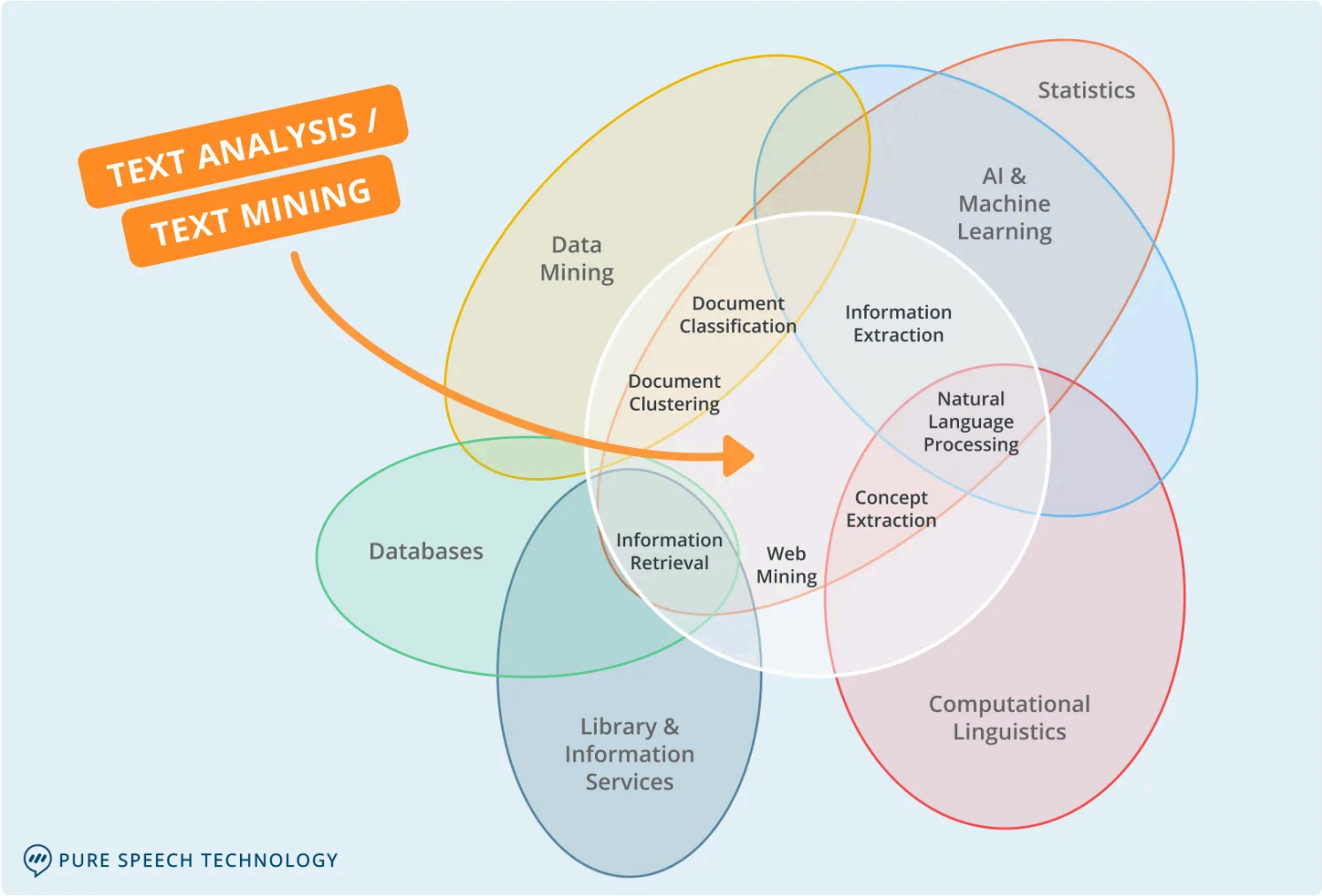
Market Research and Analysis

Lecture 3: Natural Language Process and Text Analysis

- Zhenyu Zhao
- Nankai Institute of International Economics
- Feishu: 2120253538
- Email: zzynankai@outlook.com
- Website: xishanyu2.github.io


TEXT ANALYSIS /
TEXT MINING
Data Mining
Statistics
AI & Machine Learning
Document Classification
Information Extraction
Document Clustering
Natural Language Processing
Concept Extraction
Databases
Information Retrieval
Web Mining
Library & Information Services
Computational Linguistics
PURE SPEECH TECHNOLOGY

01

自然语言处理发展历程

NLP is a subfield of AI that enables computers to understand, interpret, and generate human language through algorithms and models.

机器语言（Machine Language）、自然语言（Natural Language）——人类日常使用的语言（如中文、英文）。

自然语言处理（NLP）：让计算机理解、处理人类语言的技术。

任务分类

自然语言理解（NLU）：分类/回归任务，如文本的主题、情绪、词性，下一句预测（next sentence prediction），自然语言推理（natural language inference）——判断两段之间是否存在特定逻辑关系，公式：

$$P(y|S_1, S_2, ..., S_N) = ?$$

- y是预测的标签；$S_1, S_2, ..., S_N$是文字序列，顺序不能轻易打乱。

自然语言生成（NLG）：生成任务，如翻译、摘要生成、ChatGPT开放式对话，公式：

$$P(S_{N+1}|S_1, S_2, ..., S_N) = ?$$

- 基于S_{N+1}继续预测$S_{N+2}, S_{N+3}...$，直到形成一个完整的序列。

核心差异

NLU：仅需编码（Encoding/Embedding），NLG：需编码+解码（Decoding/Generation）

NLU的核心问题：如何对文本进行编码（将语义信息转化为数值向量，模型训练.....）

1. 最简单的编码模型：TF-IDF文本向量

TF-IDF是一种基于词频统计的编码模型，它通过计算单词在文本中的**出现频率（TF）** 和在整个数据集中的**逆文档频率（IDF）** 来衡量单词的重要性。公式：

$$TF(t,d) = \frac{单词t在文本d中出现的次数}{d中所有的单词数}$$

$$IDF(t) = ln(\frac{文本总数N}{1 + 包含单词t的文本数量n_t})$$

$$TF - IDF = TF(t,d) \times IDF(t)$$

- TF：一个单词在单个文本中出现的频率。问题：出现频率很高但不含特定语义信息的词、只在特定文本中频繁出现而在其它所有文本中均未出现的词？解决：n_t/N调权重。
- IDF：+1避免$n_t = 0$（一个单词在所有文本中均未出现），假设强迫每个单词都至少出现一次；取对数增强数值稳定性：$n_t \to N, N/(1+n_t) \to 1$；$n_t = 1, N/(1+n_t) = N/2$，N越大IDF主导（低频词的重要性被过度放大）。

词频无法准确地反映出语义信息；数据集中的单词数非常多，TF-IDF对文本编码得到的向量是高维且稀疏的。

"The professor is presenting his research in a meeting."
"The researcher is talking about his study in a seminar."
编码向量应该非常接近，但TF-IDF做不到。

2. 潜在语义分析（Latent Semantic Analysis, LSA）

通过奇异值分解（SVD）**降维**捕捉**语义**，适用于文本相似性检索和聚类等任务。

$$X = U\Sigma V^T$$

- 文本数N，不同的单词数M，数据集D
- X：TF/TF-IDF向量组成的N×M维矩阵
- U：N×r维矩阵，对N个文本进行了r维编码
- V：M×r维矩阵，对M个单词进行了r维编码

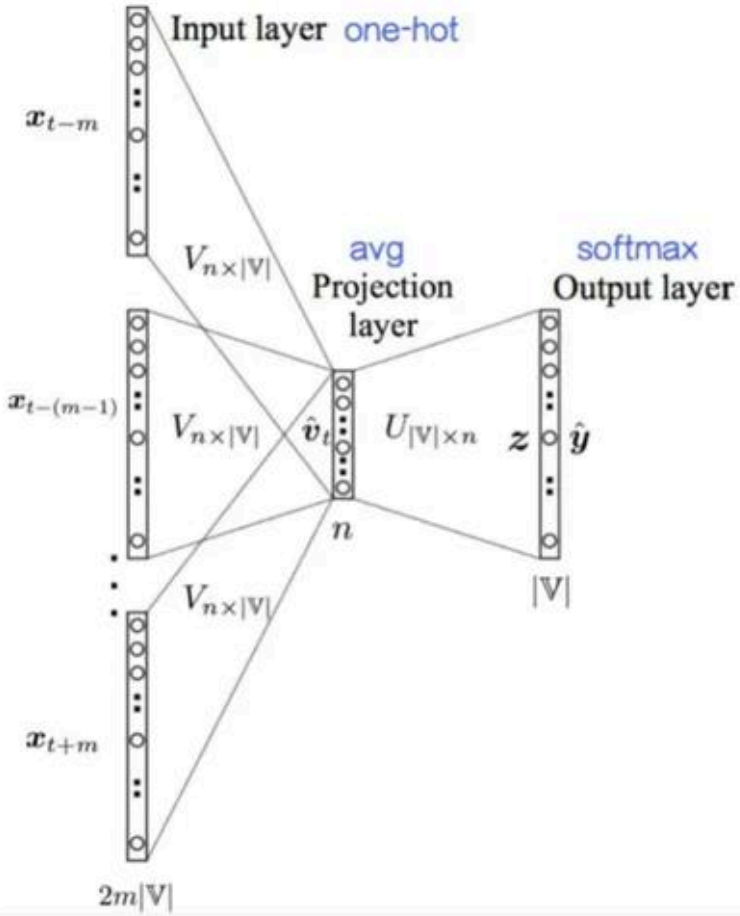
r远小于TF-IDF向量维度，U和V将会是稠密的，就可以适当地把语义表现出来；当有足够多的文本时，许多词汇之间存在共现性（Co-occurrence），例如“老师”总是和“学校”“学生”“教室”这些词一起出现。

