-Term-Ratio, Lexical Complexity/语言复杂度 7
   - Information Retrieval/信息抽取 10
   - N-Gram and Language Model/针对词汇依赖概率的语言模型 14
   - Bag-of-Words Model, Representative of Document with Lexicon Count/基于词汇的文本表示 17

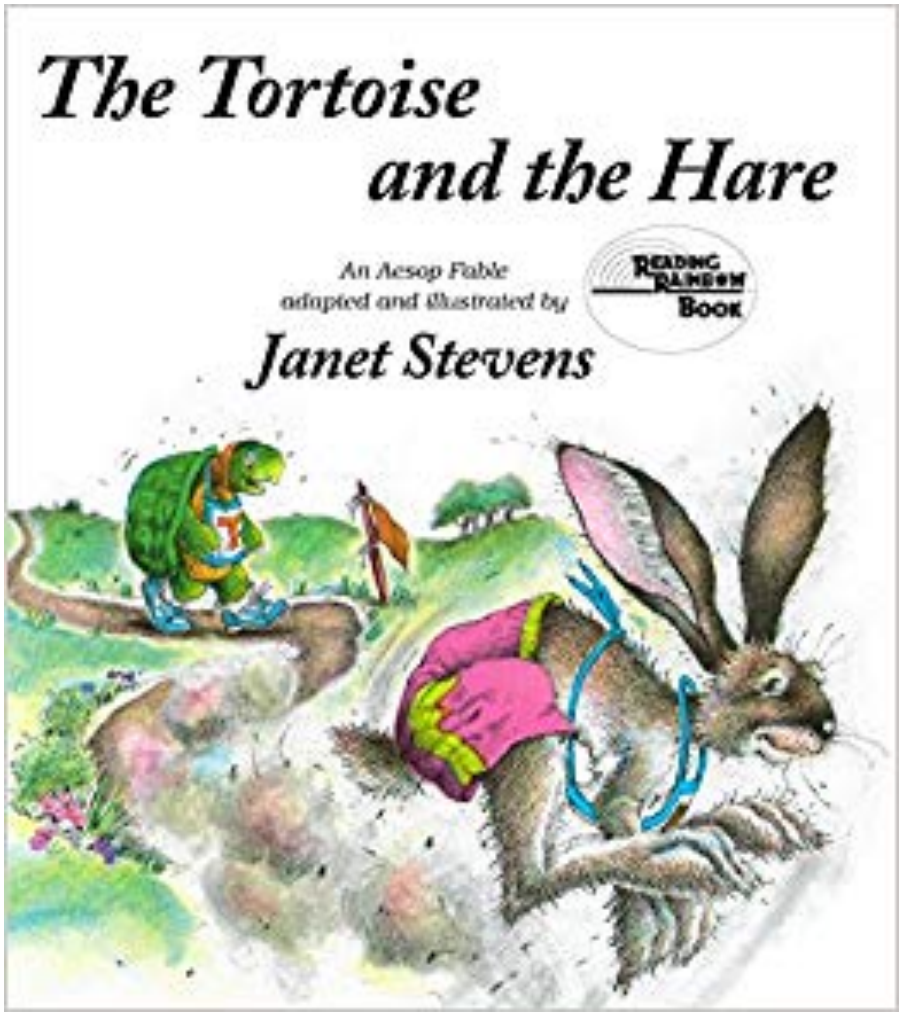
2. Count-based Vector Space Word Representation 20
   - Word w and document d/ TFIDF 21
   - Word w and context word c/互信息 26
   - From sparse representation to dense one, from SVD to LSA/潜在语义分析 27

