1. **引言**
   1. **概述**

在近十几年来，计算机领域发展迅猛，个人计算机，终端互联设配也越来越普及，几乎是人人随时随地都能上网，在当今互联网这个巨大的生态圈里面，每年互联网用户产生的数据正以惊人的速度增涨。这些逐年增长的数据中绝大多数的数据都是非结构化的数据，这些数据主要存储的形式都是文本、网页、图片、视频、音频等非结构化的形式。结构化的数据它是经过了一定的存储格式，处理规则加以封装，所以它能够被计算机根据其存储的规则来进行识别，而非结构化的数据比如说文本数据，这些数据没有规则，格式，通常情况下它是不能被计算机直接识别的。当人们想对这些数据进行分析，统计，挖掘的时候，由于其不是结构化的数据，很难通过数据挖掘的技术对其进行处理。人工筛选的方法在这个大数据时代显然是天方夜谭，虽然说人工筛选的方法在上世纪80年代之前都是的确存在的，有需求就有市场这句话放在科研的领域也是行的通的，正因为有对非结构化文本识别和分析的需求，所以研究者们也想出了各种相关的方法。文本挖掘（Text Mining），图像识别，模式识别等一些相关技术就对分析和识别这一些非结构化的数据起到很好的作用。在此篇论文中我们主要讨论一下文本挖掘技术。

文本挖掘(Text Mining)是一个从非结构化文本信息中获取用户感兴趣或者有用的模式的过程。首先它会把非结构化的文本数据通过一定的方法转换成为结构化的数据，然后再从这些结构化的数据中分析提取出我们用户所感兴趣的模式和东西。文本挖掘技术是一项非常复杂的技术，它是一个多学科混杂的一门学科，其中包含有信息技术、机器学习、统计学、数据可视化、数据库技术、数据挖掘。

集这么多门学科于一身，它在我们这个大数据时代，信息化时代发挥了重要的作用，也是在信息学科领域中比较热门的学科之一。

在生物医学领域，借助着计算机科学技术也在快速的发展，每年发表在互联网上的生物医学文献资料和研究者们在实验室产生的实验数据也正以惊人的速度增长，产生的这些文献资料和实验数据大部分都是以文本的形式存储在计算机设配上面。所以也是非结构化的数据，不能被计算机直接识别。那么我在此篇论文里主要讨论针对文本形式存储的非结构化数据的处理以及信息的提取。

* 1. **选题的背景和意义**

基因和蛋白质一直都是生物医学工程方面的两大热点词汇，人类从刚开始发现它们到现在一直都在研究它们，并从未停止过对它们的研究和实验。经过科学家们的众多次探索和实验让我们对它们有了一定的了解和认识，并且也享受到了研究成果给人类带来的益处.蛋白质是由氨基酸排列组合生成的，人体氨基酸种类有20种，这20种氨基酸排列组合可以生成成千上万种的蛋白质，随着研究者们不断深入研究蛋白质和不断实验，发表在科学文献期刊上的文献数量也随之增加，这个数量正在指数级地增长，如此多的学术文献给研究者们带来了无比多的信息和知识，但是蛋白质的种类如此繁多，有关蛋白质的文献也就越多，这么多的科学文献，研究者是不可能全部阅读完的，这样一来数量如此庞大的科学文献对于研究者们来说是一种浪费，然而研究者们发现他们在阅读每一篇文献时，仅仅只是要获取其中的蛋白质的关键信息，但是他们不得不逐一整片阅读，这大大阻碍了他们的研究效率，所以研究者们迫切需要借用计算机大数据的手段来支持他们完成更深入的探索和研究。他们希望通过计算机的文本挖掘和文本分析来帮他们迅速定位到他们想要得到的蛋白质信息，所以文本挖掘技术迫切需要应用与生物医学文献挖掘上，这也是我本论文的核心研究的方向。

在如此庞大的生物医学文献库当中加之以文本挖掘技术进行适当分析和处理，处理的结果给研究人员提供了一个客观可靠的数据基础，这大大地减少人工检阅和人工处理等繁杂的工作，提高了研究者们的效率，同时这些数据基础上研究人员们能够更深一步的论证、假设、探索、实验。从而加快了生物医学领域的发展进程。

