**keyphrase extraction调研**

**背景**

**定义**

将文本中具有代表性的词或者短语抽取出来，用来表示文本的关键内容。

**业界问题定义**

**问题建模角度1**

1. two-step ranking method；

pipeline分为2个步骤：

——生成候选keyphase；

——通过监督或非监督方法来决定保留哪些keyphase；

1. sequence labeling；

类似于实体识别方法；

1. generation using neural methods；

将keyphrase通过连接符拼接后作为输出文本；

**问题建模角度2**

* **有监督**
* 将关键词抽取算法看作是二分类问题，判断文档中的词或者短语是或者不是关键词。既然是分类问题，就需要提供已经标注好的训练语料，利用训练语料训练关键词提取模型，根据模型对需要抽取关键词的文档进行关键词抽取。
* 主要分类：
* 传统方法，基于特征工程
* 深度学习，发展趋势如何有效引入预训练模型；
* 抽取类
* 生成类
* **无监督**
* 不需要人工标注的语料，利用某些方法发现文本中比较重要的词作为关键词，进行关键词抽取。
* 主要分为3类：
* 基于统计类方法，发展趋势是增加更多特征；
* 基于图的方法，发展趋势是建词图的时候引入外部资源（doc或者主题）、权重计算做优化（引入位置信息等）；
* 基于embed类的方法，发展趋势引入预训练模型，更准确的表征语义信息；

**评测指标**

F1@k：k是事先定义的keyphrase个数，一般取3，5，10

F1@O：O等于真实keyphrase的个数

F1@M: M等于预测的keyphrase的个数

**业界经典方法梳理**

**无监督学习-统计类**

**Tfidf**

**Yake, 2018，Yet Another Keyword Extractor**

Paper：YAKE! Collection-Independent Automatic Keyword Extractor

Code: github.com/LIAAD/yake

无监督的关键词提取算法，特征提取主要考虑5个因素（需要去掉停用词），基于此5个因素计算得到每个词的重要分数：

* 大写term：针对英文，中文没用
* 词的位置：越靠近开头的句子，其重要度比后面的句子越高；
* 词频：一个词在文本中出现的频率越大，相对会更重要；但需要考虑长文本对应的词频会偏高的问题，需要进行归一化操作；
* 上下文关系：一个词与越多不相同的词共现，该词的重要性越低；
* 词在句子中出现的频率：一个词在越多句子中出现，相对越重要；

**无监督学习-词图类**

**PageRank**

通过计算网页链接的数量和质量来粗略衡量网页的重要性；根据网页间的链接关系来构建网络；构建的网络中的边是有向无权边；

主要思路：

（1）链接数量：如果一个网页被越多的其它网页链接，说明这个网页越重要，pagerank也相对较高；

（2）链接质量：如果一个网页被一个越高权值的网页链接，也说明这个网页也很重要，因此其pagerank也会相应的提高；

文本, 信件

描述已自动生成

**TextRank，2004**

TextRank: Bringing Order into Texts

思路：通过词之间的相邻关系构建网络，然后用PageRank迭代计算每个节点的rank值，排序rank值即可得到关键词。根据词间的共现关系来构建网络；构建的网络中的边是无向有权边；

图片包含 文本

描述已自动生成

**SingleRank，2008，coling**

CollabRank: Towards a Collaborative Approach to Single-Document Keyphrase Extraction

第一：不同于PageRank，每个边都有相同的分值，SingleRank会根据窗口大小词之间的距离计算不同的边权重

第二：与TextRank不同的是，SingleRank保留所有的unigrams词，然后类似TextRank方法，滑动窗口方式计算更高的n-grams词，背后的原理是，两个分值较低的unigram，有可能产生较高分值的bi-gram。

**ExpandRank，2008**

Single Document Keyphrase Extraction Using Neighborhood Knowledge

不同的是除了自己所在的文档外，还通过余弦相似度引入其他文档用于构建词图；

具体步骤如下：

1. 通过余弦相似度选出候选的k篇文章，k=0时退化为SingleRank；
2. 对K篇文章构建词图（利用词性标注，只保留名词和形容词），利用公式计算词图的词之间的边的权重，第一部分计算目标文章和扩展文章的相似度，第二部分是表示在文章中词间的共现次数：

文本, 信件

描述已自动生成

**TopicRank，2013**

[TopicRank: Graph-Based Topic Ranking for Keyphrase Extraction](https://www.aclweb.org/anthology/I13-1062.pdf)