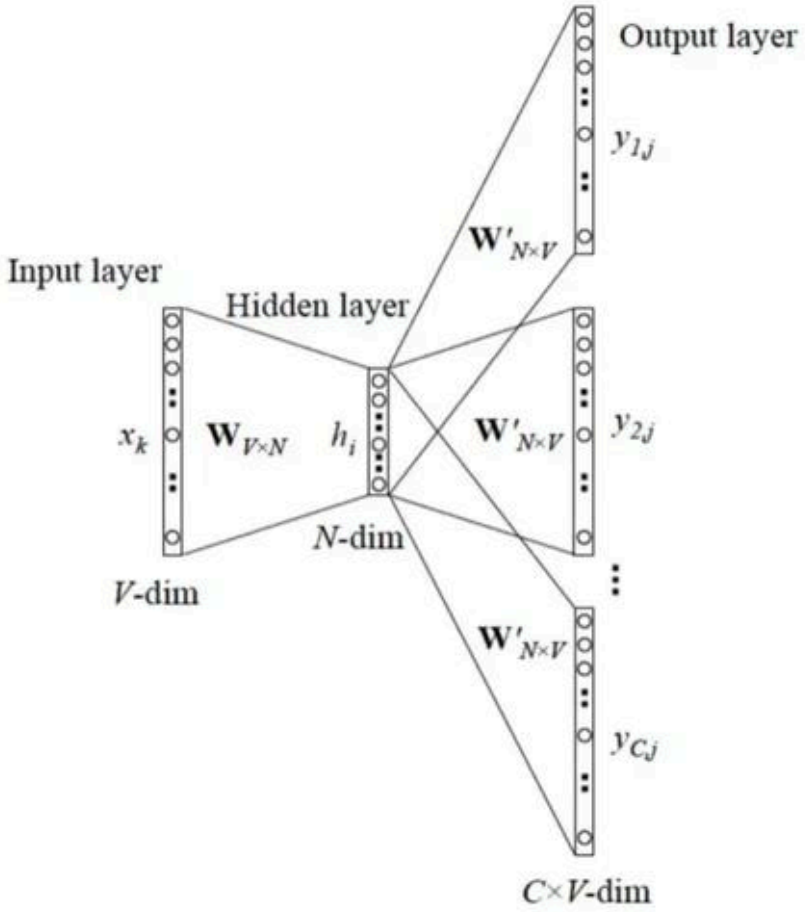
基于词频的TF-IDF和LSA缺乏通用性，如基于新闻数据集所建立的编码模型难以迁移至电影评论数据集。但每个词汇的意义具有普适性，具体到每个文本上可能会因为上下文不同而有细微差异。

3. 敲开自监督预训练的大门: Word2Vec

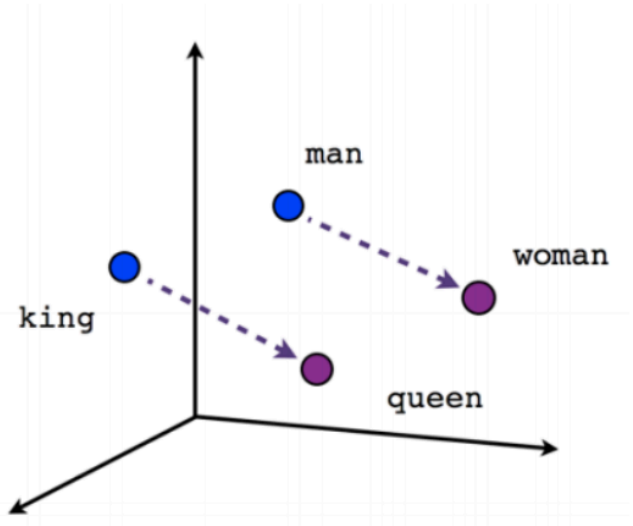
在2013年至2014年期间，Google公司以及斯坦福大学都各自开发了自己的预训练词编码模型，分别被称作Word2Vec和GloVe。**Word2Vec**设计了一个单层的神经网络，然后通过自监督（Self-supervised）方法来训练这个神经网络，等训练完成后，将这个网络中的中间层作为对词汇的编码来达成目标。而**GloVe**则是通过建立字词之间的共现矩阵（Co-occurrence Matrix）来计算一些统计性质，从而达成对词汇进行编码的目的。

Word2Vec是一种基于滑动窗口来建立训练数据，并用该数据来训练一个单层全连接神经网络的编码方法。它有两种训练策略：**连续词袋模型**（Continuous Bag-of-Words Model, **CBOW**）模型和**连续跳字模型**（Continuous **Skip-gram** Model）。CBOW模型的目标是根据上下文单词预测目标单词，而Skip- gram模型则是根据目标单词预测上下文单词。

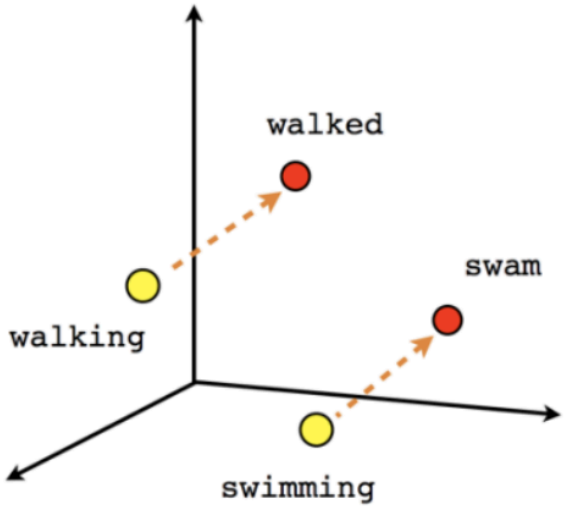

Input layer one-hot
x_{t-m}
$V_{n\times|\mathbb{V}|}$
avg
Projection layer
softmax
Output layer
$x_{t-(m-1)}$
$V_{n\times|\mathbb{V}|}$
$\hat{v}_t$
$U_{|\mathbb{V}|\times n}$
z
$\hat{y}$
n
$V_{n\times|\mathbb{V}|}$
$|\mathbb{V}|$
x_{t+m}
$2m|\mathbb{V}|$


Output layer
y_{1j}
$\mathbf{W}'_{N\times V}$
Input layer
Hidden layer
x_k
$\mathbf{W}_{V\times N}$
h_i
$\mathbf{W}'_{N\times V}$
y_{2j}
N-dim
V-dim
$\mathbf{W}'_{N\times V}$
y_{Cj}
$C\times V$-dim

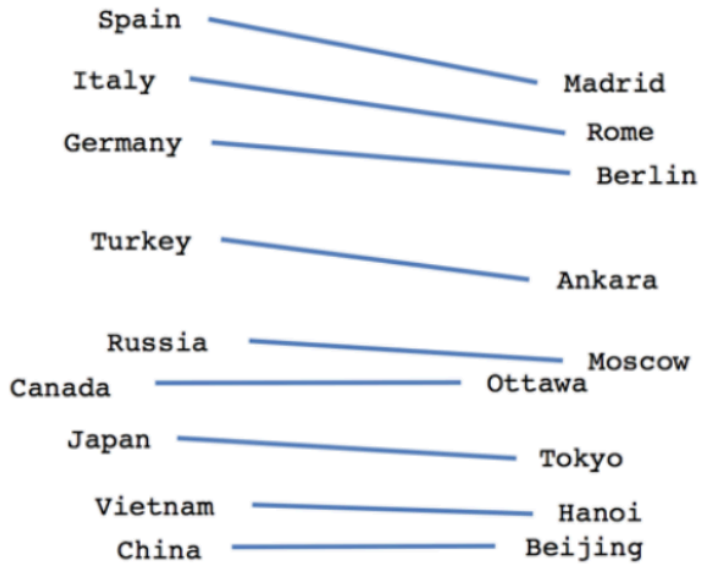
聚类特性和代数特性



Male-Female



Verb tense



Country-Capital

总结

CBOW：通过上下文预测中心词（训练快，适合高频词）。
让模型学会哪些单词更频繁地一起出现，从而捕捉单词之间的**共现关系**。

Skip-gram：通过中心词预测上下文（语义表示更优，适合低频词）。
让模型学会在给定一个单词的情况下，哪些单词更有可能与之共同出现，捕捉单词之间的**语义关系**。

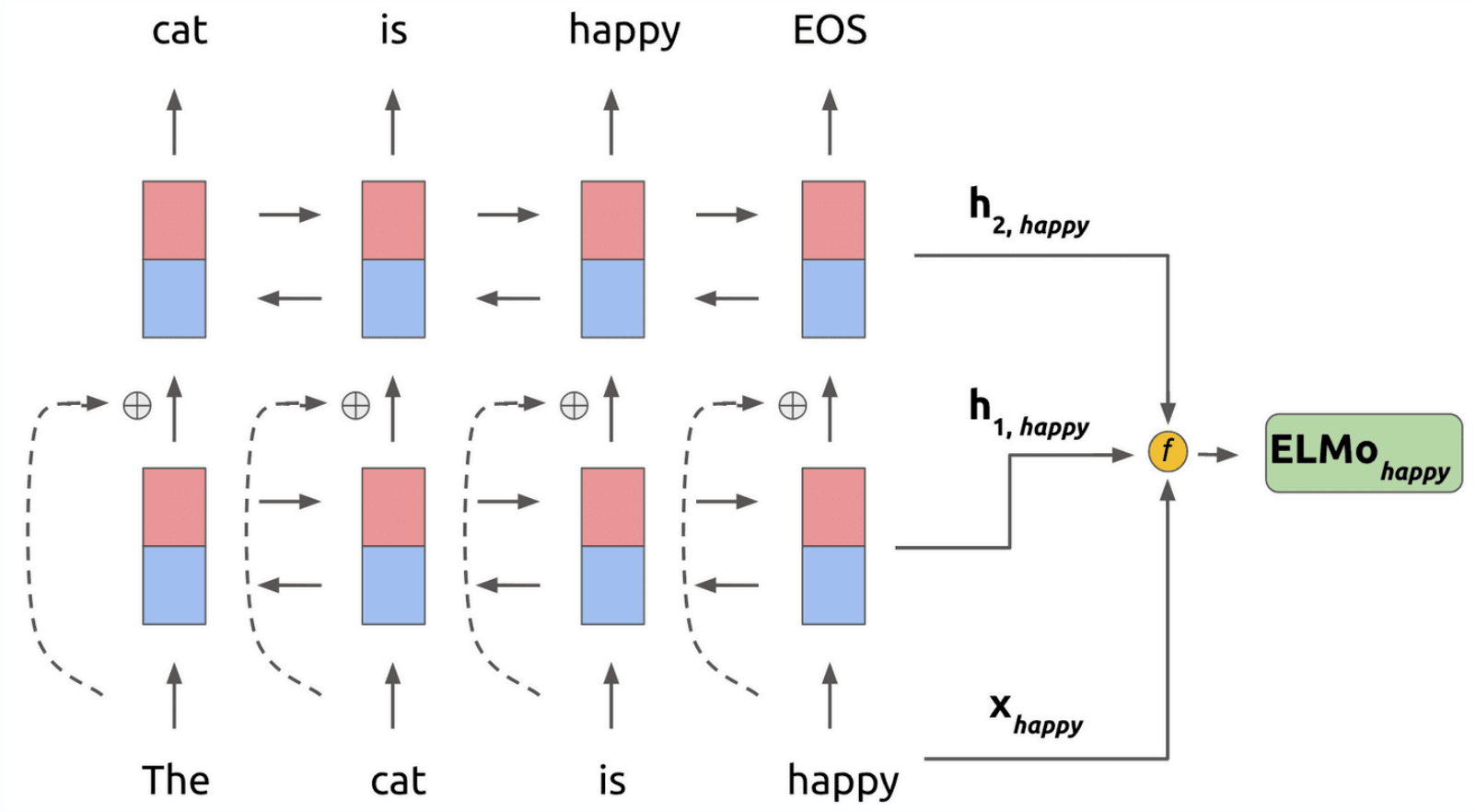
Word2Vec模型局限性：Word2Vec模型虽然能够捕捉单词之间的语义关系，但它存在一些局限性。首先，它是一种**静态词嵌入**模型，无法根据上下文动态调整词向量。 例如，“苹果”在不同的上下文中可能表示水果、科技公司或宗教象征，但Word2Vec模型生成的词向量是固定的，无法区分这些不同的语义。其次，Word2Vec模型对**低频词**的处理不够理想，因为低频词在训练数据中出现次数较少，模型难以学习到其准确的语义信息。