此外再说一点个人观点，人类文明几千年，计算机技术从上个世纪问世到现在不到一百年的时间里，取得了飞快的发展，如此同时各个行业在享受着计算机带来的高效率之下，也取得了飞快的发展，所以人类社会在过去的一百年里取得了空前地发展。计算机，它作为一个20世纪最伟大的发明之一，造福于人类，那么它有义务去帮助人类解决各行各业的问题，因为它最初被设计出来就是为了解决辅助人类分担一些繁琐，重复的工作，解放了人类的双手，充分发挥了人类的大脑。所以解决像这种生物医学领域，或者其他领域一些棘手的里上遇到的问题，计算机有义不容辞的义务。试问如果计算机学科不承担这样的义务，那还有哪门学科能解决呢？

1. **入门概念**

**2.1蛋白质以及翻译后修饰概述**

蛋白质（protein）是生命的物质基础，是有机大分子，是构成细胞的基本有机物，是生命活动的主要承担者。中学课本学习过，蛋白质是由氨基酸构成的，而氨基酸是由基因表达产生的。所以可以认为蛋白质是基因表达的产物。蛋白质翻译是蛋白质合成过程中的第一步，翻译是根据遗传密码的中心法则，将成熟的信使RNA分子（由DNA通过转录而生成）中“碱基的排列顺序”（核苷酸序列）解码，并生成对应的特定氨基酸序列的过程。此过程产生的蛋白质是没有活性的，想要具有活性，常常要进行一个系列的翻译后修饰和加工，从而使其具有一定的功能和特性。据研究统计，人类有50%~90%的蛋白质都在翻译后进行了修饰。蛋白质的翻译后的化学修饰一般是对蛋白质的附加一些生物化学官能团或是改变其结构来让它改变其化学性质，从而让它具有某项特性或功能。翻译后的蛋白质正因为有了修饰加工这一个步骤，就可能使相同的蛋白质具有完全不一样的功能和特性，也大大地增加的蛋白质的复杂性和多样性。所以这种类远远地超过了有天然20中氨基酸经过排列组合可生成的蛋白质的种类。所以，一个基因表达产生一个蛋白的说法就不存在了。据统计，人体类的可以编码合成蛋白质的基因组有两万多个，但是却可以表达出高达20万种不同功能的蛋白质。

到目前已经发现了超过400种的蛋白质翻译后修饰的方式。但是目前发现的蛋白质翻译后修饰的方式还是比较少的，还有大量的蛋白质修饰过程还是未知的。然而我们比较常见的修饰方式有磷酸化、甲基化、泛素化、乙酰化、亚硝酸化、糖基化、氧化等。

**2.2文本挖掘技术概述**

文本挖掘（Text Mining）是以计算语言学、统计数理分析为理论基础 ,结合机器学习和信息检 索技术 ,从文本数据中发现和提取独立于用户信息需求的文档集中的隐含知识。它是一个从文本信息描述到选取提取模式 ,最终形成用户可理解的信息知识的过程。它其实也是数据挖掘技术的一个分支学科，它和数据挖掘不同的是，数据挖掘技术主要是对结构化的数据进行挖掘处理，比如说关系表（Table），事务表等，这些表里面的数据（特征）在积累的时候就已经通过键值对（key-map）的方式做好了结构化的存储，所以计算机在解析数据的时候根据不同key去取相应的数据，所以结构化的数据是能被计算机所识别的，但是文本挖掘技术主要针对的是文本形式存储的非结构化数据，比如说新闻，网页，书籍等，这些文本数据是通常是由人类所撰写，文本中充满了人类语言规则和定义，这些规则因地区而异，比如说不同国家的语言不一样，计算机必须为每种语言去适应其语法规则，或者时间而异，比如说随着科研不断发展，新的名词可能被创造，这个时候必须让计算机学习人类创造的新的词语，由此可见计算机是无法直接识别这些非结构化的数据。

**2.3 Pubmed生物医学文献检索系统.**

Pubmed是美国国家医学图书馆(NLM)所属的国家生物技术信息中心(NCBI)于2000年4月开发的,基于WEB的生物医学信息检索系统。PubMed的数据库来源于MEDLINE, OLDMEDLINE, Publisher-Supplied Citations，其中包含了大量的生物医学先关的期刊，论文，综述等。它给生物医学研究人员提供了丰富的有关蛋白质，基因等重要的文献信息。