一种引入主题信息的无监督关键词提取算法；TopicRank把主题当做相似关键短语的簇，这些topics会根据在文档的重要性进行排序，然后选取top 个最相关的topics，每个topic选择一个最重要的关键短语来代表文档的核心关键词。

TopicRank算法的步骤如下：

主题识别：主要抽取名词短语来表征文档的主题，短语中有超过25%重合的单词就考虑为相似短语，用 Hierarchical Agglomerative Clustering (HAC) algorithm进行了聚类相似的短语。

图构建：图中的节点是topics，边的权重，根据两个之间的语义关系进行分配，而语义关系的强弱根据两个主题的关键短语之间的距离公式：

文本

描述已自动生成

关键短语选择：一旦topic进行排序后，选择top K个topics，每个topic选择一个最重要的关键短语作为输出，所有topics总共产生top K个关键短语。有三个策略选择一个topic最适合的关键短语：第一：选择关键短语中最开始出现在文档的那个关键短语；第二：选择频率最高的那个关键短语；第三：选择聚焦的群簇中心的那个关键短语

**PositionRank，2017，acl**

An Unsupervised Approach to Keyphrase Extraction from Scholarly Documents

https://github.com/ymym3412/position-rank

思路：提供了一种学术文档关键短语提取的无监督方法；基于图结构的算法；

它同时将单词的位置及其频率包含在文档中，以计算每个候选词的有偏PageRank分数。

图表

中度可信度描述已自动生成

算法主要2个部分组成：

* 图的构建：类似TextRank，根据POS选择关键词构建图的节点，节点的边根据窗口size共现次数来计算两个词的边的权重分值。
* Position-Biased PageRank： 根据每个词的位置的导数计算权重，若一个词出现在文档多个位置，则分值相加。核心思想是：越在一个文档靠前的位置，权重越大，同时频率出现越高，权重也越大。

**MultipartiteRank， 2018， naacl**

Unsupervised keyphrase extraction with multipartite graphs.

在TopicRank的基础上，在多部分图结构中对主题信息进行编码，并将关键词候选词和主题表示在一个图中，并利用它们之间的相互加强关系来提取关键词。

图示

描述已自动生成

**无监督学习-embed类**

**总结**

（1）论文结果MDERank其效果优于现有词图类算法和无监督学习的emd算法，有些数据上没有SIFRank好；

**embedRank， 2018**

Unsupervised keyphrase extraction using sentence embeddings

**思路**:先利用POS tags抽取候选短语，然后计算候选短语的embedding和文章embedding的cosine similarity，利用相似度将候选短语排序，得到关键的短语。

**步骤：**

1. 通过词性组合抽取候选短语：0或者更多的形容词跟着至少一个名词
2. 计算候选短语的embbeding和doc的embbeding， 计算doc的embbeding时会进行降噪处理，只保留形容词+名词，doc计算时是按照保留的词生成doc的向量，文中了对比两种doc的向量的计算方法（使用预训练的sen2vec和doc2vec）
3. 计算候选短语和doc的相似度得分并进行排序

说明：针对上述方法提取出来的top短语其实很多是表达相似的意思，为了增加短语的多样性，文章提出了embedrank++,采用信息检索中的MMR(最大边界相关性)增加多样性

**Key2vec ，2018，naacl**

**K**ey2**V**ec: Automatic Ranked Keyphrase Extraction from Scientific Articles using Phrase Embeddings

方法：首先基于规则得到候选phrase，然后使用fasttext得到 phrase/document embedding，最后基于PageRank来排序；同时考虑了phrase和document的相似度、phrase间的语义相似度；

**Sifrank，2020**

SIFRank: A New Baseline for Unsupervised Keyphrase Extraction Based on Pre-Trained Language Model

<https://github.com/sunyilgdx/SIFRank_zh>

思路：SIFRank通过引入预训练模型ELMo进行排序，首先词向量ELMo比TFIDF，TextRank相比可以保留更多语义信息，ELMo是动态的，可以改善一词多义问题。句向量SIF根据词频对词向量进行平滑反频率加权，可以更好的捕捉句子的中心主旨。

Sifrank利用POS tags抽取NP作为keyphrase candidates，然后利用ELMO 分别获得句子和候选短语的语义向量，最终通过计算两者的cos相似度进行排序；

