day02-智能分类

第一章-人工智能与机器学习

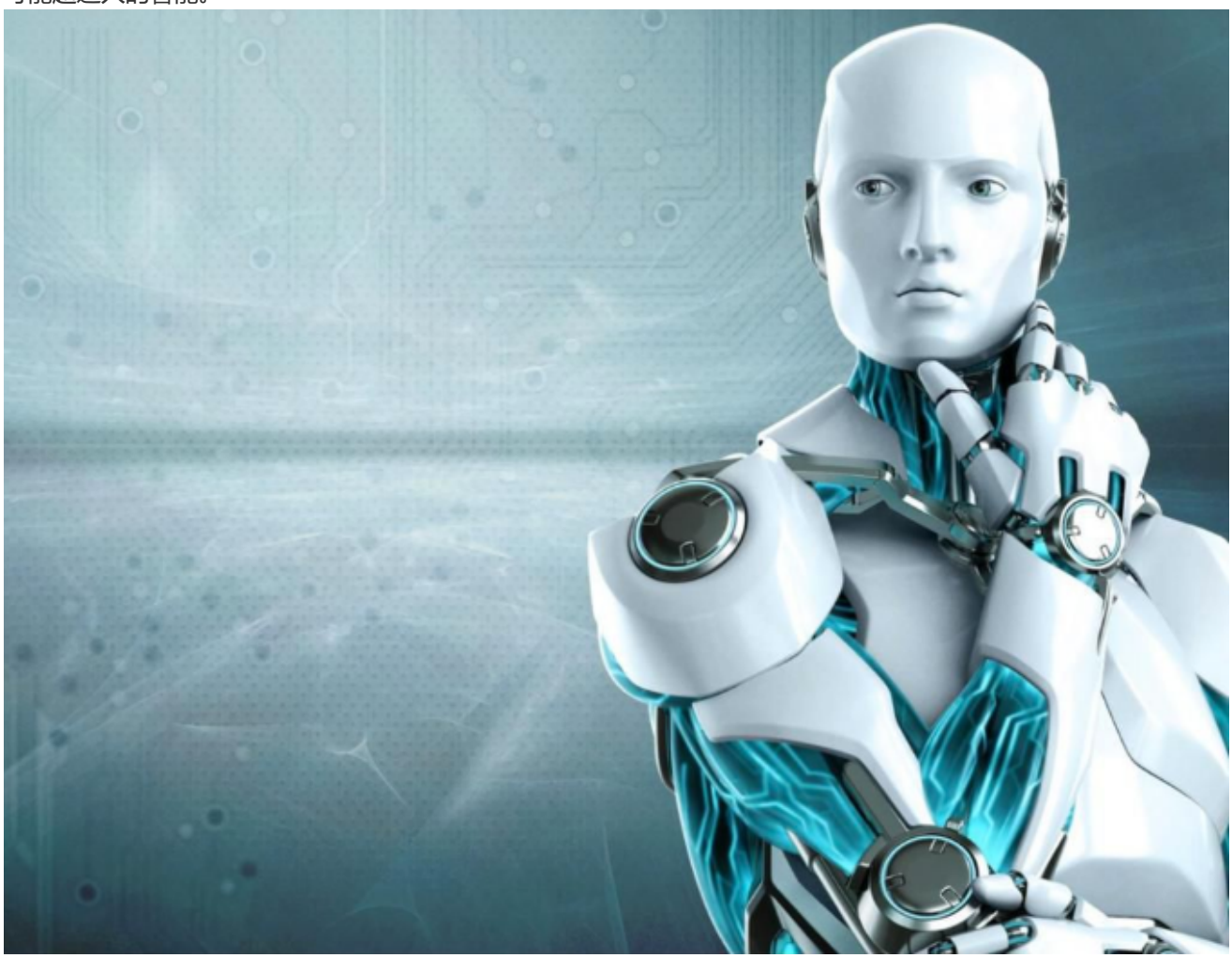
1.谈谈人工智能

1.1什么是人工智能

人工智能（Artificial Intelligence），英文缩写为AI。它是研究、开发用于模拟、延伸和扩展人的智能的理论、方法、技术及应用系统的一门新的技术科学。

计算机 计算能力 人处理事件 开车 程序

人工智能是计算机科学的一个分支，它企图了解智能的实质，并生产出一种新的能以人类智能相似的方式做出反应的智能机器，该领域的研究包括机器人、语言识别、图像识别、自然语言处理和专家系统等。人工智能从诞生以来，理论和技术日益成熟，应用领域也不断扩大，可以设想，未来人工智能带来的科技产品，将会是人类智慧的"容器"。人工智能可以对人的意识、思维的信息过程的模拟。==人工智能不是人的智能，但能像人那样思考、也可能超过人的智能。==