1. **需求分析和系统设计**

**3.1需求**

需求最初是由中山大学生物工程学院的几位师姐提出来的，她们研究蛋白质翻译后修饰的时候会经常去Pubmed生物医学文献检索系统里面去找有关蛋白质翻译后的文献，比如想要找找有关蛋白质磷酸化修饰的文献，学习这一类修饰后的蛋白质的知识，她们可以在Pubmed文献检索系统里输入phosphorylation关键字，接着就可以找到几千或者上万篇有关磷酸化的一些文献。这些根据关键字检索到的文献的数量有点庞大，人工每一篇地去阅读显得有些吃力，但是她们发现她们每次阅读文献的时候都有一个共同点，就是只要找出文献里面的关键信息，比如说只要找出某某蛋白质被某某酶在某个位点上修饰了。这样问题就来了，能不能借助计算机来识别这些关键信息并找出这些关键信息呢？如果可以的话就不用通篇阅读了，大大地节省了时间，提高了效率。于是她们就提出了这样一个需求：通过计算机来识别并找出文献中的一些关键信息，关键信息就是蛋白质的名称，酶的名称，酶修饰蛋白质的位点。

**3.2 需求分析**

接到这个需求的时候，给我的第一感觉就是要在一篇文章中找出摘要信息，用几个关键词概括出全文，这有点类似于文章目录提取和文章摘要概括。于是我去Pubmed数据库查了几篇有关于磷酸化修饰的文章。文章的主要的意思就是描述了某个激酶在某个位点上修饰了某个蛋白质，当然也有一些是无关紧要的文章，这些无关紧要的文章我们要把它们除去，因为它们并不能给我们带来知识和信息。

下面是一篇标准的Pubmed数据库中关于磷酸化修饰的一篇文章：

<ArticleTitle>GRK6 phosphorylates IκBα at Ser(32)/Ser(36) and enhances TNF-α-induced inflammation.</ArticleTitle>

<AbstractText NlmCategory="UNASSIGNED">

G protein-coupled receptor kinases (GRKs) comprise a family of seven serine/threonine kinases that phosphorylate agonist-activated G protein-coupled receptors (GPCRs). It has recently been reported that GRKs regulate GPCR-independent signaling through the phosphorylation of intracellular proteins. To date, several intracellular substrates for GRK2 and GRK5 have been reported. However, those for GRK6 are poorly understood. Here we identified IκBα, a negative regulator of NF-κB signaling, as a substrate for GRK6. GRK6 directly phosphorylated IκBα at Ser(32)/Ser(36), and the kinase activity of GRK6 was required for the promotion of NF-κB signaling after TNF-α stimulation. Knockout of GRK6 in peritoneal macrophages remarkably attenuated the transcription of inflammatory genes after TNF-α stimulation. In addition, we developed a bioluminescence resonance energy transfer (BRET) probe to monitor GRK6 activity. Using this probe, we revealed that the conformational change of GRK6 was induced by TNF-α. In summary, our study demonstrates that TNF-α induces GRK6 activation, and GRK6 promotes inflammatory responses through the phosphorylation of IκBα.

</AbstractText>

在上示例文中以标签ArticleTitle为例，ArticleTitle内容为GRK6 phosphorylates IκBα at Ser(32)/Ser(36) and enhances TNF-α-induced inflammation.在这句话中GRK6是一种激酶，IκBα是一种蛋白质，Ser(32)/Ser(36)是修饰的位点，它们三者之间的关系是GRK6激酶在Ser(32)/Ser(36)位点上磷酸化修饰了IκBα蛋白质。

这就是研究人员想要找出的知识模型。

**3.3 用户用例分析**

需求我们已经知晓了，在这基础上，我们要建立一条完整地用户使用示例，这样我们才能了解整个系统的来龙去脉。假设一个用户需要在Pubmed生物医学文献检索系统中找出蛋白质磷酸化修饰的一些信息，用户可能想要输入关键词phosphorylate（磷酸化），然后用户想得到所有（当然不可能找出所有的被磷酸化修饰的蛋白质，有些蛋白质研究人员们还在进一步在探索，这里的所有指的是目前已经发现的并且已经被收录于MEDLINE文献数据库中的所有被磷酸化修饰过的蛋白质）被磷酸化修饰过的蛋白质，以及修饰的激酶和修饰的位点，并且这些数据以一种直观、可视化的图表来展示在他面前。简单抽象一点概括的话，用户输入一个关键词，然后期望得到一个可视化的图表。下图展示了用户和系统这样一种关系。