4. 基于上下文的编码模型：ELMo

Allen Institute和华盛顿大学合作开发的ELMo（Embedding from Language Model，基于语言模型的嵌入）模型。尽管ELMo模型在发表后不久就被更先进的BERT模型所超越，因此并未得到广泛应用，但由于其设计理念直观，仍值得进行简要介绍。

ELMo是一种基于上下文的词嵌入模型，它通过**双向长短期记忆网络**（Bi-directional Long Short-Term Memory, biLSTM）对文本进行编码，能够根据上下文动态调整词向量，解决了词义歧义问题。

ELMo模型的具体架构包括一个词嵌入层和两层biLSTM。每个词首先被转换为词向量，然后输入到两层biLSTM中进行编码。在每个时间点，biLSTM的状态向量包含了当前输入词的编码、从前往后读取的LSTM状态向量和从后往前读取的LSTM状态向量，这确保了模型能够充分利用上下文信息。



总结

4.1 深度双向架构：长距离依赖的全面捕捉

传统单向模型（如前向LSTM）仅能利用历史信息，对长距离依赖（如句子开头与结尾的语义关联）捕捉能力有限。例如，在句子“The man who wore a hat left the room because he was tired”中，理解“he”指代“man”需要跨越多个子句的长距离依赖。

ELMo采用**前向语言模型（Forward LM）** 与**后向语言模型（Backward LM）** 联合训练的方式，同时捕捉单词的前文与后文信息，显著提升长距离依赖捕捉能力。

- **前向LM**：从左到右逐词预测，建模历史信息对当前词的影响。例如，在句子“The cat sat on the mat”中，前向LM根据“The cat sat on the”预测“mat”；
- **后向LM**：从右到左逐词预测，建模未来信息对当前词的约束。例如，后向LM根据“on the mat sat the cat”预测“The”。

模型通过最大化联合概率分布实现双向信息融合：$$p(t_1, t_2, \ldots, t_N) = \prod_{k=1}^{N} p(t_k|t_1, \ldots, t_{k-1}) \cdot p(t_k|t_{k+1}, \ldots, t_N)$$

其中，t_k为第k个单词，N为句子长度。这种双向建模方式使ELMo能够同时利用前文与后文信息，显著提升语义理解的准确性。

4.2 多层表示融合：语法与语义的协同优化

传统模型（如顶层LSTM）仅利用最高层输出，忽略底层语法信息。例如，在词性标注任务中，仅依赖高层语义可能导致对“run”在“He is on a run”中词性的误判。

ELMo采用**多层LSTM架构**，每层输出不同抽象层次的语义表示：

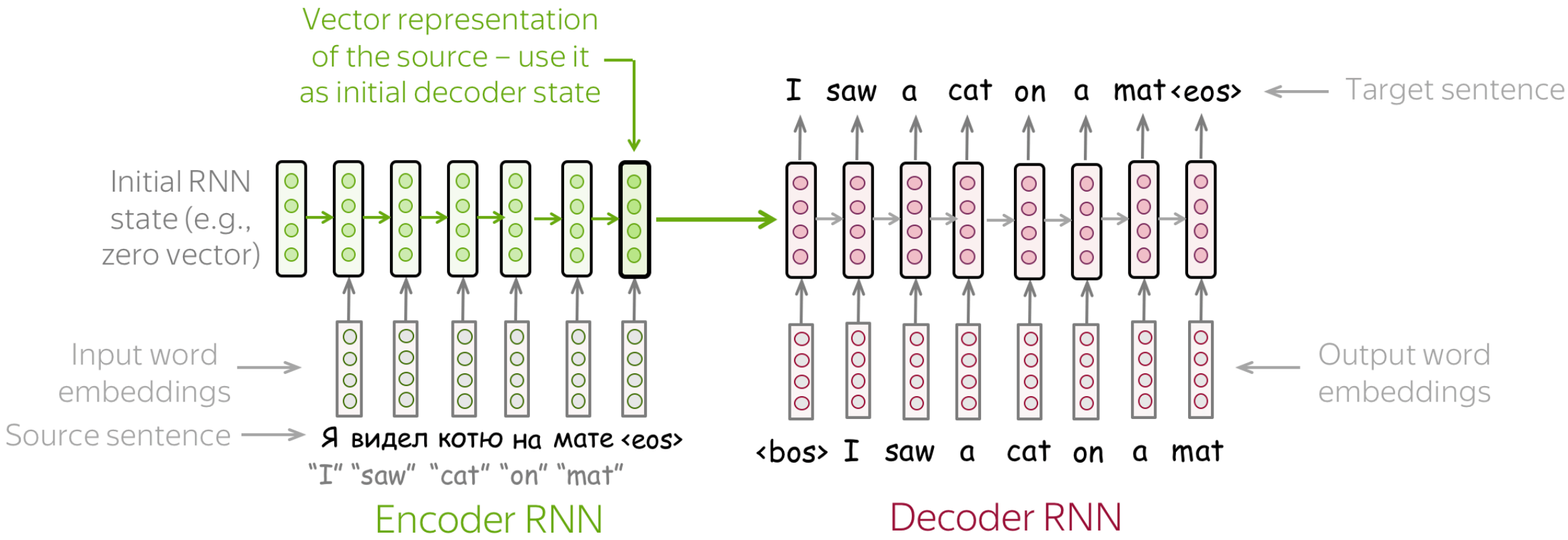
- **底层LSTM**：捕捉局部语法信息（如词性、句法结构）。例如，在句子“Running is fun”中，底层LSTM可识别“Running”为动名词形式；
- **高层LSTM**：捕捉全局语义信息（如主题、情感）。例如，高层LSTM可理解“Running”在“Running a company”中表示“经营”的抽象含义。

最终词向量通过各层输出的加权组合生成：$$\mathrm{ELMo}_k^{\mathrm{task}} = \gamma^{\mathrm{task}} \sum_{j=0}^{L} s_j^{\mathrm{task}} \mathbf{h}_{k,j}^{\mathrm{LM}}$$

其中，$\mathbf{h}_{k,j}^{\mathrm{LM}}$为第$j$层在第$k$个位置的隐藏状态，$s_j$为可学习的层权重，$\gamma$为缩放因子。这种分层融合机制使ELMo能够灵活适应不同任务对语法与语义的需求。

ELMo的核心思想可概括为：**通过双向长短期记忆网络（biLSTM）捕捉单词的上下文信息，生成与语境动态绑定的词向量表示**。

5.0 Seq2Seq模型



5. NLP的典范转移者: Transformer架构

1. 注意力机制（Attention Mechanism）
2. 缩放点积注意力（Scaled Dot-product Attention）：

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

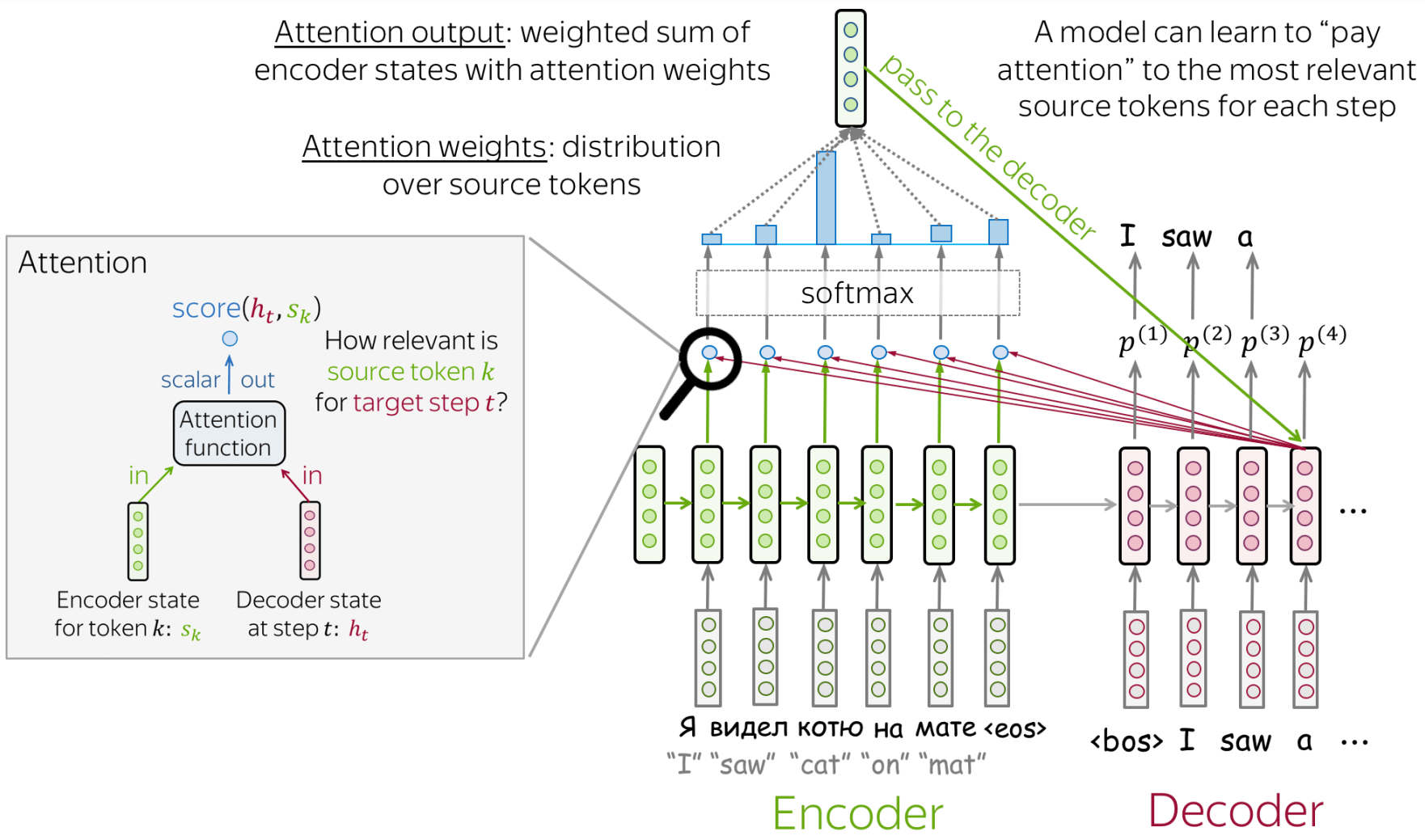
- Q (Query)、K (Key)、V (Value) 均来自输入序列。
- 归一化因子 $\sqrt{d_k}$ 防止Softmax梯度消失。