1.2人工智能的热门方向

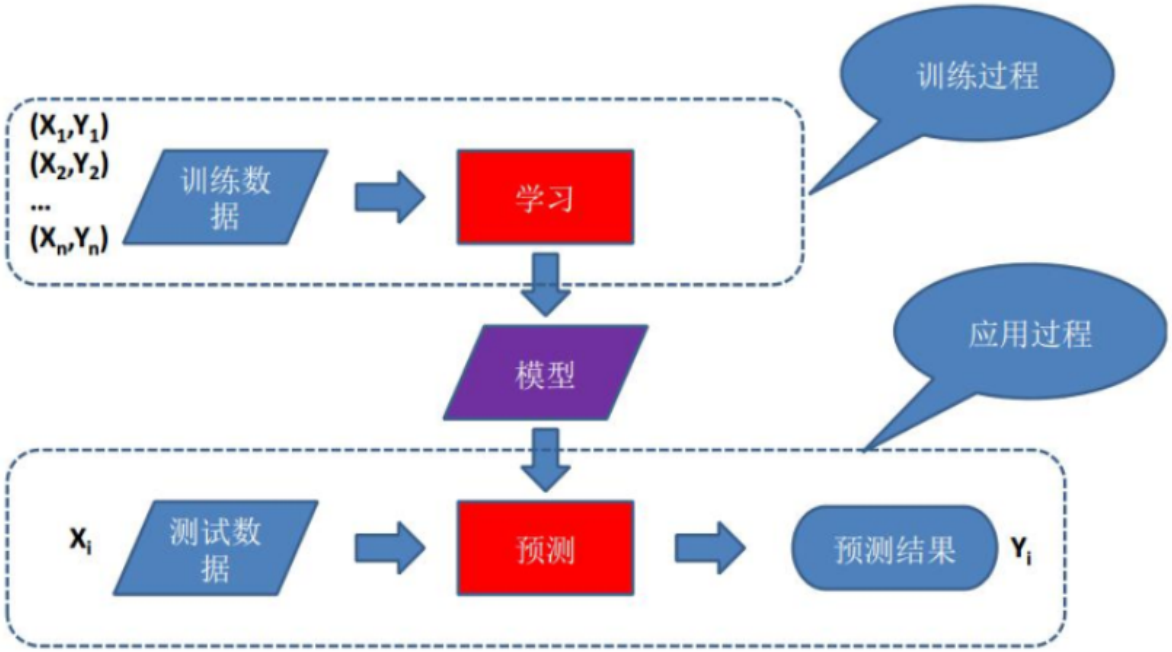
2.人工智能的三次浪潮

- 1950-1970 符号主义流派：专家系统占主导地位
  - 1950：图灵设计国际象棋程序
  - 1956 Artificial Intelligence提出
  - 1962：IBM 的跳棋程序战胜人类高手（人工智能第一次浪潮）
- 1980-2006统计主义流派：主要用统计模型解决问题
  - 1993： SVM模型
  - 1997：IBM 深蓝战胜象棋选手卡斯帕罗夫（人工智能第二次浪潮）
- 2006-至今神经网络、深度学习、大数据流派
  - 2006 DNN（深度神经网络）
  - 2016：Google AlphaGO 战胜围棋选手李世石（人工智能第三次浪潮）

3.机器学习

3.1什么是机器学习

机器学习是一门学科，它致力于研究如何通过计算（CPU和GPU计算）的手段，利用经验来改善（计算机）系统自身的性能。 ==它是人工智能的核心==，是使计算机具有智能的根本途径，应用遍及人工智能各领域。 数据+ 机器学习算法= 机器学习模型 有了学习算法我们就可以把经验数据提供给它，它就能基于这些数据产生模型。