关键词

系统

System

用户

User

图表

**3.4 系统设计**

**3.4.1 系统工作流程**

根据以上需求描述和用户用例分析，我把此次的系统工作流程大致的分成以下5个步骤：

1、从Pubmed文献数据库中获取根据关键字或者关键词检索到得文献；

2、对这些原始文献文本做一次预处理，整理出具有一定格式的文档集；

3、针对于文档集中的每一个文档，识别出文档里面的命名实体；

4、挖掘出这些命名实体之间的关系；

5、对挖掘出来的数据和关系进行可视化处理。

**3.4.2 模块化设计**

根据以上总结的5大工作流程，我们可以发现以上5个步骤顺序执行，下一个步骤总是依赖上一个步骤，并使用上一个步骤产生的数据作为这一步的输入数据。每一个步骤都有一个输入数据和输出数据，由上往下以此传递。所以我们在实现整套系统的时候要依次实现这5个步骤，并且每个步骤都有一个input和output。每一个步骤都有一个明确的功能，我们可以把每一个步骤对应地划分成系统的各个模块，然后以其功能来命名。这样我们就可以把它划分成文本获取模块，文本预处理模块，命名实体识别模块，实体关系提取模块，数据可视化模块五大模块。系统整体设计图如下：

Key Words

2、文本预处理模块

Entity set

Docment set

Docment

3、命名实体识别模块

1、文本获取模块

4、实体关系提取模块

Picture，table

Entity & Relation

系统内部

5、数据可视化模块

各个模块的输入和输出：

|  |  |  |
| --- | --- | --- |
| 模块名称 | 输入数据 | 输出数据 |
| 文本获取 | 关键字，关键词 | 粗糙，冗余的文本数据 |
| 文本预处理 | 粗糙，冗余的文本数据 | 具有一定格式的文档集 |
| 命名实体识别 | 具有一定格式的文档集 | 命名实体词库 |
| 实体关系提取 | 命名实体词库 | 关系数据 |
| 数据可视化 | 命名实体词库，关系数据 | 实体+关系图表（统计） |

各个模块内部遵循模块相互独立原则，仅在模块的出口和入口提供对外数据交接的接口，这样就便于以后维护方便。

**3.4.3 工作时序图**

**3.4.4 软件开发架构**

从用户用例上来看，我们采用C/S（客户端/服务器）的架构来实现整套系统，因为从用户用例上，这是基于请求—响应式的模型的事件过程，这也是典型的C/S架构的模型，所以采取C/S架构是非常贴切的。

由于文本集数量庞大，在处理这么多文档集的时候会非常耗时，所以显然不能时时同步响应用户的请求，当然也不能处理完所有的文档再响应用户的请求，因为处理完所有的文档集再返回给用户，用户就回觉得系统响应非常慢，非常卡，从而造成用户体验效果不佳。所以我们服务端方面要每处理完一篇文档的时候就及时的返回数据给用户，直至处理完所有的文档集。这样的话，有数据时时返回，虽然不是全部的数据，但是用户就不会觉得卡，反应慢。客户端方面，要做到异步加载服务端传回的数据并展示。

为了加快系统的运行效率，服务端在处理文档集的时候应当采用多线程的方式，并发处理每一篇文献，因为在处理每一篇文献的事件是相互独立的，前提是每个线程在处理资源的时候，不能同时访问同一个文件，因为这样的话，可能会导致结果有重复，所以在线程调度方面，对于同一个文件的处理的时候要加上互斥锁，以保证每个线程在相同时刻都在处理不同的文件。

服务端

Server

客户端

Client

START

处理文档

是否end

END

**3.4.5 开发环境**

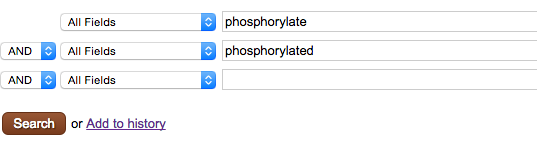
开发语言；Java 、操作系统：MacOS

Java是一种跨平台式的语言，可以方便移植到不同平台使用，所以此次开发主要采用Java语言。由于本人使用MacOS操作系统的笔记本工作，所以操作系统自然就是MacOS