**Sifrank++**：对于长文本来说，先出现的keyphrase可能更重要，因此加入了位置信息，每个词第一次出现的位置表示定义权重。

图示

描述已自动生成

**AttentionRank,2021,emnlp**

AttentionRank:Unsupervised Keyphrase Extraction using Self and Cross Attentions

git：https://github.com/hd10-iupui/AttentionRank

思路：针对文档级别的keyphrase提取；使用预训练的语言模型来计算候选者在句子上下文中的自注意力以及候选者和文档中句子之间的交叉注意力，以评估候选者的局部和全局重要性。

图示

描述已自动生成

**MDERank，2022**

MDERank: A Masked Document Embedding Rank Approach for Unsupervised Keyphrase

Git：<https://github.com/LinhanZ/mderank/tree/main/MDERank>

图示

描述已自动生成

**根据论文结果，其效果优于现有词图类算法和无监督学习的emd算法，有些数据上没有SIFRank好；**

整体思想：step1:基于分词和词性结果选取名词性短语作为候选keyphrase；

Step2:为了衡量不同phrase间的重要性，依次在原始input中mask掉此phrase后得到mask input embedding，然后计算mask后和没有mask前embedding的相似度；若越不相似表示该phrase对input的语义越重要；

由于bert在预训练时，没有将keyphrase包含更多的语义信息作为目标，因此为了适配keyphrase抽取任务，提出了一个预训练模型Kpebert；

Kpebert：在bert预训练基础之上，又新增了一个loss：

文本

描述已自动生成

正样本（认为是keyphrase，pseudo label）：yake提取出的keyphrase

负样本（非keyphrase）：基于分词和词性得到的名词性短语/正样本集合

**有监督学习-深度学习类-抽取**

**总结**

（1）从论文结果来看，JointKPE效果优于BLING-KPE和CDKGen；

**SKE，Span Keyphrase Extraction ，2020**

Keyphrase Extraction with Span-based Feature Representations

亮点：实体识别会出现实体交叉的现象，该方法通过引入词汇span来避免这种问题；

图形用户界面, 文本, 应用程序, 电子邮件

描述已自动生成

思路：本质上还是属于经典的two-step methods；只是不同的是使用bert+lstm来编码phrase embedding;最后预测每个候选phrase是否是phrase的概率；

图示, 箱线图

描述已自动生成

候选phrase使用的是stanford corenlp，语法分析树，pattern： (JJ|JJR|JJS|V BG|V BN) ∗ (NN|NNS|NNP |NNP S|V BG)+

**JointKPE ，2020**

paper: [**Capturing Global Informativeness in Open Domain Keyphrase Extraction**](https://arxiv.org/abs/2004.13639)

git：https://github.com/thunlp/BERT-KPE

思路：step1:首先经过预训练模型得到char embedding，然后遍历ngram，用cnn将ngram的word embedding集成到一个phrase embedding；

Step2: 用一个feedforward将phrase embedding投影到一个量化的分数，若是一个词对应多个位置，则取最大的那个分数作为该phrase的最终分数；最后利用pairwise ranking loss，计算估计的信息分顺序；

文本

描述已自动生成

Step3:训练时采用多任务训练，跟keyphrase chunking任务一起训练，chunking任务是一个分类问题，判断ngram是不是一个phrase；推理阶段，信息分高的作为keyphrase；

徽标

中度可信度描述已自动生成

**部分评测集上效果好于BLING-KPE和CDKGen；**

**BLING-KPE，2019**

Paper：Open Domain Web Keyphrase Extractioni Beyond Language Modeling

Git：https://github.com/microsoft/OpenKP

图形用户界面, 文本, 应用程序, 聊天或短信

描述已自动生成

思路：特点引入了visual embedding和positio embedding；

模型主要分为3部分：

* 词embedding（3个concat）：elmo embedding、position embedding（sin）、visual embedding
* Convolutional transformer：使用cnn得到ngram embedding，然后通过高tranformer来建模k-gram间的交互；
* 最终输出所有可能的ngram是否是keyword的概率；