3.2人工智能、机器学习、深度学习的关系

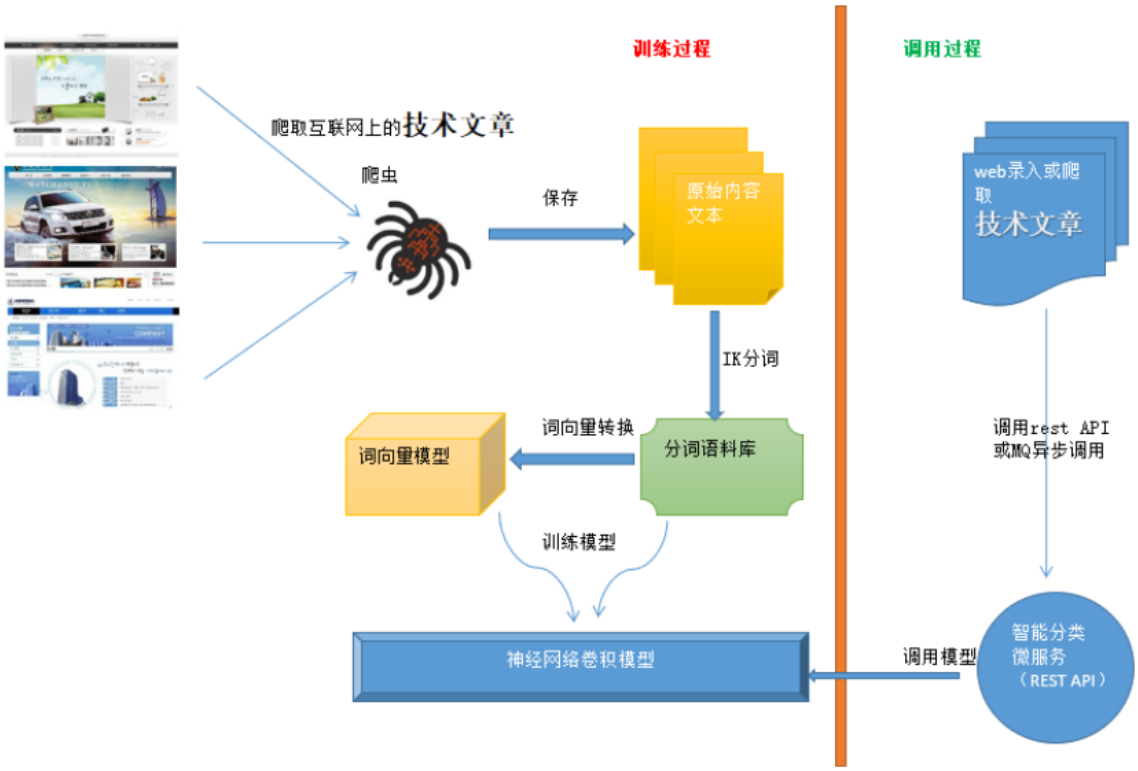
机器学习作为人工智能的一种类型，可以让软件根据大量的数据来对未来的情况进行阐述或预判。深度学习是实现机器学习的一种技术。

第二章-智能分类

1.需求分析

通过机器学习，当用户录入一篇文章或从互联网爬取一篇文章时可以预测其归属的类型。

2.智能分类的执行流程



第三章-智能分类涉及到的技术

1.IK分词器

1.1IK分词器简介

IK Analyzer 是一个开源的,基于java 语言开发的轻量级的中文分词工具包。从2006年12 月推出1.0 版开始,IKAnalyzer 已经推出了4 个大版本。最初,它是以开源项目Luence为应用主体的,结合词典分词和文法分析算法的中文分词组件。从 3.0版本开始,IK发展为面向 Java的公用分词组件,独立于 Lucene项目,同时提供了对 Lucene的默认优化实现。在 2012版本中,IK实现了简单的分词歧义排除算法,标志着 IK分词器从单纯的词典分词向模拟语义分词衍化。

1.2快速入门

- 创建工程引入依赖

```
        <dependency>
           <groupId>com.janeluo</groupId>
           <artifactId>ikanalyzer</artifactId>
           <version>2012_u6</version>
       </dependency>
```

- 编写代码

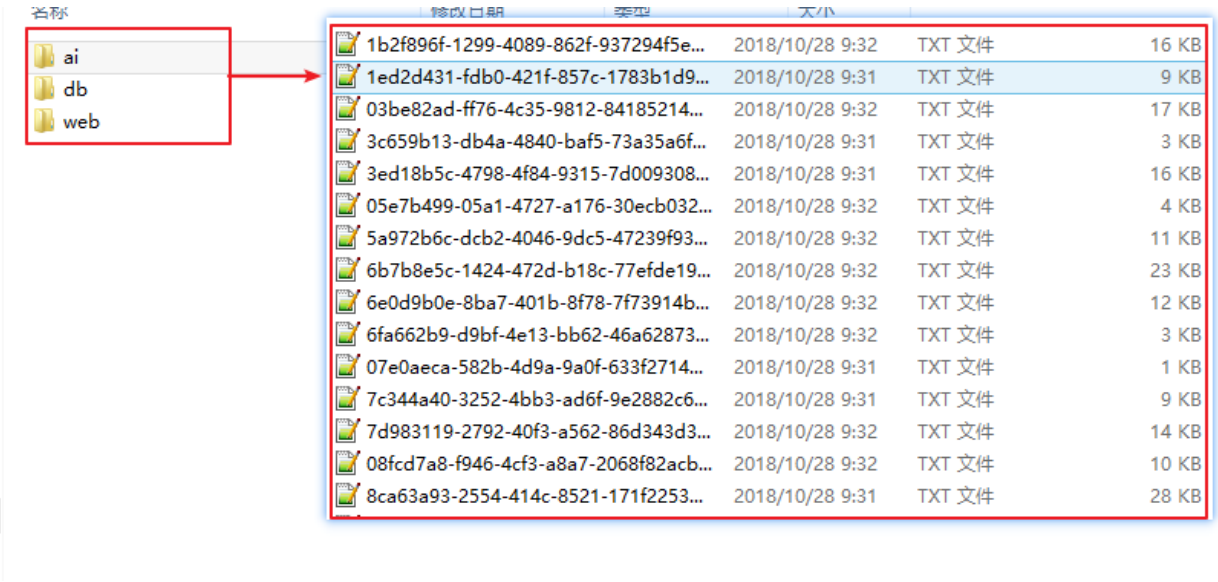
```
public class IKDemo {
```

```
    public static void main(String[] args) throws IOException {
        //0创建StringReader对象
        String str ="IK是基于中文的轻量级分词器";
        StringReader reader = new StringReader(str);
        //1.创建Ik分段器(参数二userSmart 为切分粒度 true表示最大切分 false表示最细切分)
        IKSegmenter ikSegmenter = new IKSegmenter(reader,false);
        //2.进行分词
        Lexeme lexeme = null;
        while ( (lexeme = ikSegmenter.next()) != null){
            System.out.println(lexeme.getLexemeText());
        }
    }
}
```