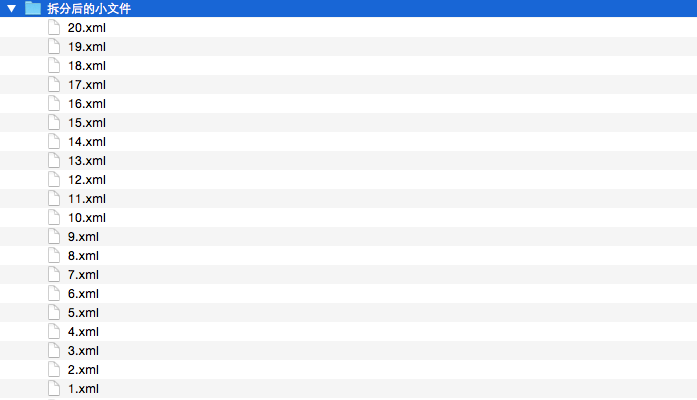
1. **程序设计和系统实现**

**4.1 文本数据源获取**

由于需求提出者提出了主要针对蛋白质磷酸化修饰的信息挖掘，所以关键词即为’磷酸化’（phosphorylate）。Pubmed文献检索系统上收录的大部分是以英文为语言的生物医学文献，所以我们要把关键字’磷酸化’翻译成phosphorylate来作为Pubmed文献检索系统的关键词。Pubmed文献检索系统支持AND和OR来接连关键词，这样就可以限制和放宽检索的范围，因为此次研究的对象是整个和磷酸化有关的蛋白质文献，所以我们在输入关键词的时候把磷酸化的主动式（phosphorylate）和被动式（phosphorylated）两个词通过OR的方式来扩大检索的范围，通过检索后，最后我们从中得到了76604个文献文本结果。文献结果以XML的文件格式下载到本地文件系统上面，这样我们就得到了一个最原始的文本数据。



通过以上方法得到了单一的文件，里面包含了76604篇有关磷酸化的文献，文件大小1.05GB，一共2000多万行，这么大的文件肯定不能同时加载到内存里去处理，所以我就把这么大的文件拆分成20个50多M的小文件，这样每个文件里包含了3000-4000篇的文献，用1-20来对每个拆分过的文件命名，这样下来，我们就得到了20个XML文件，这20个文件就是我们文本数据源，也是文本数据获取模块的输出，文本预处理模块的输入。



**4.2 文本预处理**

文本预处理是文本挖掘中的一个关键的步骤，预处理的结果可以直接影响到最终挖出数据的质量和效果，所以这一步非常关键。文本预处理的主要的思想就是把源文本数据简单化，结构化，便于下一步的处理。

经过上一步的处理，已经得到了源文本数据，这一步就对这些源文本数据作进一步的处理。仔细观察这20个XML文件的内容，下面是一篇XML的大致的一个格式：

<Data>

<PubmedArticle>

<MedlineCitation>

<PMID Version="1">25901902</PMID>

<Article>

<ArticleTitle>title</ArticleTitle>

<AuthorList>...</AuthorList>

<Abstract>

<AbstractText> articeText <AbstractText>

...

</Abstract>

</Article>

...

</MedlineCitation>

<PubmedData>…</PubmedData>

</PubmedArticle>

<PubmedArticle>

......

</PubmedArticle>

</Data>

发现里面对一篇文献描述的时候附加了很多的信息，比如说创建日期、收录日期、公司版权、作者等等，这些信息都是我们不关注的，无关紧要的信息，但是却占用非常大的空间内存，同时这些冗余的数据可能还会对接下来的文本处理带来不必要的麻烦。我们只关注数据是这篇文献的标题<ArticleTitle>，文献内容<ArticleTitle>，pubmed\_ID标示符<PMID>，所以在文本预处理的阶段，我们要把这些冗余，无关紧要的数据给去掉，只要求保留Pubmed标示符、文献标题、文献内容这三个关键数据。这样下来，我们就做到简化了数据的目的。接下来我们要使这些文本数据具有一定的结构（这里的结构化并不是只能被计算机直接识别的那种结构化）。于是我们把每一个文件里面的3000-4000文献拆分成3000-4000个小txt文档，txt文档的文件名就是序号加文献标题，txt文档里的内容就是文献内容，于是我们就可以得到70000多个文献文档，这就是我们对文本预处理之后得到的文档集（Document Set），每一篇文档都对应于一篇文献，并且包含了文献的标题和内容。