3. 自注意力（Self-attention）：计算序列内词间相关性。
4. 多头注意力（Multi-head）：并行多个注意力头，增强模型表达能力。

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, \cdots, \text{head}_h)W^O$$

5. 前馈层（Feed Forward）：引入非线性（ReLU），升维再降维（Bottleneck设计）。
6. 位置编码（Positional Encoding）：通过正弦/余弦函数添加位置信息。

5.1 注意力机制


Attention output: weighted sum of encoder states with attention weights
Attention weights: distribution over source tokens
A model can learn to "pay attention" to the most relevant source tokens for each step
pass to the decoder
softmax
I saw a
$p^{(1)}$ $p^{(2)}$ $p^{(3)}$ $p^{(4)}$
Attention
score(h_t, s_k)
How relevant is source token k for target step t?
scalar out
Attention function
in
in
Encoder state for token k: s_k
Decoder state at step t: h_t
Я видел котю на мате <eos>
"I" "saw" "cat" "on" "mat"
<bos> I saw a ...
Encoder
Decoder

Attention Is All You Need

Ashish Vaswani* Google Brain avaswani@google.com

Noam Shazeer* Google Brain noam@google.com

Niki Parmar* Google Research nikip@google.com

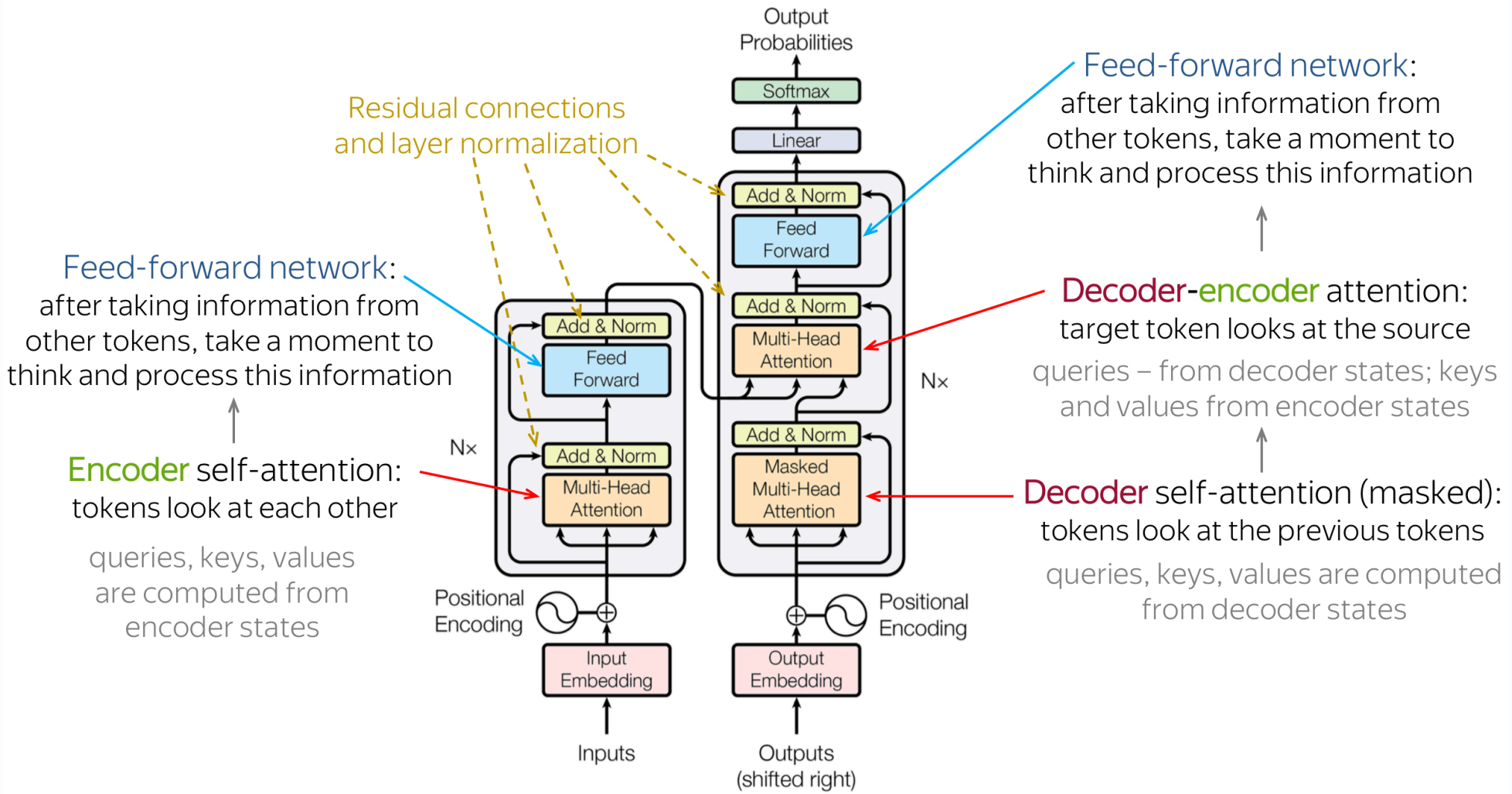
Jakob Uszkoreit* Google Research usz@google.com

Llion Jones* Google Research llion@google.com

Aidan N. Gomez* † University of Toronto aidan@cs.toronto.edu

Łukasz Kaiser* Google Brain lukaszkaiser@google.com

Illia Polosukhin* ‡ illia.polosukhin@gmail.com

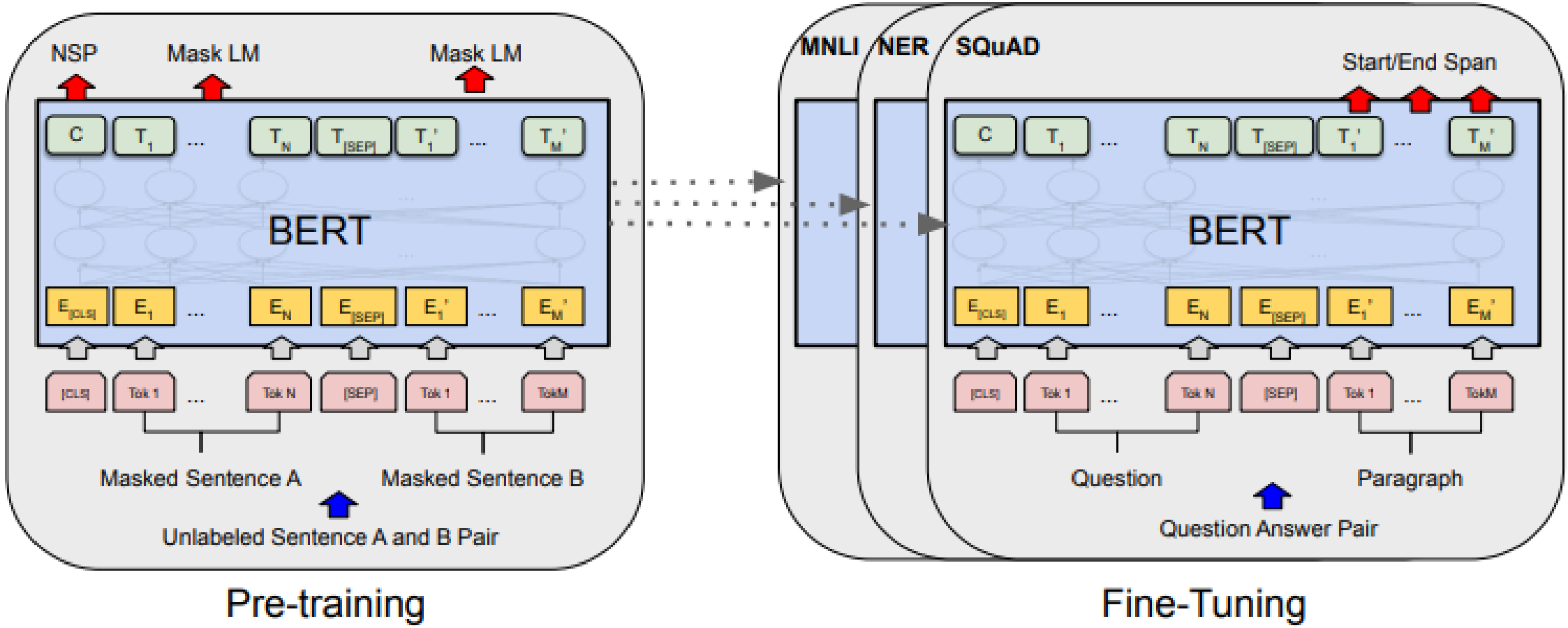

Output Probabilities
Softmax
Linear
Add & Norm
Feed Forward
Add & Norm
Multi-Head Attention
Add & Norm
Masked Multi-Head Attention
Add & Norm
Feed Forward
Add & Norm
Multi-Head Attention
Nx
Nx
Positional Encoding
Positional Encoding
Input Embedding
Output Embedding
Inputs
Outputs (shifted right)
Residual connections and layer normalization
Feed-forward network: after taking information from other tokens, take a moment to think and process this information
Encoder self-attention: tokens look at each other
queries, keys, values are computed from encoder states
Feed-forward network: after taking information from other tokens, take a moment to think and process this information
Decoder-encoder attention: target token looks at the source
queries – from decoder states; keys and values from encoder states
Decoder self-attention (masked): tokens look at the previous tokens
queries, keys, values are computed from decoder states

6. 大语言模型的先锋：BERT与GPT

BERT与GPT是由Google和OpenAI在2018年先后提出的预训练模型架构，它们均基于Transformer架构构建。

BERT的全名是Bidirectional Encoder Representations from Transformers (来自变换器的双向编码器表示)。

BERT基于Transformer架构，采用**双向编码**方式，通过**掩码语言模型**（Masked Language Model）和**下一句预测任务**（Next Sentence Prediction）进行预训练，适用于多种自然语言理解任务，如问答系统、文本分类等。简而言之，BERT通过不断堆叠Transformer(Trm)层，增加模型的深度和参数数量，以处理更大规模的数据。


NSP
Mask LM
Mask LM
BERT
Masked Sentence A
Masked Sentence B
Unlabeled Sentence A and B Pair
Pre-training
MNLI
NER
SQuAD
Start/End Span
BERT
Question
Paragraph
Question Answer Pair
Fine-Tuning

此处讨论的是GPT-1，这是由OpenAI在2018年推出的模型，其发布时间稍早于BERT几个月。尽管GPT-1与当前广泛使用的ChatGPT在技术上采用了相似的原理，但两者在实际应用方面存在较大差异。

GPT同样基于Transformer架构，但采用**单向生成**方式，专注于预测下一个词，适用于自然语言生成任务，如文本续写、对话生成等。

总结

从训练方式来看，BERT的训练是一次看完整个文本，然后去处理填空以及判断上下文的问题，在这样的训练方式下，BERT主要被赋予了**理解文本的能力，而非创造文本的能力**。这也就是为什么虽然BERT可以大杀四方，但这些任务多数还是属于NLU的问题。