1.3构建分词语料库

1.3.1需求

按照十次方划分的频道，分别在CSDN上抓取各类文章，并以分词形式保存，词之间用空格分隔



1.3.2代码实现

- 修改tensquare_common模块的pom.xml ，引入依赖

```
<dependency>
    <groupId>com.janeluo</groupId>
    <artifactId>ikanalyzer</artifactId>
    <version>2012_u6</version>
</dependency>
```

- 修改tensquare_common模块 ，拷贝资料中`分词工具类IKUtil`，`HTML工具类(用于将文本中的html标签剔除)HTMLUtil`
- 在配置文件 tensquare_article_crawler , application.yml中添加配置

```
ai:
  dataDir: F:/workspace/sk/sz/ai/crawler
```

- 修改tensquare_crawler模块，创建ArticleTxtPipeline，用于负责将爬取的内容存入文本文件

```
@Component
public class ArticleTxtPipeline implements Pipeline {

    @Value("${ai.dataDir}")
    private String dataDir;

    private String channelId;

    public void setChannelId(String channelId) {
        this.channelId = channelId;
    }

    @Override
    public void process(ResultItems resultItems, Task task) {
        try {
            //获得标题
            String title = resultItems.get("title");
            //获得内容(去掉HTML标签)
            String content = HTMLUtil.delHTMLTag(resultItems.get("content"));


            //将标题+正文分词后保存到相应的文件夹
            PrintWriter printWriter = new PrintWriter(new File(dataDir + "/" + channelId + "/" +
UUID.randomUUID() + ".txt"));
            String splitStr = IKUtil.split(title + " " + content, " ");
            printWriter.print(splitStr);

            printWriter.close();

        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

- 修改ArticleTask类，新增爬取db 和web的任务

```
@Component
public class ArticleTask {

    @Autowired
    private ArticleProcessor articleProcessor;


    //@Autowired
```

```
    //private ArticleDbPipeline articleDbPipeline;

    @Autowired
    private ArticleTxtPipeline articleTxtPipeline;


    @Autowired
    private RedisScheduler redisScheduler;



    @Scheduled(cron="35 31 09 * * ?")
    public void aiTask(){
        Spider spider = Spider.create(articleProcessor);
        spider.addUrl("https://blog.csdn.net/nav/ai");
        articleTxtPipeline.setChannelId("ai");
        spider.setScheduler(redisScheduler);
        spider.addPipeline(articleTxtPipeline);
        spider.addPipeline(new ConsolePipeline());
        spider.start();
    }

    @Scheduled(cron="20 33 09 * * ?")
    public void dbTask(){
        Spider spider = Spider.create(articleProcessor);
        spider.addUrl("https://blog.csdn.net/nav/db");
        articleTxtPipeline.setChannelId("db");
        spider.setScheduler(redisScheduler);
        spider.addPipeline(articleTxtPipeline);
        spider.addPipeline(new ConsolePipeline());
        spider.start();
    }

    @Scheduled(cron="20 34 09 * * ?")
    public void webTask(){
        Spider spider = Spider.create(articleProcessor);
        spider.addUrl("https://blog.csdn.net/nav/web");
        articleTxtPipeline.setChannelId("web");
        spider.setScheduler(redisScheduler);
        spider.addPipeline(articleTxtPipeline);
        spider.addPipeline(new ConsolePipeline());
        spider.start();
    }