用Java程序实现上述的过程。首先XML文件的内容是一种它本身就是一个标示性可扩展标记语言（Extensible Markup Language），我们可以通过一些java开源库提供的API来对其进行解析。dom4j是一个非常优秀的JAVA开源库，处理的效率高，对于这么庞大的文本数据来说，dom4j的处理比其他的开源库的效果会更好。首先对20个XML文件进行for遍历，对于每一个文件，遍历每一篇文献<PubmedArticle>从中解析得到一些属性和标签，由于只要文献标题和文献内容，所以直接取<ArticleTitle>和<AbstractText>这两个标签下面的内容，然后再把<AbstractText>内容输出到新建文档中，文档命名为序号<ArticleTitle>。如果我们只在单线程拆分这些文件的话，效率可能不够高，我们也可以用多线程来预处理，用多线程的话，就不用for遍历这20个文件了，可以直接启动20个线程，每个线程里处理一个文件，启动20个线程的好处就是每个线程都处理不同的文件，不会发生资源互斥的问题。

代码里面我先创建一个XMLParser静态工具类，里面全部实现两个静态方法，一个是解析XML得到里面的文献标题articleTitle和文献内容articleContent，另个是把文献标题和内容输出到一个文本文档里面，构成文档集中的一个文档。解析函数的代码如下：

public static void parseXml(String fileName) throws Exception {

SAXReader reader = new SAXReader();

Document document = reader.read(new File(fileName));

Element root = document.getRootElement();

// 将解析出来的Data下的PubmedArticle放在list中

List list = root.elements();

int k = 0;

for (Iterator i = list.iterator(); i.hasNext();) {

k++;

//输入的文件名称

String outFileName = "";

//输入的文件内容

String fileContent = "";

//PubmedArticle标签

Element PubmedArticle = (Element) i.next();

//MedlineCitation标签

Element MedlineCitation = PubmedArticle.element("MedlineCitation");

//！！！最关键的文献标签

Element Article = MedlineCitation.element("Article");

//最关键的文献的标题

Element ArticleTitle = Article.element("ArticleTitle");

//文献主体内容

Element Abstract = Article.element("Abstract");

//输入文件名称设置为序列号+文献的标题

outFileName = k + ". " + ArticleTitle.getText();

System.out.println(outFileName);

//因为linux下面文件名字的最大的长度为256，所以超过了的话要进行裁剪

if (outFileName.length() > 250) {

outFileName = outFileName.substring(0, 250);

}

//Abstract有可能为空

if( null == Abstract ){

continue;

}

//Abstract里面有很多段落，段落的列表

List eList = Abstract.elements();

for (Iterator j = eList.iterator(); j.hasNext();) {

Element AbstractText = (Element) j.next();

fileContent += AbstractText.getText() + "\n";

}

addDocSet(outFileName, fileContent);

}

}

写入文档集代码如下：

//文档集目录

private static String DOC\_SET\_PATH = "xxx";

//增加到文档集里面去

private static void addDocSet(String fileName, String fileContent)

throws Exception {

if (fileName.equals("")) {

return;

}

//由于java会把"/"当成子目录处理，所以要替换成：进行区别

fileName = fileName.replace("/", ":");

File file = new File(

new File("DOC\_SET\_PATH"),

fileName + ".txt");

OutputStream os = new FileOutputStream(file);

os.write(fileContent.getBytes());

os.flush();

os.close();

}

代码中有四个值得注意的地方，一是XML文件里面有些文献的标题的长度非常长，超过了linux文件系统下面支持的最大文件名称长度256字节，所以如果出现超过256的字节要进行相应的裁剪。二是有些PubmedArticle标签下没有包含Abstract标签，所以在处理的时候要加以判断，如果Abstract为空的话，那么就跳过这篇文献的处理，如果还继续输出一个空白的txt文档的话，空的文档对我们的数据挖掘根本没有贡献，反而在文本挖掘的时候或者统计的时候还可能会导致误差。三是有些Abstract标签下面会包含多个子标签，这些子标签里面的内容都是文献的内容，所以在输出的时候要把他们全部输出，不仅仅只输出单个的AbstractText。四是文献标题字符串中会有一些”/”这样的字符，我们把它作为文件名称输出的话，java的File类会认为这是一个子目录的标示符（类似dir/sub里面’/’表示的子目录的意思），所以在处理文献标题的时候要把标题里面的’/’字符用:代替，这样File类就会把这当成文件名称去处理。