GPT的训练过程是不断地预测下一个字应该是什么，类似于文字接龙游戏，因此，在理解类任务上，尽管GPT的表现通常不如BERT，但一旦进人生成类任务的领域，GPT则显示出巨大的优势。随着生成模型突飞猛进地发展，人们逐渐认识到，通过少样本学习（Few-shot Learing）、思维链（Chain-of-Thought，CoT）等与以往截然不同的新技术和新观念，生成模型在NLU问题上也能有出色的表现，且方法更为巧妙和直观。

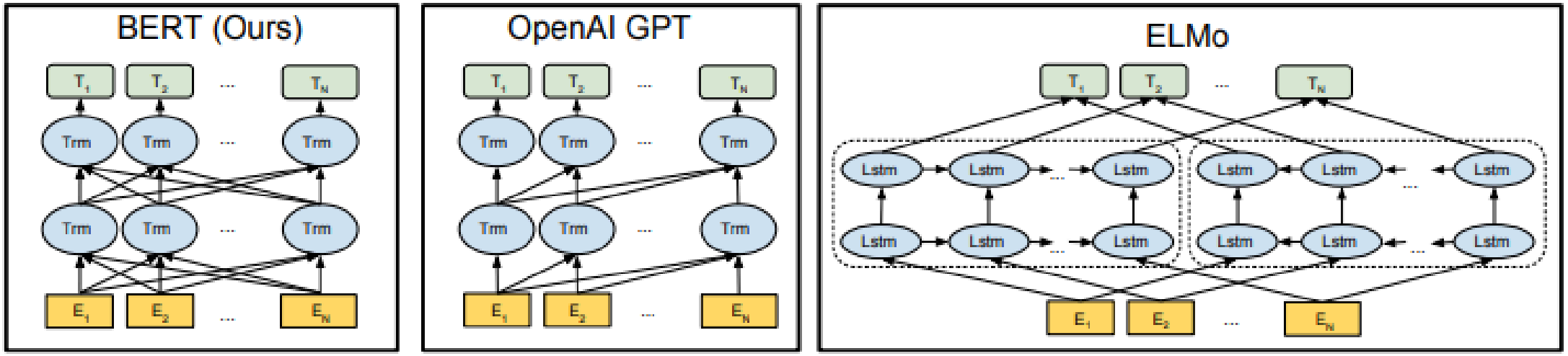


Figure 3: Differences in pre-training model architectures. BERT uses a bidirectional Transformer. OpenAI GPT uses a left-to-right Transformer. ELMo uses the concatenation of independently trained left-to-right and right-to-left LSTMs to generate features for downstream tasks. Among the three, only BERT representations are jointly conditioned on both left and right context in all layers. In addition to the architecture differences, BERT and OpenAI GPT are fine-tuning approaches, while ELMo is a feature-based approach.

GPT系列模型演进

- GPT-1：微调解决下游任务、引人特殊字符、多任务目标函数
- GPT-2：零样本学习、自然语言提示词
- GPT-3：少样本学习、示例的重要性
- InstructGPT：监督微调、奖励模型、强化学习

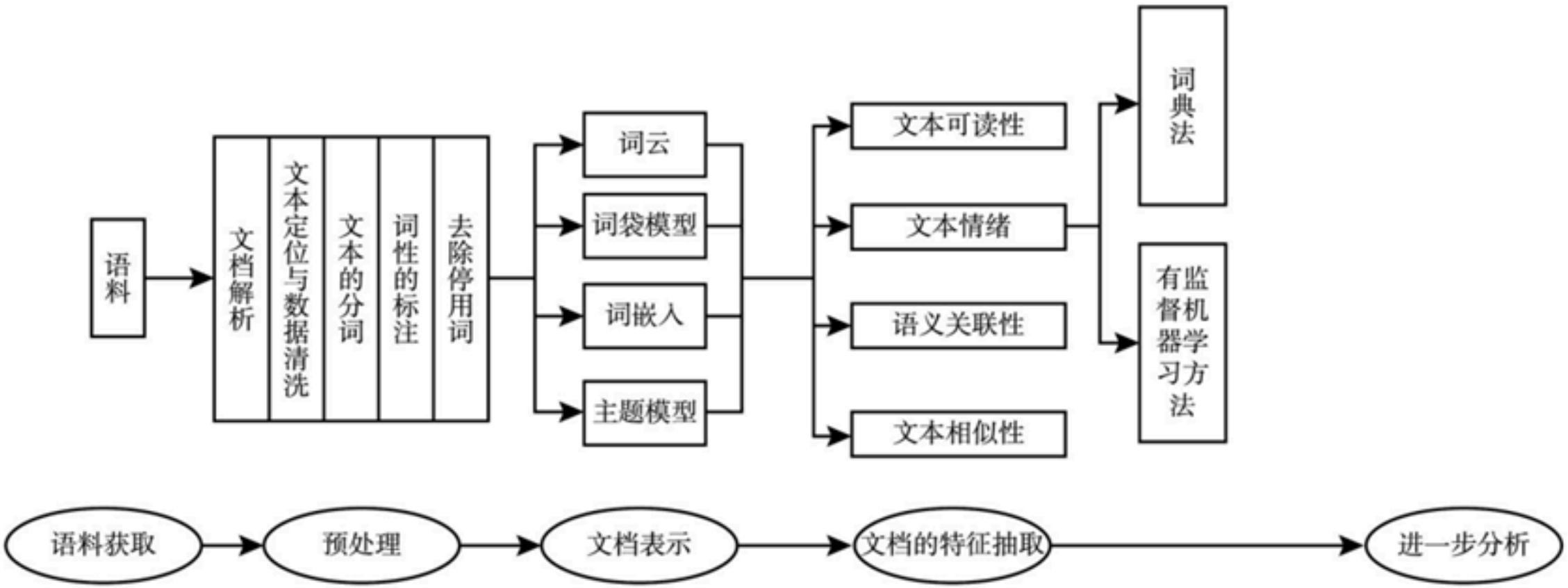
涌现现象

在大语言模型中主要体现在模型的性能随着规模的增加呈现出非线性的变化趋势。在模型规模较小时，增加模型规模对性能的提升效果不明显；但当模型规模超过某个临界值后，性能会突然显著提升，并随后呈现线性增长的趋势。这种现象表明，模型的性能并非简单地随着规模的增加而线性提升，而是存在一个质变的过程。在这个过程中，模型的参数之间仿佛形成了某种协同作用，使得模型具备了更强的处理能力和智能特征。

02

文本分析全流程

中文文本大数据的分析全流程



1. 语料获取

- 手工收集：手工收集需要人工逐一查找和整理文本，耗时费力且效率较低，但对于一些特定的、难以通过自动方式获取的数据，仍然是一种重要的获取途径。
- 网络抓取：通过编程语言编写网络数据采集程序，可以自动化地从互联网上抓取大量的文本大数据。这种方法不仅高效，而且能够获取实时的、大规模的数据，为研究和分析提供了更多的可能性。

经济领域的的文本信息来源可归纳为四类：

1. 上市公司披露的文本信息——财务报告、电话会议、业绩说明会、招股说明书等。
2. 财经媒体报道——华尔街日报、纽约时报、金融时报、上海证券报、中国证券报等。
3. 社交网络文本——Seeking Alpha、Twitter、新浪微博、东方财富股吧等。
4. 其他文本——搜索指数、分析师报告、问询函、LinkedIn、Glassdoor、专利文本、政府工作报告等。

2. 文本预处理

文本数据往往包含大量的噪声和冗余信息，因此在语料获取后需要对文本进行预处理，以提高后续分析的准确性和效率。

2.1 文档解析

- 指将原始的文本数据转换成计算机可处理的结构化形式，以便进行后续的文本分析和挖掘。在经济领域大量文档常以PDF格式存储并披露，这些电子文档在计算机领域统称为富格式文档。
- 在进行PDF文档解析的过程中需要注意两个方面：
  i. PDF的生成并不是一个可逆的过程，从中恢复的文本和格式也并不总是完全准确的。
  ii. 解析PDF文档时需要选择精准的文档结构解析工具，才能确保其准确性和可靠性以进行后续的文本分析：深度学习方法、PDFlux、PDFTables.....

2.2 文本定位与数据清洗

- 文本定位，是指从原始的文本数据中定位出特定的文本信息。在文本分析中，我们往往需要从大量的文本数据中提取出我们感兴趣的内容，如关键词、实体、事件等。例如，使用正则表达式定位MD&A的开头和结尾找到对应的MD&A部分。
- 数据清洗，是指对从文本中提取的数据进行整理和修正，以去除噪音、冗余和错误的信息。主要包括各种特殊字符、标点符号、超文本标记语言（HTML）、直译式脚本语言（JavaScript）代码以及图片等内容。

2.3 文本分词

- 是将连续文本拆分为具有最小完整语义的词语的过程，它为后续的文本处理和特征提取提供了基础。
  - 英文文本的分词相对简单，一般按照空格或标点符号进行分割，然后进行词性还原（Lemmatization）和词干提取（Stemming）。
  - 中文文本的分词，由于词语之间没有明显的分隔符，因此需要利用分词工具或算法进行分词，中文分词工具常用的有jieba、HanLP等。
- 中文文本分词存在三个难点：切分颗粒度、歧义词的识别和新词的识别。

2.4 词性标注（Part-of-speech Tagging）

- 是指将给定一段文本中的每个词语确定其基本的语法属性并赋予标签的过程。
- 中英文在词性标注方面存在显著的差异：
  - 英文单词通常通过词尾的变化来表示不同的词性，例如动词变为名词常常在词尾添加“-ing”或“-ment”等后缀。
  - 中文词性的识别主要依赖于句子的语法结构和语义含义，即需要从上下文中推断词语的角色。

2.5 去停用词

- 停用词（Stop Words），是在文本中频繁出现但没有实际意义的词语。通过去除停用词，从而减少数据维度并提高处理效率。停用词通常是一些高频词，如冠词、介词、代词、连词等。
  - 在中文中，停用词包括标点符号和特殊符号，以及“的” “和” “了” “是” “在” “我” “你”等。
  - 在英文中，停用词包括“the” “and” “of” “is” “it” “you”等。
- 但在某些特定任务中，有时需要保留某些常见的停用词，因为它们可能携带了一些重要的语义信息。例如文本情感分析中需要保留标点符号和语气词。

3. 文档表示

3.1 词云（Word Cloud）

3.2 词袋模型（Bag of Words，BOW）

顾名思义就是假设存在一个包含了所有词语的“袋子”，对于给定的句子就可以用“袋子”中词语出现的频次来表示，也即将句子视作是词语的集合从而将文本向量化。主要包括独热表示法（One-hot Representation）和词频-逆文档频率法（Term Frequency-inverse Document Frequency，TF-IDF）