}
```

2.Deeplearning4j之Word2Vec

2.1Deeplearning4j简介

Deeplearning4j是第一个为Java和Scala编写的商业级开源分布式深度学习库。DL4J可与Hadoop和Apache Spark集成，可将AI引入业务环境，以便在分布式GPU和CPU上使用。

官网: https://deeplearning4j.org/

中文网: https://deeplearning4j.org/cn/overview

2.2Word2Vec简介

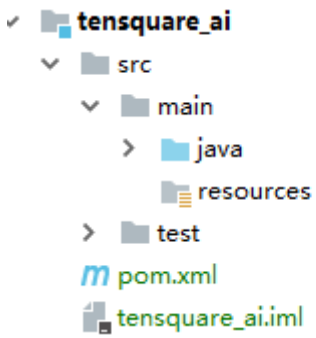
Word2vec是由Google在2013年开源的一款用于词向量计算的工具 ，word2vec可以在百万数量级的词典和上亿的数据集上进行高效地训练；其次，该工具得到的训练结果——词向量（word embedding），可以很好地度量词与词之间的相似性。Deeplearning4j为java实现了一个分布式的Word2vec.

说白了Word2Vec就是一种工具, 将自然语言中的字词转为计算机可以理解的稠密向量（Dense Vector）, (计算机可以理解的语言)

2.3构建词向量模型

我们在第三章中已经构建了分词语料库,但是有三个文件夹,很多文件,不便于进行词向量转换. 所以我们先把分词语料库合并成一个文件, 再通过Word2Vec词向量转换,构建词向量模型.

2.2.1分词语料库合并实现



- 修改tensquare_common模块 ，拷贝资料中`文件工具类FileUtil`
- 创建模块tensquare_ai ,引入坐标依赖

```
<dependency>
    <groupId>com.tensquare</groupId>
    <artifactId>tensquare_common</artifactId>
    <version>1.0-SNAPSHOT</version>
</dependency>
```

- 创建application.yml

```
ai:
  #语料库汇总文本文件
  wordLib : F:/workspace/sk/sz/ai/article.txt
  #语料库目录
  dataDir : F:/workspace/sk/sz/ai/crawler
```

- 创建启动类

```
@SpringBootApplication
@EnableScheduling
public class AiApplication {
    public static void main(String[] args) {
        SpringApplication.run(AiApplication.class,args);
    }
}
```

- 创建业务类

```
package com.tensquare.ai.service;


import org.springframework.beans.factory.annotation.Value;
import org.springframework.stereotype.Service;
import utils.FileUtil;

import java.io.IOException;
import java.util.List;

@Service
public class Word2VecService {

    //语料库目录
    @Value("${ai.dataDir}")
    private String dataDir;

    //语料库汇总文本文件
    @Value("${ai.wordLib}")
    private String wordLib;

    //词语合并
    public void  mergeWord(){
        List<String> files = FileUtil.getFiles(dataDir);
        try {
            FileUtil.merge(wordLib,files);
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

- 创建任务类TrainTask

```
@Component
public class TrainTask {

    @Autowired
```

```
    private Word2VecService word2VecService;

    @Scheduled(cron = "0 50 9 * * ?")
    public void train(){
        System.out.println("合并词料库开始...");
        word2VecService.mergeWord();
        System.out.println("合并词料库结束...");

    }

}
```

2.2.2构建词向量模型实现

- pom.xml引入依赖

```
    <properties>
        <dl4j.version>1.0.0-beta</dl4j.version>
        <nd4j.version>1.0.0-beta</nd4j.version>
        <nd4j.backend>nd4j-native-platform</nd4j.backend>
    </properties>
    <dependencies>
        <dependency>
            <groupId>org.deeplearning4j</groupId>
            <artifactId>deeplearning4j-core</artifactId>
            <version>${dl4j.version}</version>
        </dependency>

        <dependency>
            <groupId>org.deeplearning4j</groupId>
            <artifactId>deeplearning4j-nlp</artifactId>
            <version>${dl4j.version}</version>
        </dependency>

        <dependency>
            <groupId>org.nd4j</groupId>
            <artifactId>${nd4j.backend}</artifactId>
            <version>${nd4j.version}</version>
        </dependency>

        <dependency>
            <groupId>com.tensquare</groupId>
            <artifactId>tensquare_common</artifactId>
            <version>1.0-SNAPSHOT</version>
        </dependency>

    </dependencies>
```

- 在application.yml中增加配置

```
ai:
  #词向量模型文件(名字随便取)
  vecModel: E:/source/tensquare/article.vecmodel
```