细讲下visual embedding：4类特征；（针对的是网页）

——size：word出现在文本中的长宽

——location：2维向量

——font：字体大小，是否加粗

——dom：是否是在具体的html tags中出现

**joint-layer RNN，Emnlp，2016**

Keyphrase Extraction Using Deep Recurrent Neural Networks on Twitter

思路：基于一个2层的RNN模型将关键词和关键短语提取当做一个标注分类任务，判断每个词是否是关键词或者关键短语。模型的第一层用来做关键词识别任务，第二层用来做关键短语识别任务，最后将两个任务损失函数进行权重融合，作为最终的损失函数；

图片包含 桌子, 线, 建筑, 挂

描述已自动生成

**有监督学习-深度学习类-生成**

**CopyRNN，2017， acl**

Paper：Deep Keyphrase Generation

Git：https://github.com/memray/OpenNMT-kpg-release

思路：定义为生成式问题，在输出时将每一个关键短语也看做一个词序列（sequence of words），直接用beam search生成概率最大的top K phrase；

主要特点：使用RNN Encoder-Decoder model对keyphrase中的token序列进行预测，但是这么做存在OOV问题，因为数据集中的一些词是不存在于vocabulary中，而这些词很可能是keyphrase且存在于原文中，因此作者引入copyNet对原始词表进行扩充，缓解OOV问题

Copying Mechanism；在 RNN 之外引入一个 pointer network，去计算原文本中的每一个词应该被 copy 到输出的概率。在加入 copying mechanism 之后，每一个预测词 yt 的概率由两部分来决定，一部分是通过原模型框架 decoder 部分得到的预测词yt的生成概率，另一部分是从源文本中复制这个词的概率。

存在2个问题：1. Beam search生成的关键词没有语义依赖性，导致生成数据可能存在冗余。2. Beam search数量是人为定义的，这就不能动态的根据文章调整关键词个数。

**one2seq，2020，acl**

**One Size Does Not Fit All: Generating and Evaluating Variable Number of Keyphrases**

**Git：https://github.com/memray/OpenNMT-kpg-release**

**思路：**提升keyphrase多样性，生成可变数量的keyphrase；在encoder-decoder基础上，新增了Semantic Coverage模块来在生成keyphrase的过程中保留更多源数据的语义信息，这种方式可以让生成的keyphrase具备多样性；

图示

描述已自动生成

**one2set，2021，acl**

One2Set: Generating Diverse Keyphrases as a Set

Git：https://github.com/jiacheng-ye/kg\_one2set

图示

中度可信度描述已自动生成

One2seq存在一个问题，不同的关键词拼接顺序对模型的效果有点影响；去掉顺序函数带来的诸多问题。在此范式下，我们提出了模型SetTrans，该模型利用一组固定的学习控制代码作为条件来并行生成一组关键短语。为了解决训练过程中每个预测和目标之间没有对应关系的问题，提出了一种通过二部匹配的K步目标分配机制，大大增加了多样性并减少了生成的关键短语的重复率

图示

描述已自动生成

给定一组固定的学习控制代码作为输入条件，该模型为每个代码并行生成一个关键短语或一个特殊的 ∅ 标记。 在训练过程中，提出了K-step目标分配机制来动态确定每个代码对应的目标。 主要思想是模型首先在没有任何监督的情况下自由预测K步，看看每个代码可以粗略地生成什么keyphrase，然后根据模型的猜想和目标使用二部匹配找到最优分配。 给定每个代码和目标的对应关系，然后使用单独的集合损失来纠正模型的猜想，其中一半的代码被训练来预测当前的关键短语集，而其他的则被训练来预测不存在的关键短语集

**CDKGen，2020**

Keyphrase Generation with Cross-Document Attention

Git：https://github.com/SVAIGBA/CDKGen

思路：生成任务，将keyphase拼接起来作为生成文本；

特点：采用transformer作为编码器和译码器，增加copy机制（解决oov问题），考虑融合从相似文档中抽取的潜在话题分布信息，然后进行生成

图示

描述已自动生成

**An Empirical Study on Neural Keyphrase Generation，naacl，2021**

结论：

（1）one2seq这种形式优于one2one；

图形用户界面, 应用程序

描述已自动生成

（2）one2seq这种结构中，多个keyphrase的组合顺序对模型效果的影响？

四种排序方式：

* 出现在原始文本中的顺序（正反）；
* 字符顺序（正反）；
* 长度顺序（正反）；
* 随机

结果：greedy decoding，总体来看，出现在原始文本中的顺序（正反）效果最忧；

beam search对目标的组成顺序不敏感；

（3）Copy mechanism helps present prediction while worsening absent performance.