词袋模型存在以下问题：

1. 仅仅是把文本看做是词语的无序集合，词语与词语之间是孤立的，忽略了词语在文本中的顺序，只考虑词语的频次信息。由此生成的文本向量丢失了词汇的上下文语义信息。
2. 在词汇量较大的情况下会使得文本向量维度过度扩充，并存在大量的0值，此时的向量是高维且稀疏的，这会导致计算和存储资源的浪费。

3.3 词嵌入（Word Embedding）

- 是一种将高维稀疏的词语空间嵌入到相对低维的连续向量空间的技术，实现将每个词语转化为一个实数向量。这样的表示方式能够捕捉词汇之间的语义关联和相似性，并以词语在向量空间中的位置来表示它们之间的语义含义，因此在向量空间中距离越近的词其语意就越相似。
- 常用的词嵌入算法包括Word2vec、GloVe等，其中Word2vec是一种浅层的神经网络模型，广泛应用于经济领域文本分析中。

3.4 主题模型（Topic Model）

- 是一种提取一系列文档中潜在主题信息的统计模型，可以深入挖掘文档主题信息。具体来讲是根据每个文档中的词语统计信息，从而推断该文档中所包含的主题分布和每个主题的词语分布。能完成词到主题的推理，因为不同主题下单词的出现频率不同。

LDA（Latent Dirichlet Allocation，潜在狄利克雷分布）

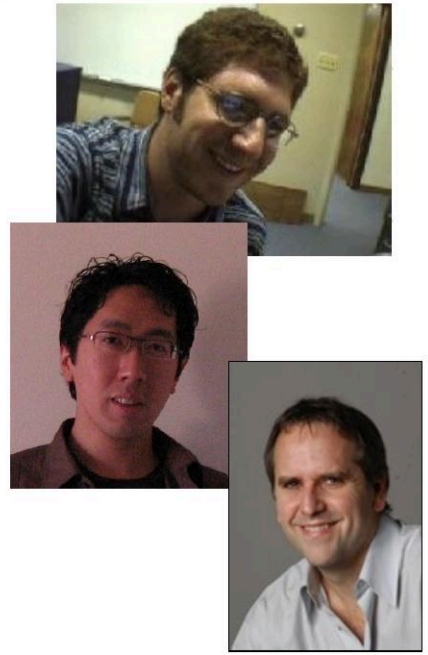
- **文档-主题分布**：每篇文档的主题比例服从狄利克雷分布。
- **主题-词分布**：每个主题的词汇概率也服从狄利克雷分布。

Latent Dirichlet Allocation

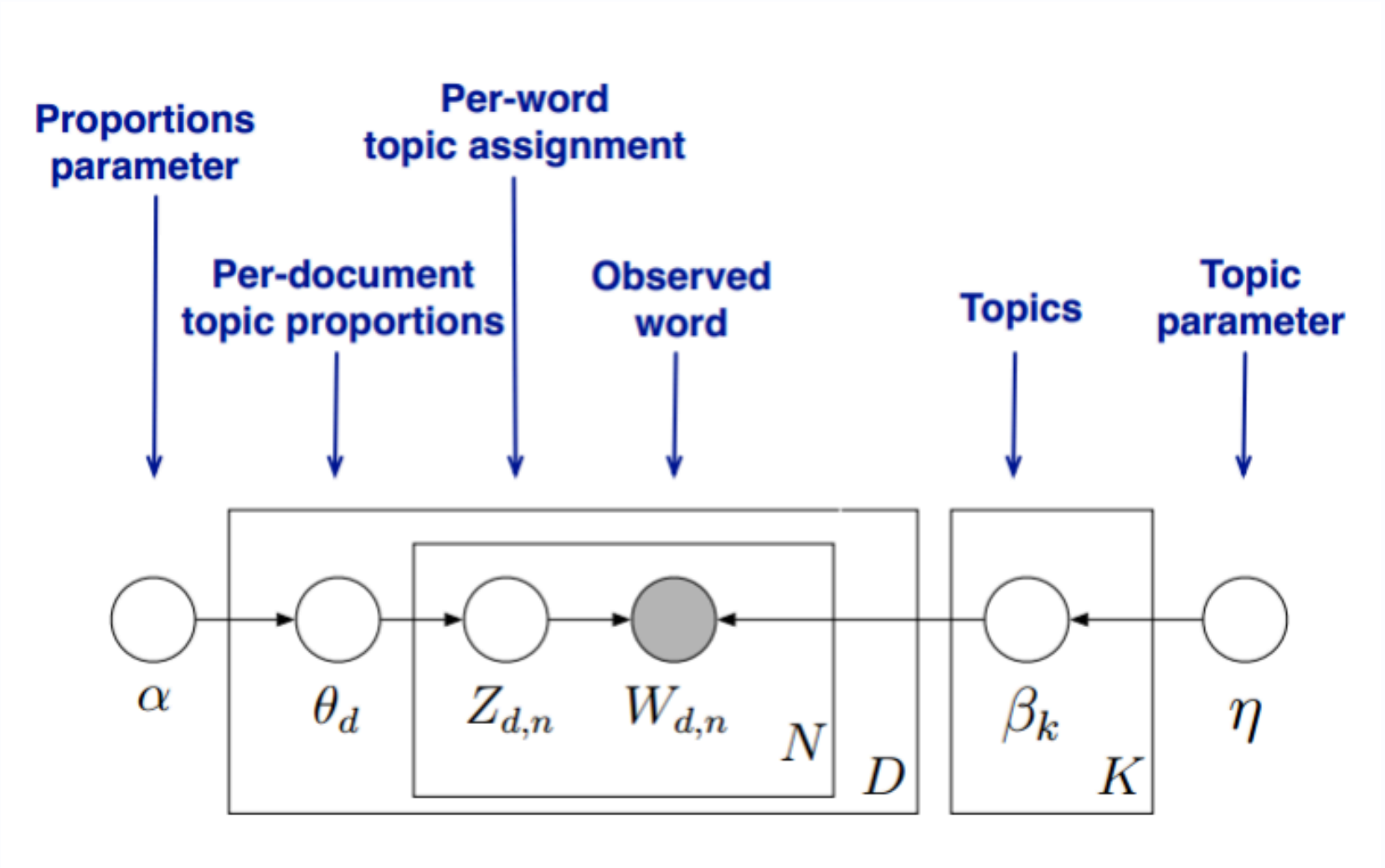
David M. Blei
Computer Science Division
University of California
Berkeley, CA 94720, USA

Andrew Y. Ng
Computer Science Department
Stanford University
Stanford, CA 94305, USA

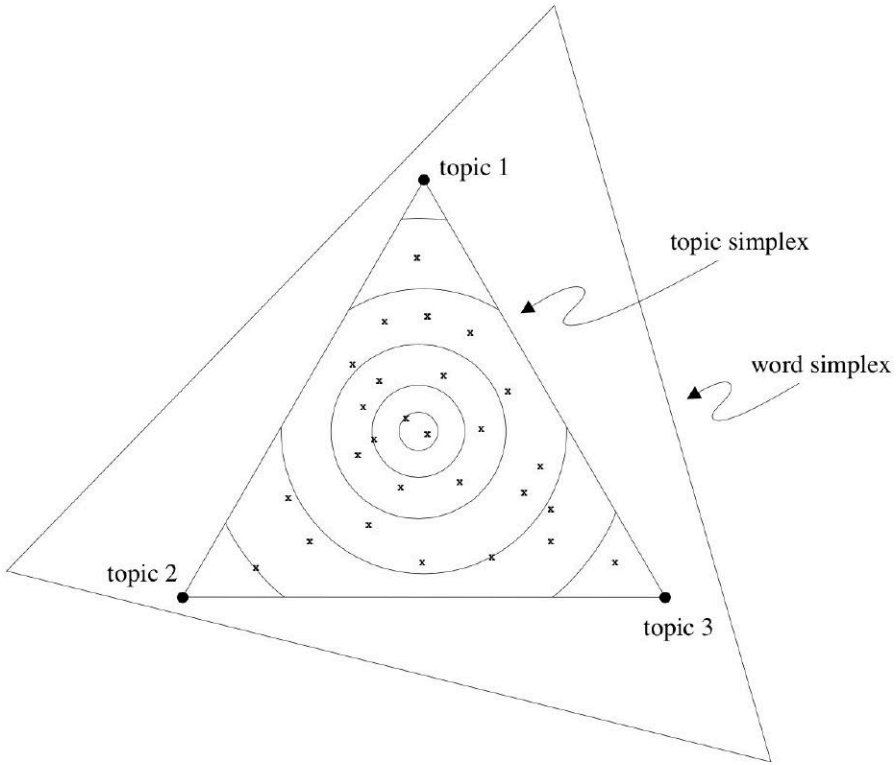
Michael I. Jordan
Computer Science Division and Department of Statistics
University of California
Berkeley, CA 94720, USA



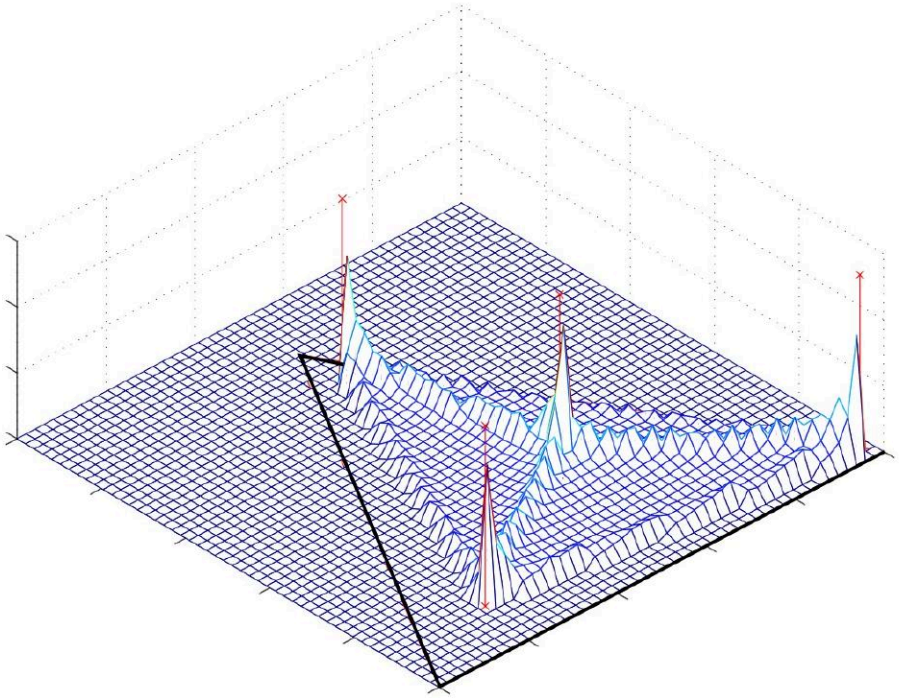
):://dl.acm.org/citation.cfm?id=944937


Proportions parameter
Per-word topic assignment
Per-document topic proportions
Observed word
Topics
Topic parameter
α
θ_d
$Z_{d,n}$
$W_{d,n}$
N
D
β_k
K
η

