中文分词对布尔检索系统性能影响相关研究

霍超凡

2019 年 12 月 10 日

摘要

布尔检索系统的关键在于索引的建立，对于英文，词与词之间使用空格或其他标点符号相分割，而对于类似中文的语言，建立索引的首要前提是分词，分词在自然语言处理技术是一项基础操作，在信息检索方面的分词和自然语言处理技术方面的分词有所不同，评价信息检索中的分词算法性能，不在于其分词是否准确，而在于将分词算法引入检索模型中能否提高检索系统的性能。本次实验实际测试了几类经典的分词算法，这些分词算法包括基于词典查找的分词方法、N-gram方法和基于HMM的分词方法，并且介绍几种用于自然语言处理的分词算法，如基于CRF的分词算法和基于BiLSTM的分词方法。最后对一些传统的方法进行了测试。

1 概述

分词算法包含以下几类，第一方法是一种暴力式的N-gram方法，这个方法将句子中词的所有可能组成情况都找出来，这样做的好处是查找词的召回率达到100%，但是精度极低。第二类方法是人为构造一个词典，根据词典进行分词，这类方法的劣势是不能很好的处理一些词典之外的词汇。第三类方法是基于统计的方法，基于HMM的分词方法属于这一类，首先将分词问题转换成标注问题，将句子序列是为观测状态，构造四种隐藏状态，句子中的每一个字只能处于词首、词中、词尾、单字四个隐藏状态之一。使用MP算法计算模型内部的各个参数，在分词时，使用维特比算法计算各个可能状态转移路径的概率，从中选取概率最高的作为最终分词结果。另外两种基于统计的方法是MHMM和CRF分词方法，这两类方法需要人为提取特征。第四类方法是基于学习的方法，这类方法和基于统计的方法不同，基于统计的方法是人为构造特征，然后使用训练集计算一些参数，而基于学习的方法是构造一个具有强表达力的模型，让模型拟合从序列到序列映射的函数，这类方法一般都将BiLSTM作为自己模型的核心部件。

2 分词算法

2.1 使用N-gram分词

N-gram分词算法将一个句子切分成若干个gram，对这些gram建立索引，方法简单且有效，这种方法不需要任何先验知识，适用于为那些动态变化的文档建立索引，比如我们经常使用的Sublime Text 就是根据这个原理进行建立索引查找，用户对程序中的变量取什么名字检索系统是未知的，用户输入的检索词也是未知的，这样建立的索引能够适应于任何查询，并且检索的结果的召回率能够达到100%。

2.2 基于词典查找的分词方法

基于词典的分词方法通过查找词典确定分词结果，根据句子和词典中的词之间的匹配方式又可以分为若干种方法，基于扫描的方向不同分为前向查询、逆向查询和双向查询，根据匹配的方式分为最大匹配和最小匹配，有的方法还结合语料库中词出现的概率，在遇到冲突时，优先选择出现概率较高的词。本次实验细致的考察了前向最大匹配算法、反向最大匹配、双向最大匹配和基于词频的匹配方法。

2.3 基于HMM的分词方法

将分词问题看成序列标注问题，一个句子是一个被观测的序列，被观测的序列中隐藏着一个序列，我们需要给定观测序列确定序列。HMM、模型构造了四种隐藏状态B、M、E、S，分别对应于词首、词中、词尾、单字，句子中每个字处于四种状态的一种，四个状态之间的转移满足一定的限制，词首状态可以转移到其它三种状态的其中任意一种状态中，单字状态可以转换到词首状态可以以保持其原有状态，词中状态只能从词首状态转移而来，并且词中状态只能转移到词尾状态，词尾状态既可以由词中状态转移得到又可以由词首状态转移得到，初始状态只能是词首和单字状态。初始状态的概率由语料库中单字词和多字词所占的比例确定，状态的转移概率也需要根据语料库统计求得。训练HMM模型仍然使用MP算法最大化训练集中各个词在模型中的概率。

2.4 其他分词方法

MHMM模型和CRF模型也使用处理序列标注问题，它克服了HMM的一些缺陷，人为引入几条特征，模型性能依赖于人为所规定的特征。随着神经网络的兴起，一些人将循环神经网络用于处理序列映射问题，循环神经网络相比感知器、卷积网络最大的特点在于其能够处理不定长的数据，这种网络结构普遍应用于自然语言处理中。由于传统RNN不能排除噪声，在RNN的输入端、输出端和内部状态转移端增加一个门结构，这个门由一个包含单隐层的万能逼近器组成，相当与一个分类器，能够排除噪音，什么数据能够进入网络，当网络到达什么状态才输出都通过门控制。将循环神经网络应用到我们的分词方面，我们需要处理的工作是如何将句子编码作为神经网络的输入，这个过程叫做词嵌入，中文由几万个字如何对这些字进行编码成为一个问题，一个理想的编码的方式是将两个相似的字具有相似的编码，这样的处理会大大减轻输入门的压力，自word2vec被提出后，将word2vec引入到分词方面，将句子中的词使用word2vec编码送入循环神经网络中，输出的序列为标注序列。