**初步实施方案**

现状：目前基于深度学习的方案效果优于统计类、词图类；因此我们的方案将基于embed来设计；

问题建模方案：

（1）无监督：baseline定为MDERank这种基于embed的无监督方法；

（2）有监督：bert-kpe作为baseline

关键词标注方案：

（1）首先构建候选keyphrase：分词以及词性标注后，名词性短语，形容词+名词；从候选keyphrase中选择标注； spacy、stanfordcorenlp、jieba（目前看jieba效果好一些）

（1）初步看了下pke工具抽取出的keyphrase后选集，

问题：从候选集可以看出在中文数据上，分词结果是比较差的，因此需要针对业务文本进行预处理以及选择合适的分词工具；

**无监督方法强依赖于好的候选phrase；**

（2）**依赖第一步抽取的候选phrase**，基于无监督baseline出一个版本，但评估需要标注评测集，初期可以先预测出来自己标注看看大概情况；

（3）有监督的方法以及评测集均需要标注数据，所以需要安排标注人力；

# KBIR, 2022, naacl

Paper: Learning Rich Representation of Keyphrases from Text

Git: <https://github.com/jingcangcang/kbir_keybart>

本篇论文最大的贡献在于提出了一种关键词生成的预训练手段，并且证明了此预训练方法对于其他下游任务如NER, QA, RE，summarization也有提升效果。

具体的，如上图所示，



1. MLM：masked language model（masked Token Prediction） 单字符掩码任务
2. Infil：Keyphrase infilling 关键词填充任务，类似于 SpanBERT，掩码于整个关键词
3. LP：Length Prediction，关键词长度预测，将长度预测作为分类任务
4. KRC： Keyphrase Replacement Classification 随机替换同类关键词，二分类是否被替换了

# Bert-kpe, JointKPE,2021

[**Capturing Global Informativeness in Open Domain Keyphrase Extraction**](https://arxiv.org/abs/2004.13639).

论文思想：用信息分infomative scores来表示phrase的重要性，用phraseness表示phrase的关键词"样式"。

给定一个文档D, Step1, 提取所有的候选关键短语,方法是遍历所有的n-gram, 用hierarchical架构来建模ngram表示：首先用预训练的语言模型如bert对文档D的词生成词嵌入H={h1,h2,...,hn};然后遍历ngram，用CNN把ngram的word emb集成到一个[phrase embedding](https://www.zhihu.com/search?q=phrase%20embedding&search_source=Entity&hybrid_search_source=Entity&hybrid_search_extra=%7B%22sourceType%22%3A%22article%22%2C%22sourceId%22%3A%22508879335%22%7D" \t "_blank)(g\_i\_k=CNN\_k(h\_i, ..., h\_i+k-1)),

Step2, 用informative ranking network, 综合phrase出现的每个地方的局部信息分，来得到全局信息分。首先用一个feedfoward层把context-specific表示g\_i\_k投影到一个量化的分数f(c\_i\_k, D)=Feedforward(g\_i\_k)，其中c\_i\_k是长度为k的kgram。对于在文档中出现多次的kgram,其上下文分别是{c\_j\_k, ..., c\_l\_k, ..., c\_m\_k}，信息分分别是{f(c\_j\_k,D), ..., f(c\_l\_k, D), ..., f(c\_m\_k,D)}，采用max-pooling把局部信息分转换成全局信息分，即max{f(c\_j\_k,D), ..., f(c\_l\_k, D), ..., f(c\_m\_k,D)}。最后，利用pairwirse ranking loss，计算估计的信息分顺序和gt顺序的损失：L\_rank=max(0, 1 - f(positivesample, D) + f(negativesample, D))，即要求gt中，前面phrase的信息分比后面phrase的信息分大于等于1，否则就产生损失。

训练时用多任务训练，跟keyphrase chunking任务一起训练，以便更好的平衡phraseness和[informativeness](https://www.zhihu.com/search?q=informativeness&search_source=Entity&hybrid_search_source=Entity&hybrid_search_extra=%7B%22sourceType%22%3A%22article%22%2C%22sourceId%22%3A%22508879335%22%7D" \t "_blank). chunking任务是分类问题，即判断ngram是否一个phrase：L\_chunk = CrossEntropy(P(c\_i\_k = y\_i\_k))。

在推理阶段，最高信息分的ngram作为keyphrase.