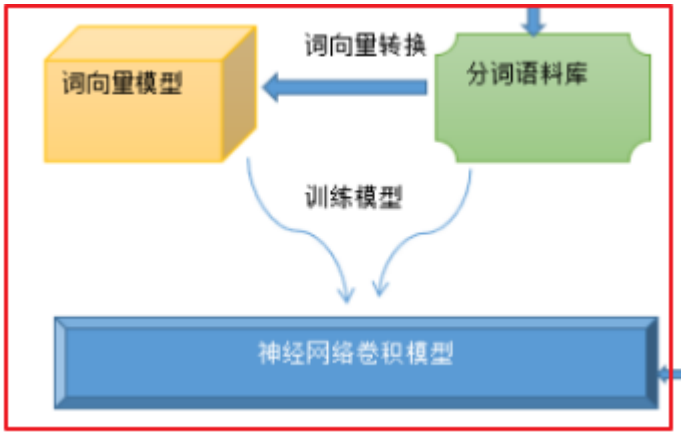
- 拷贝词向量工具类VecBuildUtils到tensquare_ai
- 修改服务类Word2VecService，增加方法用于构建词向量模型

```
    /**
     * 构建词向量
     */
    public void build(){
        VecBuildUtils.build(wordLib,vecModel);
    }
```

- 修改任务类TrainTask

```
@Component
public class TrainTask {

    @Autowired
    private Word2VecService word2VecService;

    @Scheduled(cron = "50 41 10 * * ?")
    public void trainModel(){
        //1. 合并分词
        //System.out.println("合并词料库开始...");
        //word2VecService.mergeWord();
        //System.out.println("合并词料库结束...");

        //2.构建词向量
        System.out.println("词向量构建开始...");
        word2VecService.build();
        System.out.println("词向量构建结束...");

    }
}
```