**4.3命名实体识别和实体关系提取**

**4.3.1命名实体识别概述**

命名实体识别是文本挖掘过程中一个重要的过程，它在文本挖掘中的作用就是找出文本中具有特定意义的实体词语，比如说人名，歌名，影视名等等，同样地，生物医学命名实体识别(Biomedical Named Entity Recognition, Biomedical NER)也是生物医学文本挖掘过程中的关键任务,其目的是从生物医学文本集合中识别出指定类型的名称, 如蛋白质、基因、核糖核酸、脱氧核糖核酸等。这是进一步抽取关系和其他潜在信息的关键步骤。

命名实体识别是整个文本挖掘过程中的关键任务，因为有了识别出的实体，接下来的步骤才可能得到执行，同时它也是非常复杂的任务，里面夹杂着概率统计学，人工智能，机器学习，自然语言处理等各方面的学科知识。目前命名实体识别的方法主要分成三类，一是基于规则的方法，二是基于词典的方法，三是基于概率统计的方法，基于规则的方法就是特定的实体词语在文本中总是以一定的格式和规则呈现的，比如说书名，它一般是出现在《》书名号里面的。大多数的实体名词都是没有明显的特征和规则的，所以这种基于规则的实体命名方法非常不精准，而且局限性很大，但是它简单快捷。第二种命名实体识别的方法是基于词典的，它的思想就是把所有我们想研究的实体名词都收录在一本词典上，当在解析文本的时候就回去词典库里面去进行匹配，就像我们去查字典一样，这一方法需要人工录入巨量的词语到词典库里，它在识别词语的时候不会根据上下文的语境来识别，更加不会识别一词多义，有歧义的语法情况，只能匹配词典库里的词语，词典库里面有就能匹配，没有就不能匹配，所以这种方法在处理人类语言的时候不免捉襟见肘，漏洞百出。第三种实体识别的方法是基于概率统计的方法，也可以说成是基于机器学习的方法，这种方法也是在当今命名实体识别领域里面研究的重点，因为这种方法能够较为精确地识别出文本中的实体名词。它能够根据上下文语义环境来识别出里面的文本中的实体，并且能够识别出一词多义的词语。它的主要思想就是我们人工的将训练文档集中的实体名词找出来，并给其赋予一个词性标注，再根据上下文关系统计出一套可以让计算机学习的识别模型（语料库），然后计算机就能够根据这个模型来识别出未知文本中的实体名词了。这样说的比较抽象，打个比方，我们要识别south china university of technology（华南理工大学）这个学校实体名词，首先我们的（识别模型）语料库里面有了这个词语，并且这个词语我们已经给与了其学校词性的标注，统计出了五个单词中每个单词出现在这五个单词中的概率和出现在整个语料库的次数，接下来有了这套概率，计算机在识别未知文本中的south | china | university | of | technology词语的时候就每个单词出现在上下文的环境下的概率最高的就是当它出现在语料库中词组south china university of technology学校实体名词，这样计算机就识别出了一个学校实体名词。目前有很多种建立语料库的概率模型，如贝叶斯模型、隐马尔可夫模型、支持向量机等等，这些方法都需要大量的训练数据，但是相对准确。

**4.3.2 ABNER命名实体识别工具**

生物医学领域上的应用非常广泛，ABNER，BallE等工具是比较出著名的命名实体识别的工具，本文以ABNER为实验工具来进行实验和开发，我们选择它的原因主要是第一，它是基于JAVA平台下开发的软件；第二，它是一个开源的系统；第三，它能够提供给我们JAVA的API接口，可以让我们进行第二次开发，我们能够把它的代码嵌入到我们的系统中去。