在 NLP 中，为什么要将词汇/段落/文章进行向量化？

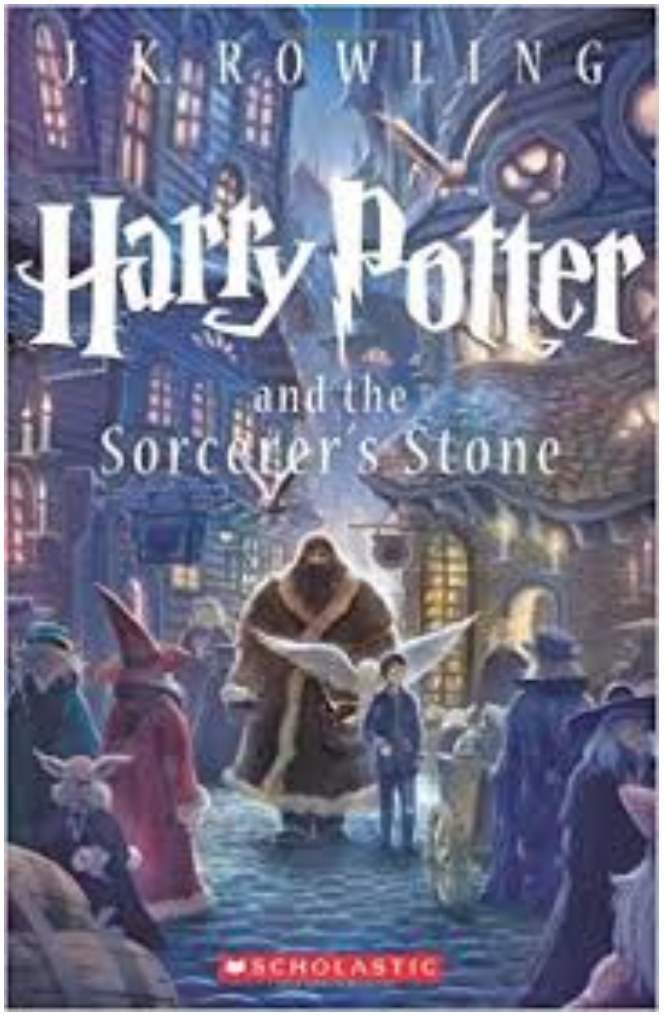
随机向量？Word representation.
文本分类？主题建模？

Lexicon-centric NLP Computation I

Type-Term-Ratio, Lexical Complexity/语言复杂度



vs.



Lexicon-centric NLP Computation III

Type-Term-Ratio, Lexical Complexity/语言复杂度

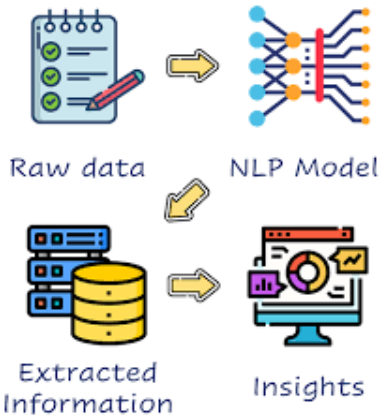
TTR (Type-Token Ratio) is the ratio obtained by dividing the types (the total number of different words) occurring in a text or utterance by its tokens (the total number of words).

$$\text{TTR} = \frac{\text{Amount of unique words}}{\text{Amount of total words}}.$$

A high TTR indicates a high degree of lexical variation while a low TTR indicates the opposite.

Lexicon-centric NLP Computation I

Information Retrieval/信息抽取



Unstructured data

"AP: 72 yo f w/ ESRD on HD, CAD, HTN asthma p/w significant hyperkalaemia & associated arrhythmias." [1]

Abbreviations | Acronyms | Medical jargons

Meaningful information

Attributes	Medical text
Age	72 years old (yo)
Gender	female (f)
Condition	hypertensive disease (HD)
Symptom	hyperkalemia