3构建卷积神经网络模型（CNN）

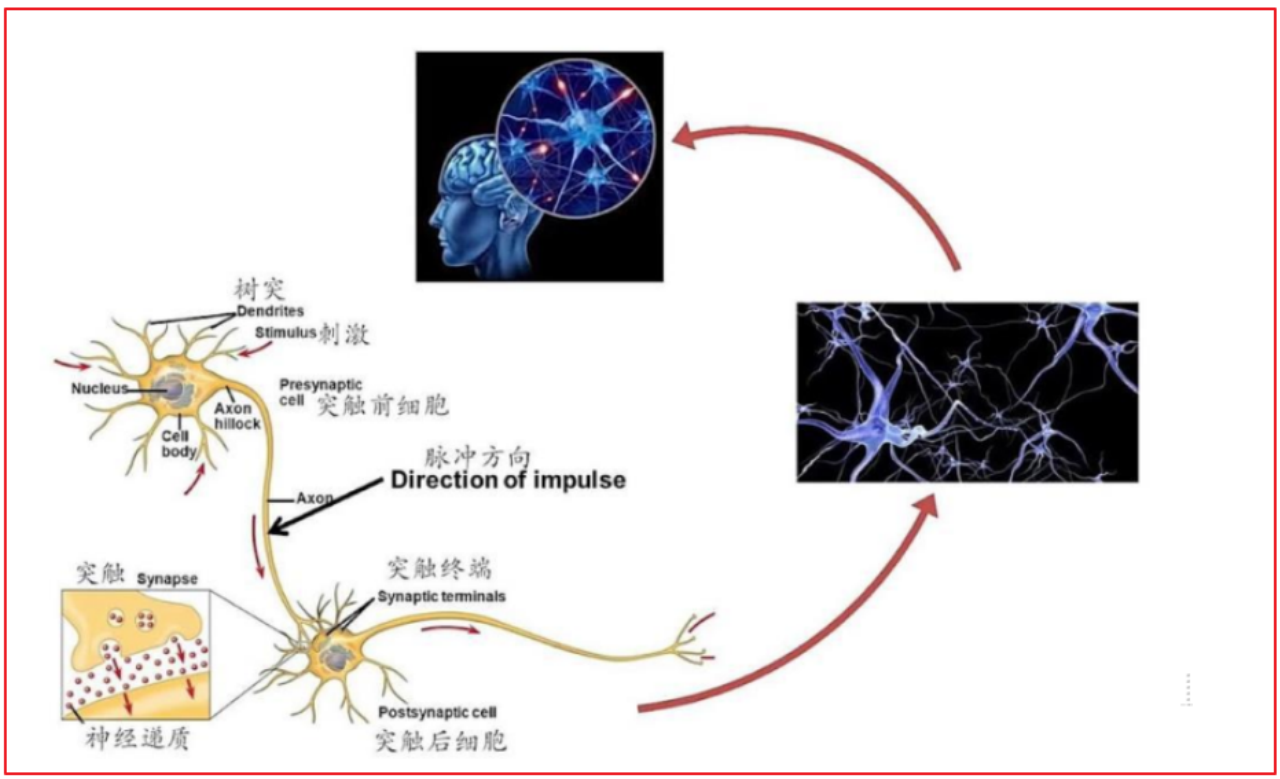


3.1卷积神经网络模型

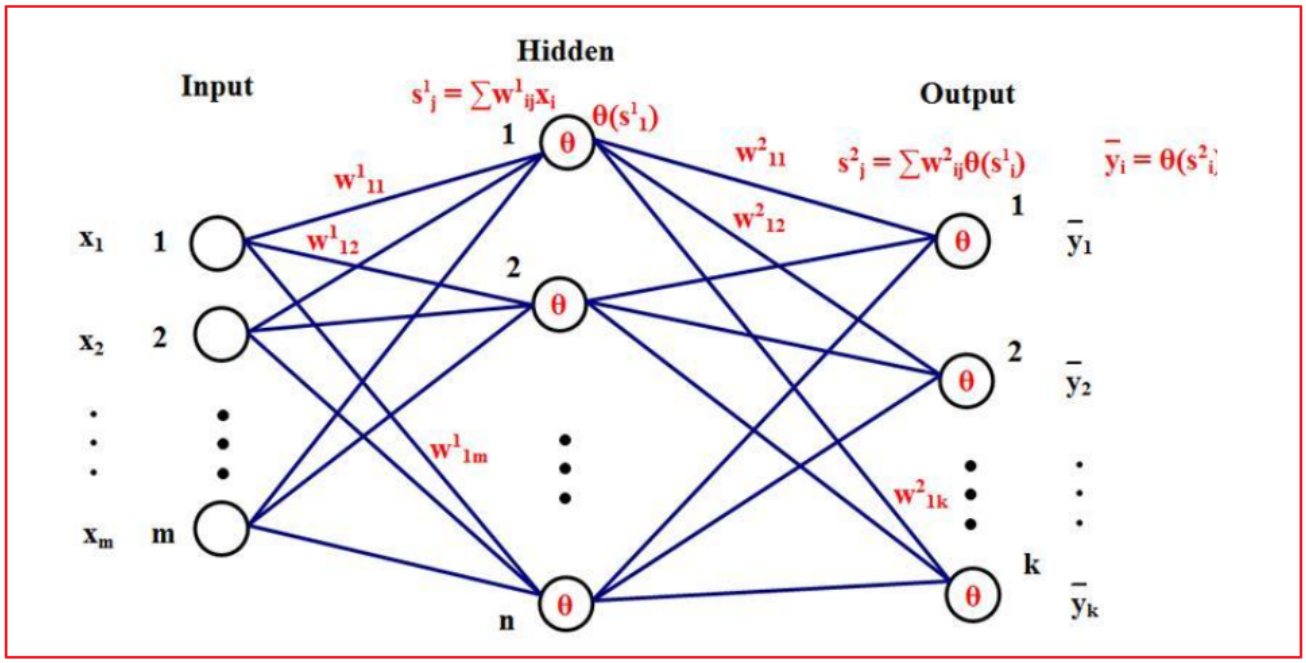
3.1.1神经网络

神经网络可以指向两种，一个是生物神经网络，一个是人工神经网络。

- 生物神经网络: 一般指生物的大脑神经元，细胞，触点等组成的网络，用于产生生物的意识，帮助生物进行思考和行动.一个神经元可以通过轴突作用于成千上万的神经元，也可以通过树突从成千上万的神经元接受信息。上级神经元的轴突在有电信号传导时释放出化学递质，作用于下一级神经元的树突，树突受到递质作用后产生出电信号，从而实现了神经元间的信息传递。化学递质可以使下一级神经元兴奋或抑制。