3 评价算法指标

受上次实验影响，本次实验和上次实验的思路基本相同，上次实验是关于英文的词干提取算法，本次实验是关于中文的分词方法，评价中文分词方法和上次实验评价stemmer算法所使用的标准十分相似。我们在指定标准时需要再次注意，我们信息检索评价分词效果，并不在于这个词切分的是否正确，切分的结果是否满足词是最小的不可再分的语义单位，我们评价分词算法需要将其放置在具体的检索环境下，检测算法对检索系统性能的影响。但是由于标准词汇一般会成为我们检索所使用的检索词，我们仍然要将分词的结果和标准分词结果相比较。

在实验中，我创新性地将编辑距离应用到比较算法分词结果和标准分词结果的准确度上，将句子中词与词之间的分割位置提取出来组成一条分割点序列，通过比较算法分词结果的分割点序列和标准分词结果的分割点序列之间的编辑距离，如果编辑操作插入操作较多，则说明欠分词，同理，如果删除操作较多，则说明分词过度，如果替换操作较多，这说明分词的结果错误较多。例如：对于“我们都是中国人“，算法分词结果为”我们“、”都“、”是“、“中国”、“人“，分割点序列为2，3，4，6，标准分词结果为”我们“、”都“、”是“、”中国人“，对应的分割点序列为2，3，4，两者之间的编辑距离为1，编辑操作为删除‘6’。这说明我们错误的在位置6插入了分割点，表现为过分割。使用OSI表示过分割指数，表示在分词过程中错误将一个词分成两个词所占的比例，USI表示欠分割指数，表示分词过程中将两个词划分到一个词占所有词的比例，WST表示错误分割的比例。

为了比较算法对外来词的处理能力，引入外来词识别能力指数OOVI，这个指数表示正确识别外来词占全部外来词的比例。使用OOV来表示外来词占语料库全部词的比例。

关于算法在实际布尔检索系统的性能，像上次实验一样，并没有实际搭建一个布尔检索系统来测试，而是利用布尔检索系统自身的特点，索引是根据分词结果建立的，算法的分词结构和标准分词结果相同，那么这个词在实际布尔检索中一定能够被检索出，按照这个想法。将每一个句子视为一个文档，统计分词结果各个词在语料库中出现的句子总数，和标准分词结果的句子总数进行对比，计算精度P、召回率R和F1。

4 实验

4.1 实验环境

本次实验均在Python环境下进行。

4.2 实验所用的数据集

本次实验所使用的数据集为学术界所使用的标准数据集，这个数据集曾经被用于第二届国际分词Bakeoff，数据集来自于香港大学（CITYU）、北京大学（PKU）和微软研究所（MSR）。数据集详细情况见表1。

表 1: 各个语料库具体情况

语料库	词语种类	词语个数	字种类	字个数
CITYU	69,085	1,455,629	4,023	2,403,355
PKU	55,303	1,109,947	4,698	1,826,448
MSR	88,119	2,368,391	5,167	4,050,469

4.3 实验结果

结果一：N-gram算法中gram长度对分词性能的影响 使用不同长度的gram对语料库中的单词进行分词，得到如表2所示的结果。从表2可以发现，随着gram 长度的增加欠分割指数增大，而过分割指数减小，这和我们预期的相同，令人惊讶的即使不采用任何复杂的算法，把单词分成1-gram，F1值可以达到0.8以上，这说明在语料库中有较多的单词以单字符出现。如果使用任意长度的gram对句子进行分词的话，召回率和外来词识别率均为1.0，但是具有极低的精度。

表 2: N-gram算法中gram长度对算法的影响（所测试的数据集为PKU

	OSI	USI	WSI	P	R	F1	OOVI	OOV
N-gram-1	0.597	**0.013**	**0.004**	0.793	**0.868**	**0.829**	0.017	0.031
N-gram-2	0.035	0.113	0.224	**0.912**	0.183	0.304	0.361	0.031
N-gram-3	0.008	0.301	0.187	0.882	0.080	0.147	0.222	0.031
N-gram-4	0.005	0.407	0.120	0.883	0.056	0.105	0.138	0.031
N-gram-5	**0.004**	0.466	0.100	0.878	0.050	0.096	0.106	0.031
N-gram	-	-	-	0.073	1.000	0.137	1.000	0.031