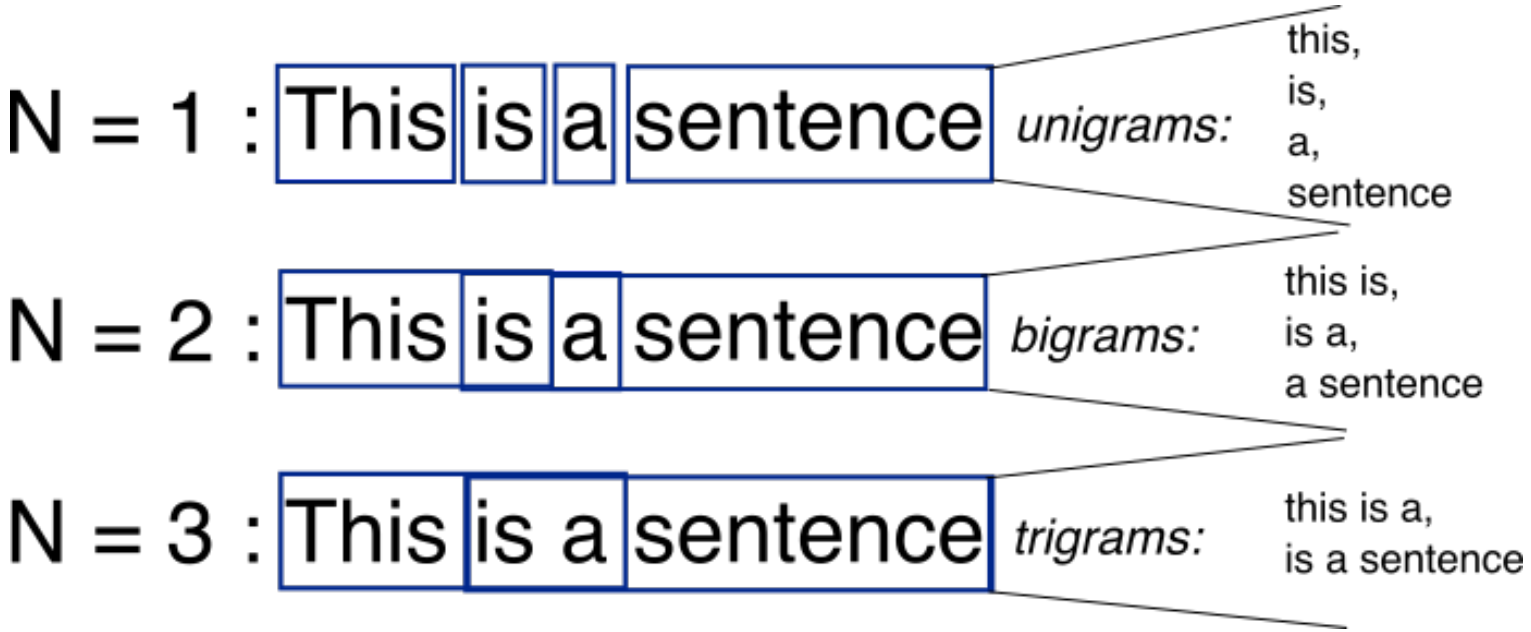
信息抽取（information extraction），简称 IE，即从自然语言文本中，抽取出特定的事件或事实信息，帮助我们将海量内容自动分类、提取和重构。这些信息通常包括实体（entity）、关系（relation）、事件（event）。例如从新闻中抽取时间、地点、关键人物，或者从技术文档中抽取产品名称、开发时间、性能指标等。能从自然语言中抽取用户感兴趣的事实信息，无论是在知识图谱、信息检索、问答系统还是在情感分析、文本挖掘中，信息抽取都有广泛应用。
信息抽取主要包括：
关系抽取：通常我们说的三元组 (triple) 抽取，主要用于抽取实体间的关系。
实体抽取：也就是命名实体识别。[a]

[a] https://cloud.tencent.com/developer/article/1680805.

Information Retrieval (IR) Compute the relevance of a document for a given word.
e.g., (word $\times$ document) matrix!

Count-based Vector Space Word Representation I

N-Gram and Language Model/针对词汇依赖概率的语言模型



Count-based Vector Space Word Representation III

N-Gram and Language Model/针对词汇依赖概率的语言模型

N-gram 基于这样的想法，它的第一个特点是某个词的出现依赖于其他若干个词，第二个特点是我们获得的信息越多，预测越准确。我想说，我们每个人的大脑中都有一个 N-gram 模型，而且是在不断完善和训练的。

N-gram 模型是一种语言模型（Language Model，LM），语言模型是一个基于概率的判别模型，它的输入是一句话（单词的顺序序列），输出是这句话的概率，即这些单词的联合概率（joint probability）。[a]

[a] https://blog.csdn.net/songbinxu/article/details/80209197.

Count-based Vector Space Word Representation III

Bag-of-Words Model, Representative of Document with Lexicon Count/基于词汇的文本表示

词袋模型能够把一个句子转化为向量表示，是比较简单直白的一种方法，它不考虑句子中单词的顺序，只考虑词表（vocabulary）中单词在这个句子中的出现次数。[a]

词袋模型是 n-gram 语法模型的特例 1 元模型。该模型忽略掉文本的语法和语序等要素。[b]

[a] https://blog.csdn.net/u012328159/article/details/84719494.
[b] https://zhuanlan.zhihu.com/p/53302305.

Count-based Vector Space Word Representation II

Word w and document d/ TFIDF

Term-Document matrix!

	Romance	Thriller	Detective	War	History	Cartoon
心跳	20	20	17	19	1	0
信物	90	0	1	0	0	0
娃娃	3	6	2	3	0	109
宝贝	2034	15	22	0	1	29
战马	1	0	2	23	28	0
马克沁机枪	0	0	0	23	2	0
罪犯	1	6	68	9	8	0
⋮		...	...			
崇祯	0	0	0	0	8	0
成吉思汗	0	0	0	0	14	0
嘉峪关	0	0	0	2	26	0
黑色	23	34	35	7	8	16
早上	129	214	334	64	29	123
的	534	896	345	1023	789	1084
你	633	597	543	825	886	721