Dirichlet Distribution



Dirichlet Distribution



4. 文档的特征抽取

4.1 文本可读性

- 通过文本中一些语言或者词语的特征来计算衡量阅读文本的认知负载。反映了读者从文本中获取信息的难易程度，它往往会影响读者对文本内容的判断。通过对上市公司年度财务报告的可读性进行研究，可探究上市公司所传达的信息质量和市场对信息的反应。例如年报可读性不同的公司在企业利润、融资约束和融资方式等方面的区别，公司管理者策略性操控年报可读性的行为，年报可读性对分析师盈余预测的影响。
- 衡量方法：
  - 一是计算句子的长度以及复杂词语的出现次数。例如迷雾指数（Fog Index），最早由Li（2008）应用到了文本分析中，迷雾指数越小，年报的可读性越强。
  - 二是采用年报中的字数和年报电子文档的大小来衡量年报的可读性。
  - 三是基于平实英语（Plain English）的可读性指标，即Bog Index。Bog Index除了考虑句子长度和词语难度外，还包括了语态、动词、俚语、专业术语、抽象词汇、冗余词汇和过度细节等方面。
  - 四是基于机器学习的方法，通过人工对句子的可读性进行标注，并且基于这些标注得到的数据训练模型，然后将其建模成一个回归任务，通过模型计算出文本的可读性。

4.2 文本情绪

从文本数据中获取其中所包含的情感倾向。通常将其视为一个分类任务，针对句子中所包含的情绪，将其分类为积极、消极和中性，计算文本的情绪值或语气语调。

词典法

$$积极语调(Pos) = 积极、正面词的个数/文本总词数$$

$$消极语调(Neg) = 消极、负面词的个数/文本总词数$$

$$净正面语调 = (Pos - Neg)/(Pos + Neg)$$

- 核心在于情绪词典的构建与维护：
  - 英文词典：Henry词典(Henry Word Lists，2008)、LM词典(Loughran and McDonald Word Lists，2011)、哈佛大学通用调查词典(Harvard General Word Lists，GI)、文辞乐观与悲观词典(Diction Optimism and Pessimism Word Lists)等。
  - 中文词典：大连理工大学情感词汇本体库、中国知网词库和清华大学褒贬意词典等通用型情绪词典。
- 词典法的局限性：
  - 静态的词典无法利用词语的上下文语境信息。
  - 专业词典的构建较为依赖专家经验，词典的通用性差。

4.3 语义关联性

根据某一类词语去识别文本语义特征的过程。具体而言，首先依照某一类关键词构建词表，然后计算词表中词语在文档中的词频，进而识别出文本中与关键词语义相关的语义特征。运用词嵌入技术，根据空间中词向量之间的距离（即语法和语义的相似性）来处理词语语义关联性问题。

4.4 文本相似性

用于衡量文本数据间包括内容、主题等相似性的度量，文本相似性分析在经济领域发挥着广泛的作用，被用于抄袭检测、评估企业产品或服务的差异和文本聚类等方面。

- 文本相似性分析的步骤：
  - 一是将文本进行向量化。两种方法：词袋模型、词嵌入。
  - 二是计算文本向量之间的相似度。目前多运用余弦相似度来衡量：

$$\text{cosine similarity}(d_1, d_2) = \frac{a \times b}{\|a\| \times \|b\|} = \frac{\sum_1^n w_{ai} \times w_{bi}}{\sqrt{\sum_1^n w_{ai}^2} \times \sqrt{\sum_1^n w_{bi}^2}}$$

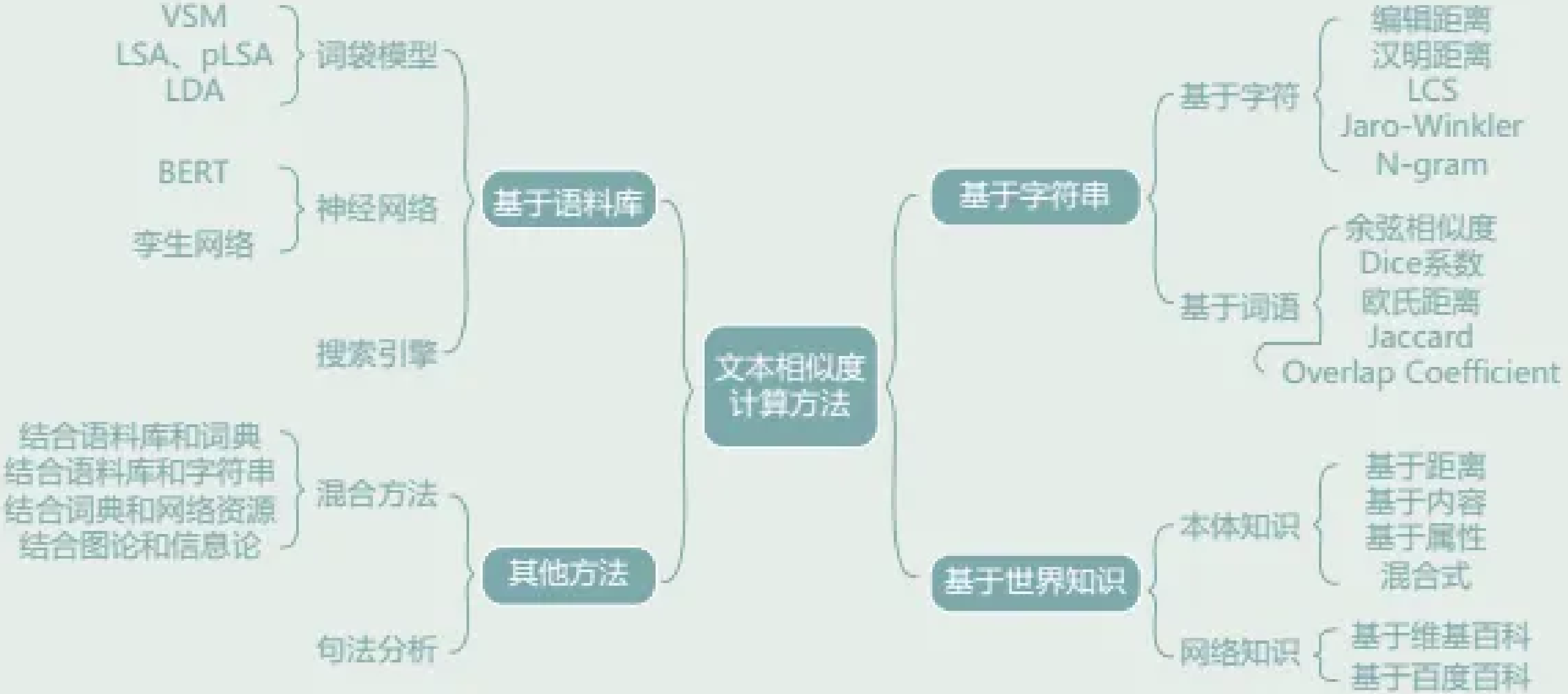


图2　文本相似度计算方法分类

03

网络爬虫实战

目前网络数据采集一般有两种方法：一种是通过应用程序接口(Application Programming Interface，**API**)来获取数据；另一种是**网络爬虫**(Web Crawler)。

Python中使用的API通常分为两种：数据API和库的API。

- 库API是开发者和使用者之间的一个桥梁。
- 数据API一般是前端和后端的桥梁。后端设计一个可展示数据的接口(API)给前端人员调用以获取数据，这种API一般以URL形式存在。同时有些API不仅可以在网站内部使用，而且也向外界提供。可以通过访问这些URL来获取数据，就不需要去抓取解析网页的HTML代码，直接接入API获取用json格式存储的数据。

API技术通常受限于平台开发者，网站会对每天的接口调用数量做限制以减小网站的负荷。在实际的网络数据采集过程中，大部分的网站都没有提供API，因此一般情况下，网络爬虫是更加通用的方法。

网络爬虫，是一种按照一定的规则，自动地爬取万维网信息的程序或者脚本。它可以根据网页的链接地址自动获取网页内容。

我们在浏览器里看到的网页其实是由以下三层信息构成的一个共同体：

构成	语言	解释
结构层	HTML	页面的元素和内容
表现层	CSS	网页元素的页面样式
行为层	JavaScript	网页模型的定义与页面交互

HTTP协议(超文本传输协议HyperText Transfer Protocol)：简单来说就是客户端和服务端进行数据传输的一种规则。例如，浏览器通过DNS查询，然后根据URL把数据取回来，这个过程用的就是HTTP协议。HTTP请求的过程如下：浏览器根据域名查询IP地址；浏览器与WEB服务器建立TCP连接；浏览器给WEB服务器发送HTTP请求；服务器端响应请求，浏览器得到HTML代码；浏览器处理HTML代码，并请求HTML代码中的资源；关闭TCP连接；浏览器对页面进行渲染呈现给用户。

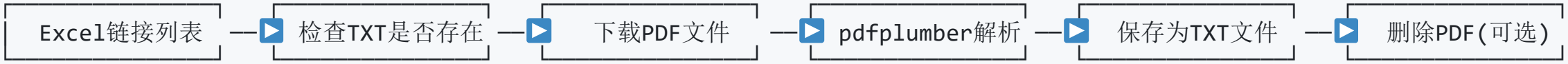
Robots协议：其目的是保护网站数据和敏感信息，确保用户个人信息和因素不被侵犯，是国际互联网通行的道德规范。对搜索引擎可抓取的网站内容范围作了约定，包括网站是否希望被搜索引擎抓取。网络爬虫以此协议内容判断内容是否允许被抓取。