结果二：基于词典查找的分词算法对比 表3给出了前向最大匹配算法FMM、反向最大匹配算法BMM、双向最大匹配算法BiMM和最大化概率匹配算法MPM，可以从表中总结出以下几条规律，这几个算法的性能相差不大，各个指标均在0.95以上，而双向最大匹配和最大化概率匹配普遍优于前向传播和反向传播，双向匹配的精度较高，最大化概率匹配的召回率较高。

表 3: 基于词典查找的分词算法对比

指标	P			R			F1		
语料库	CITYU	MSR	PKU	CITYU	MSR	PKU	CITYU	MSR	PKU
FMM	0.956	0.967	0.956	0.960	0.986	0.957	0.958	0.976	0.966
BMM	0.953	**0.974**	0.955	0.956	0.983	0.970	0.955	0.974	0.963
BiMM	**0.964**	**0.974**	**0.964**	0.959	0.985	0.964	**0.962**	**0.979**	0.964
MPM	0.942	0.955	0.952	**0.977**	**0.995**	**0.985**	0.959	0.974	**0.968**

结果三：基于词典提取和HMM算法的分词效果 最大化概率匹配方法的欠分割长度和错误分割比例均比较低，相比其它算法而言，这种方法能够较大概率将单词划分正确，这也是为什么这个算法的召回率明显高于其它算法的原因。

表 4: 基于词典提取和HMM算法的分词效果对比

指标	OSI			USI			WSI		
语料库	CITYU	MSR	PKU	CITYU	MSR	PKU	CITYU	MSR	PKU
FMM	0.150	0.128	0.121	0.020	0.011	0.020	0.011	0.009	0.010
BMM	0.153	0.128	0.121	0.020	0.011	0.020	0.010	0.009	0.010
BiMM	**0.125**	0.104	0.096	0.020	0.011	0.023	0.010	0.009	0.010
MPM	0.144	0.116	0.107	**0.017**	**0.009**	**0.019**	**0.006**	**0.004**	**0.008**
HMM	0.150	**0.064**	**0.045**	0.020	0.048	0.065	0.011	0.028	0.032

结果四：以上三类算法对比 表5显示HMM算法小劣于基于词典匹配的分割算法，但是HMM不依赖于词典，能够较好的处理外来词汇。

5 结论

经过以上算法我们可以得到以下结论：

1. N-gram算法基本思想简单，虽然在本次实验的数据库中它的性能并不优于其它算法，但是它也有其自己的适用范围，它适合于出现对语料库和检索词均未知的情况，比如

表 5: 不同算法在PKU数据集上的性能对比

	OSI	USI	WSI	P	R	F1	OOVI	OOV
N-gram-1	0.597	**0.013**	**0.004**	0.793	0.868	0.829	0.017	0.031
FMM	0.121	0.020	0.010	0.956	0.957	0.966	0.020	0.031
BMM	0.121	0.020	0.010	0.955	0.970	0.963	0.020	0.031
BiMM	0.096	0.023	0.010	**0.964**	0.964	0.964	0.020	0.031
MPM	0.107	0.019	0.008	0.952	**0.985**	**0.968**	0.017	0.031
HMM	**0.045**	0.065	0.032	0.944	0.932	0.938	**0.540**	0.031

程序中的变量名，这种情况下，N-gram能够适应外来词汇。

2. 在基于词典查找的算法中，双向匹配和基于最大概率的匹配算法具有较高的性能，其中最大概率匹配算法正确分割比例较大，具有较高的召回率。

3. HMM算法不依赖于词典，能够较好的处理外来词汇，但是它的精度和召回率并不如传统的查找词典的方法。

6 文献评注

本次实验参考的文献并不多，这领域的一些文献极难阅读，没读懂的文章没有列在参考文献列表中，其中影响较为深刻的是[4]和[5]，[5]将中文分词问题视为标注问题，这篇文章具有开篇意义。[4]使用现代方法进行分词，神经网络的结果大致包括词嵌入和序列建模，更加细节的部分记不清楚，实验使用的数据集来自于[1]，参考的代码包裹[2][3]，实验中的MPM 算法参考[2]，HMM算法的实现来自于[3]。

参考文献

[1] Second international chinese word segmentation bakeoff data. `http://sighan.cs.uchicago.edu/bakeoff2005/`. Accessed December 08,2019.

[2] machine-learning-journey. `https://github.com/xlturing/machine-learning-journey`. Accessed December 08,2019.

[3] Chinese-word-segmentation. `https://github.com/jwchennlp/Chinese-Word-segmentation`. Accessed December 08,2019.

[4] Deng Cai, Hai Zhao, Zhisong Zhang, Yuan Xin, Wu Yongjian, and Feiyue Huang. Fast and accurate neural word segmentation for chinese. 04 2017.

[5] Nianwen Xue and Libin Shen. Chinese word segmentation as lmr tagging. 07 2003.