Count-based Vector Space Word Representation III

Word w and document d/ TFIDF

Assume there are N documents. The below is a simple way to vectorize a word w^i:

$$\boldsymbol{w}_i = (\boldsymbol{w}_{i1}, \boldsymbol{w}_{i2}, \cdots, \boldsymbol{w}_{iN}),$$

here $\boldsymbol{w}_{in}$ = number of occurrences of w^i in document d^n, $n = 1, 2, \cdots, N$

In another word, the term-document matrix $tf(w^i, d^n)_{I \times N}$ stacks all rows of $\boldsymbol{w}_i$.

	d^1	d^2	$\cdots$	d^N	
w^1					$=\boldsymbol{w}_1$
w^2					$=\boldsymbol{w}_2$
$\vdots$			$tf(w^i, d^n)$		$\vdots$
w^I					$=\boldsymbol{w}_I$

Count-based Vector Space Word Representation IV

Word w and document d/ TFIDF

One may add the strength of association, e.g., TF·IDF,

$$tf \cdot idf(w^i, d^n) = tf(w^i, d^n) \times idf(w^i),$$

here $tf(w^i, d^n)$ is the term frequency of w^j in the j-th document, and inverse document frequency, $idf(w^i) = \log_e \frac{N}{\#\{d^n | w^i \text{ in } d^n\}}$.

	d^1	d^2	$\cdots$	d^N	
w^1					$=\boldsymbol{w}_1$
w^2					$=\boldsymbol{w}_2$
$\vdots$			$tf \cdot idf(w^i, d^n)$		$\vdots$
w^I					$=\boldsymbol{w}_I$

Count-based Vector Space Word Representation

Word w and context word c/互信息

Assume we consider J context word around the word w^i, the word w^i is vectorized by all the context word c^j, $(j = 1, 2, \cdots, J)$:

$$\boldsymbol{w}_i = (\boldsymbol{w}_{i1}, \boldsymbol{w}_{i2}, \cdots, \boldsymbol{w}_{iJ}),$$

here $\boldsymbol{w}_{ij} =$ number of co-occurrences of w^i with context word c^j.

Point-wise mutual information is used to consider further word-context info.

$$PMI(w^i, w^j) = \log_2 \frac{P(w^i, w^j)}{P(w^i)P(w^j)},$$

here $P(w^i, w^j)$ calculates the probability of two words co-occurring in a context.

Please noted that $\frac{P(w^i, w^j)}{P(w^i)P(w^j)} = \frac{P(w^i|w^j)}{P(w^i)}$, or $\frac{P(w^j|w^i)}{P(w^j)}$.

Count-based Vector Space Word Representation I

From sparse representation to dense one, from SVD to LSA/潜在语义分析

在谈论潜在语义分析之前，请回忆一个矩阵分解算法——奇异值分解 (Singular Value Decomposition) [a]。

问题

- SVD 的算法对象，步骤，和结果是什么？
- SVD 带来哪些应用？

[a]以生物信息专业课程为例，《生物信息数学基础（代数部分）》中有关 SVD 的章节安排

Count-based Vector Space Word Representation II

From sparse representation to dense one, from SVD to LSA/潜在语义分析

The representation is always sparse.
Singular Value Decomposition (SVD) –> Latent Semantic Analysis (LSA)

定理 (Singular Value Decomposition)

For an $m \times n$ matrix A of rank r, there exists a factorization (Singular Value Decomposition $=$ SVD) as follows:

$$A = U\Sigma V^T,$$

where U and V are orthogonal matrices, $\Sigma = \begin{pmatrix} \Delta & 0 \\ 0 & 0 \end{pmatrix}$, $\Delta = diag(\sigma_1, \cdots, \sigma_r)$, $\sigma_i = \sqrt{\lambda_i}$ (which is called as singular value of A), $i = 1, 2, \cdots, r$, $r = Rank(A)$, λ_i is the eigenvalue of AA^T.

Generally, $\sigma_{i=1}^{R} U_i V_i^T$ is a rank-R approximation of an arbitrary A, which yields to a dense matrix!

Count-based Vector Space Word Representation III

From sparse representation to dense one, from SVD to LSA/潜在语义分析

The idea of LSA.

> 潜在语义分析是一种无监督学习方法，主要用于文本的话题分析；其特点是通过矩阵分解发现文本与单词之间的基于话题的语义关系；最初应用于文本信息检索。[a]
>
> [a] https://cloud.tencent.com/developer/article/1659515.

	Romance	Thriller	Detective	War	History	Cartoon
Term 1						
Term 2						
⋮			Sparse!			
Term V						

SVD converts an original sparse term-document matrix a dense one. LSA uses dense rows to represent the embedding of the terms.