通过**开发者工具**，我们可以轻松查看网页的各种元素，包括页面结构和网络请求等重要信息。`F12`

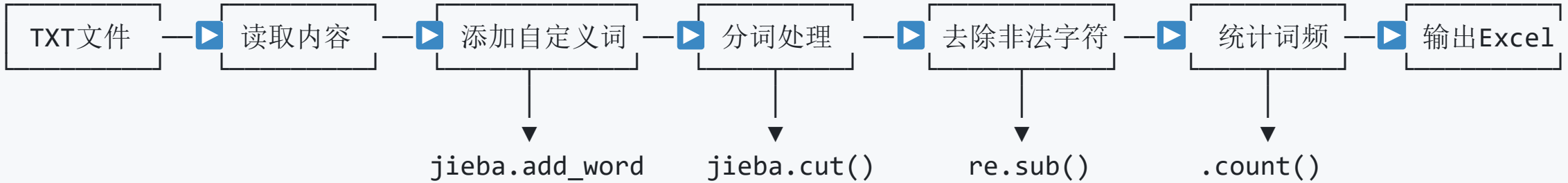
爬虫三部曲：发送请求、解析数据、存储数据

年报数据获取与处理全流程


```python
# 完整爬取流程示意
def crawl_annual_reports(year):
    # 1. 确定搜索日期范围（年报在次年1-4月披露）
    disclosure_date = f"{year+1}-01-01~{year+1}-04-30"
    # 2. 分页请求巨潮资讯API
    all_reports = []
    for page in range(1, total_pages + 1):
        response = requests.post(api_url, data={...})
        all_reports.extend(response.json()["announcements"])
    # 3. 过滤和清洗
    reports = filter_reports(all_reports, exclude_keywords=['摘要','英文','已取消'])
    # 4. 保存到Excel
    save_to_excel(reports, f"年报链接_{year}.xlsx")
```




```
# 文件命名规范：格式：{股票代码}_{公司简称}_{年份}.pdf，示例：000001_平安银行_2024.pdf
```




```python
# 添加并过滤停用词
STOPWORDS = {'的', '了', '是', '在', '和', '与', '或', '及', '等', '我们', '公司', '本', '年度', '报告',
'根据', '进行', '可以', '需要', '已经', '没有', '这些', '那些', '一个', '这个', '那个', '各', '通过', '其中'}
def filter_stopwords(words):
    return [w for w in words if w not in STOPWORDS]
```

```python
# 计算词频(Term Frequency)
def calculate_tf(keyword_counts, total_words):
    return [count / total_words if total_words > 0 else 0
            for count in keyword_counts]
```

目录结构

```
项目目录/
├── 年报链接_2023.xlsx              # 爬取的链接
├── 年报文件/
│   ├── 2023/
│   │   ├── pdf年报/                # 原始PDF（处理后可删除）
│   │   │   ├── 000001_平安银行_2023.pdf
│   │   │   └── ...
│   │   └── txt年报/                # 提取的文本
│   │       ├── 000001_平安银行_2023.txt
│   │       └── ...
│   └── 2024/
│       ├── pdf年报/
│       └── txt年报/
├── 词频分析结果.xls                 # 最终统计结果
└── logs/                           # 运行日志
    └── processing_20250101.log
```

扩展分析

1.**情感分析**：基于词典或BERT模型分析年报文本情绪；2.**主题建模**：使用LDA提取年报主题分布；
3.**对比分析**：同行业公司关键词频次对比；4.**时间序列**：多年度关键词频次变化趋势；
5.**可视化**：生成词云、热力图、趋势图。

延申阅读

玩转Python之“手把手”教你爬数据（一）

玩转Python之“手把手”教你爬数据（二）

Python爬虫实战：从抓取年报并分析数据开始

用Python处理中文文本分析数据——以中文年报为例

干货 | Python爬虫与金融文本分析（基础篇）

干货 | Python爬虫与金融文本分析（进阶篇）

干货 | Python爬虫与金融文本分析（实战篇）

次回予告：最强中文NLP预训练模型ERNIE

ERNIE（文心，全称 Enhanced Representation through kNowledge Integration）是百度在2019年提出的预训练语言模型，在BERT的基础上引入了**知识增强**（Knowledge Integration），特别适用于中文NLP任务。通过整合外部知识，使预训练语言模型在多种NLP任务（如命名实体识别、阅读理解、文本分类等）上表现更好。

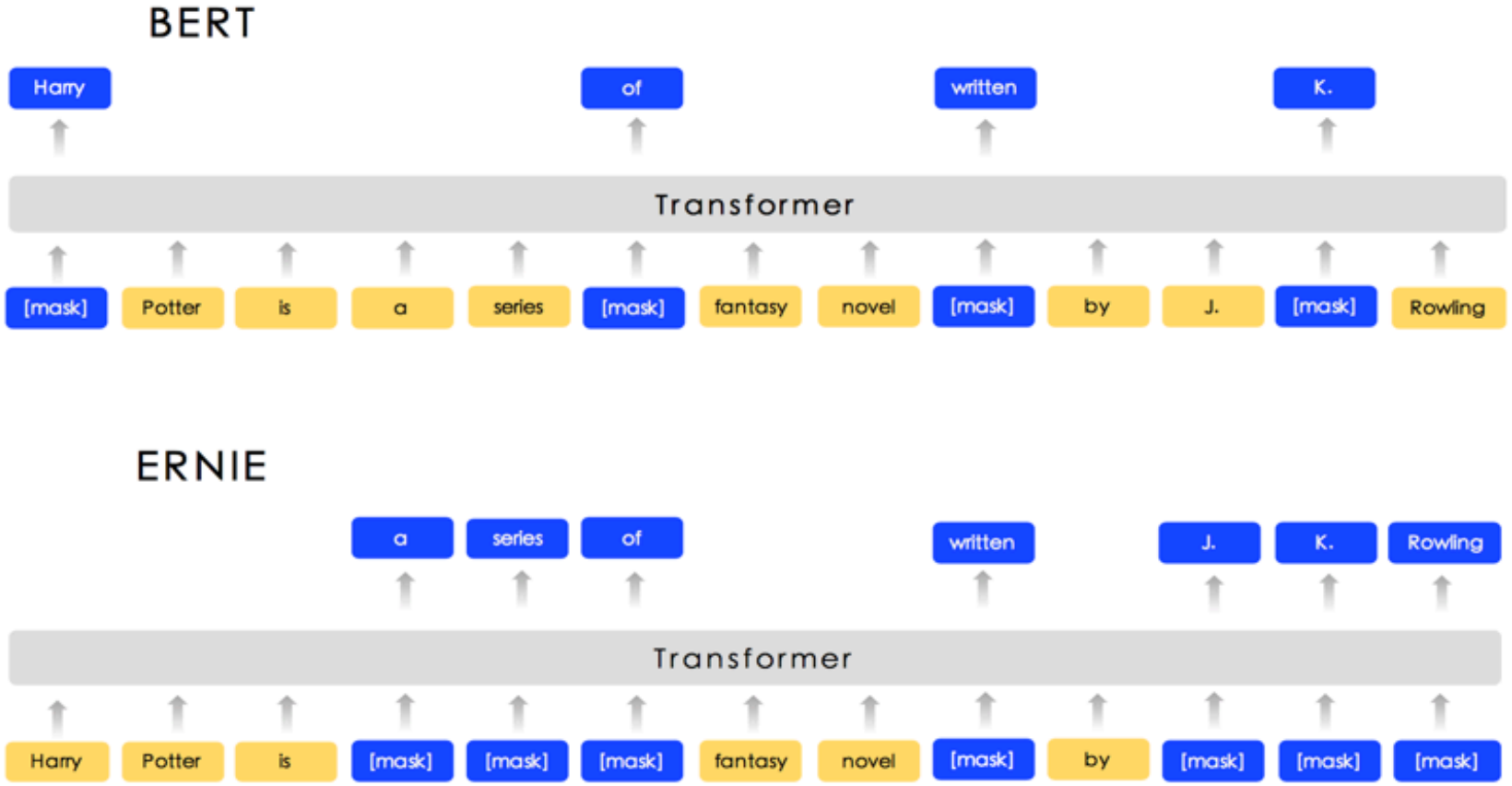
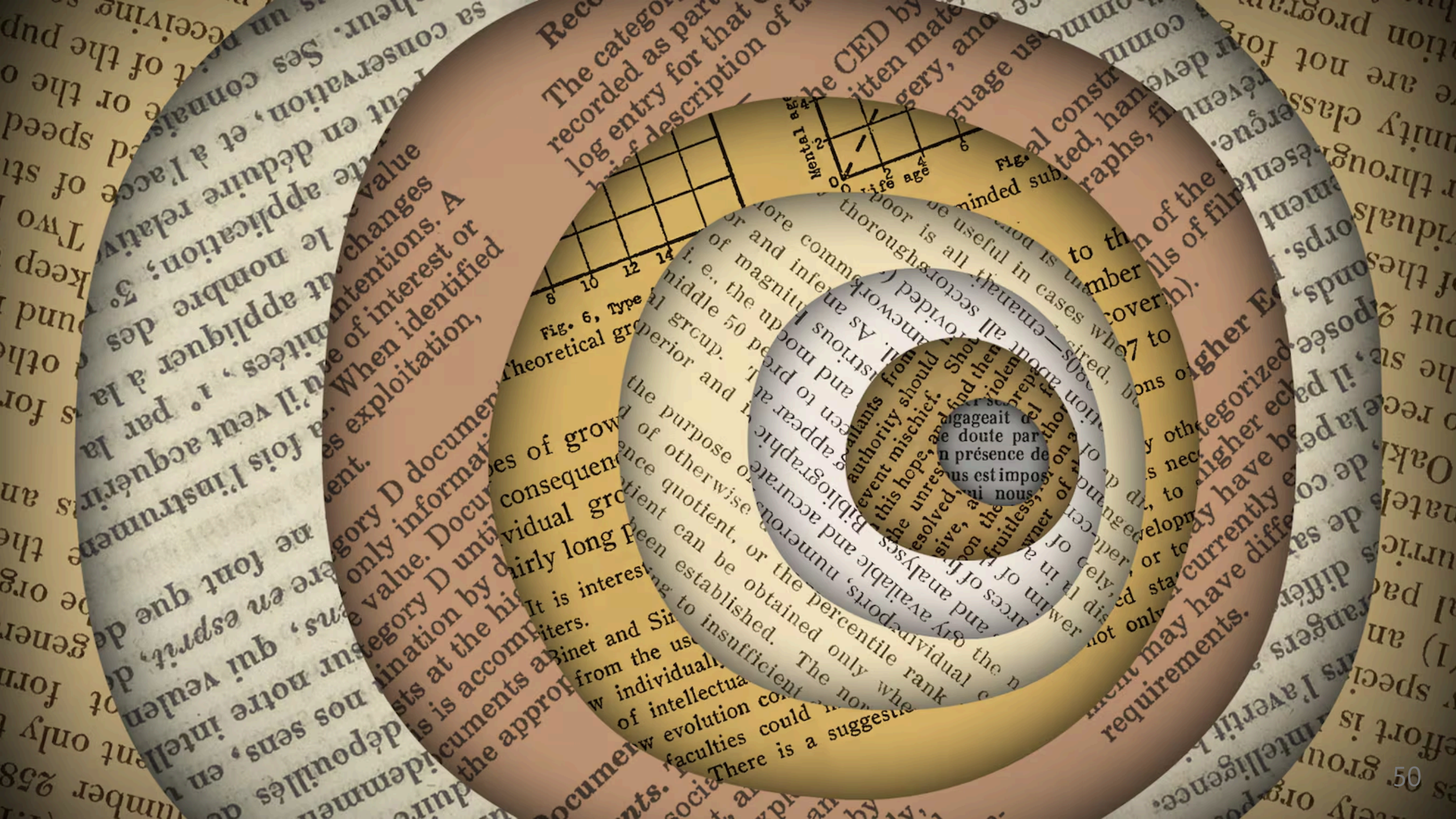


Figure 1: The different masking strategy between BERT and ERNIE


Mental age
Life age
Fig. 6, Type
theoretical gr