ABNER是一款用于生物医学文本分析的软件，主要用于识别生物医学实体。它所使用的识别的方法就是基于统计概率的方法，首先它包含了NLPBA和BioCreative两个庞大的语料库其中包含了绝大多数已经发现的生物医学的实体，这两个语料库的F1测度分别是70.5和69.9。它目前能够识别出得实体的类型大致有五类：蛋白质，DNA，RNA, 细胞系，细胞类型。此外，它不仅提供我们识别命名实体的接口，还提供了训练数据集、培养语料库的接口，所以理论来说，我们只要有足够多的语料库，我们就可以识别更多类型的实体，也不仅仅仅限于生物医学实体的识别。ABNER本质上就是一个基于概率统计的机器学习系统，采用了基于上下文特征的线性条件随机场的方法，线性条件随机场方法又是建立在隐马尔科夫模型（Hidden Markov Model，HMM）上的，隐马尔科夫模型是一个可以通过可观察到的状态（好比文本单词，这是我们可以观察到的状态）根据相关概率去发现隐藏的状态（好比文本单词的词性标注，单词词性标注是我们未知），所以如果发现了未知文本的词性标注，就可以识别出了实体。

应用ANBER工具到我们的系统。首先我们进入ANBER网站<http://pages.cs.wisc.edu/~bsettles/abner/>，下载abner.jar可执行的jar包，这也是可以依赖的三方库。如果直接运行的话，可以得到以下的画面，输入一段待识别的文本，按下annotation按钮，它就可以识别出了文本中的生物医学实体，并且用不同的颜色进行标注和高亮显示出来。现在我把它通过三方库的方式引用到我的工程里面。首先实例化一个Tagger的对象Tagger t = new Tagger();再条用Tagger里面的方法getEntities(fileContent, "PROTEIN");函数里面的第一个参数表示文献文本字符串，第二个是具体的哪一个标签，我们要识别文本里面的蛋白质实体就可以用"PROTEIN"作为函数的第二个参数。这样我们就可以获取文献里面所有的蛋白质的实体，下面是部分关键的代码：

public static void recognizerSingleFile(File file) throws Exception {

String fileContent = getAbstractContent(file);

Tagger t = new Tagger();

String protein[] = t.getEntities(fileContent, "PROTEIN");

for (int i = 0; i < protein.length; i++) {

System.out.println(protein[i]);

}

}

由于系统还要挖掘出与蛋白质磷酸化修饰有关的激酶和修饰位点，所以我们必须还要继续识别文本中的激酶和修饰位点的实体名词，但是ABNER工具只提供了蛋白质，DNA，RNA, 细胞系，细胞类型五个类型的实体，所以我们还要对这些实体进行训练，培养一个能够识别激酶和修饰位点的语料库。ABNER工具给我们提供了训练语料库的接口。我们可以初始化一个Trainer的实例，调用Train里面的train(trainFile, modelFile)方法，trainFile就是训练文本的路径，modelFile就是语料库模型文件的路径，比如想识别一个磷酸化修饰的位点(Ser32)，我们可以在trainFile的文件里面填写训练的规则，官方的demo里面介绍了如何填写训练集文件，下面是官网demo的一个例子：

<pre>

IL-2|B-DNA gene|I-DNA expression|O and|O NF-kappa|B-PROTEIN B|I-PROTEIN activation|O ...

</pre>

训练文件的规则概括成四点：一，每一句话要按照每个token（可以理解为每个单词）空格间开，并且一句话写在一行；二，每一个token的左边的单词，右边是词性标注，两者用‘|’符号间隔开；三，每一个实体的第一个单词的词性标注前要加前缀B-，剩下其他的单词前面要加前缀I-；四，每个不是实体的单词的词性标注要设为O。用着我们训练一段能够识别位点的训练集：

GRK6|O phosphorylates|O IκBα|O at|O Ser32|B-POSITION or|O Ser36|B-POSITION and|O enhances|O TNF-α-induced|O inflammation|O .|O

生成一个model，然后用model作为语料库去识别出文档里面的为位点实体。代码：

//第一步 训练得到语料库

Trainer t = new Trainer();

t.train(trainTextPath, modelTextPath);

//第二步，用训练出来的语料库去识别文本

Tagger t = new Tagger(new File(Config.MODEL\_PATH));

String res[] = t.getEntities("GRK6 phosphorylates at Ser32 and enhances TNF-α-induced inflammation.", "POSITION");

经过以上的步骤，我们可以通过abner里面自带的语料库去识别文本中的命名实体，也可以自己训练想要识别的实体的语料库来识别。

**4.4 实体关系提取**

由于Abner里面的语料库只训练了蛋白质，DNA，RNA，细胞等五类的实体，至于蛋白质激酶，蛋白质的修饰位点这些实体，ABNER并没有提供识别这些实体的语料库，自己从头训练的话有点不切实际，因为训练的的数据量越多越精确，