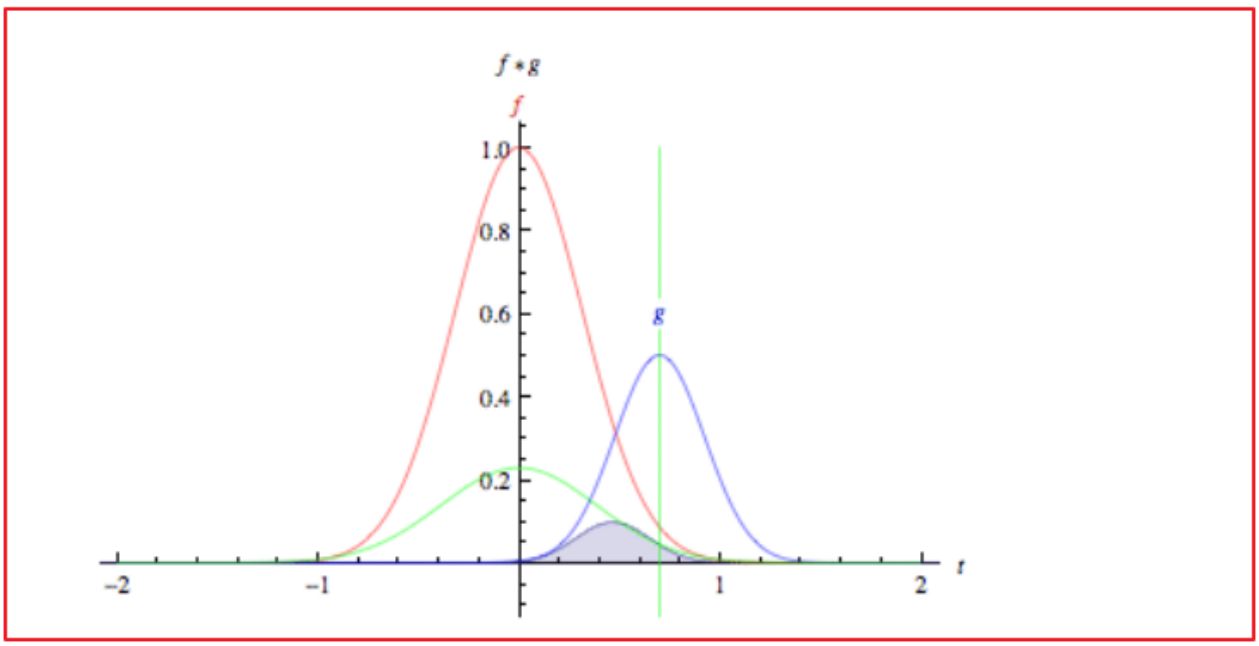


- 人工神经网络:（Artificial Neural Networks，简写为ANNs）也简称为神经网络（NNs）或称作连接模型（Connection Model），它是一种模仿动物神经网络行为特征，进行分布式并行信息处理的算法数学模型。这种网络依靠系统的复杂程度，通过调整内部大量节点之间相互连接的关系，从而达到处理信息的目的.是一种应用类似于大脑神经突触联接的结构进行信息处理的数学模型。在工程与学术界也常直接简称为“神经网络”或类神经网络。



3.1.2卷积

英文中的 to convolve 词源为拉丁文 convolvere，意为“卷在一起”。从数学角度说，卷积是指用来计算一个函数通过另一个函数时，两个函数有多少重叠的积分。卷积可以视为通过相乘的方式将两个函数进行混合



我么通过一个故事了解卷积:

比如说你的老板命令你干活，你却到楼下打台球去了，后来被老板发现，他非常气愤，扇了你一巴掌（注意，这就是输入信号，脉冲），于是你的脸上会渐渐地（贱贱地）鼓起来一个包，你的脸就是一个系统，而鼓起来的包就是你的脸对巴掌的响应，好，这样就和信号系统建立起来意义对应的联系。下面还需要一些假设来保证论证的严谨：假定你的脸是线性时不变系统，也就是说，无论什么时候老板打你一巴掌，打在你脸的同一位置（这似乎要求你的脸足够光滑，如果你说你长了很多青春痘，甚至整个脸皮处处连续处处不可导，那难度太大了，我就无话可说了哈哈），你的脸上总是会在相同的时间间隔内鼓起来一个相同高度的包来，并且假定以鼓起来的包的大小作为系统输出。

好了，那么，下面可以进入核心内容——卷积了！如果你每天都到地下去打台球，那么老板每天都要扇你一巴掌，不过当老板打你一巴掌后，你5分钟就消肿了，所以时间长了，你甚至就适应这种生活了.....如果有一天，老板忍无可忍，以0.5秒的间隔开始不间断的扇你的过程，这样问题就来了，第一次扇你鼓起来的包还没消肿，第二个巴掌就来了，你脸上的包就可能鼓起来两倍高，老板不断扇你，脉冲不断作用在你脸上，效果不断叠加了，这样这些效果就可以求和了，结果就是你脸上的包的高度随时间变化的一个函数了（注意理解）；如果老板再狠一点，频率越来越高，以至于你都辨别不清时间间隔了，那么，求和就变成积分了。可以这样理解，在这个过程中的某一固定的时刻，你的脸上的包的鼓起程度和什么有关呢？和之前每次打你都有关！但是各次的贡献是不一样的，越早打的巴掌，贡献越小，所以这就是说，某一时刻的输出是之前很多次输入乘以各自的衰减系数之后的叠加而形成某一点的输出，然后再把不同时刻的输出点放在一起，形成一个函数，这就是卷积，卷积之后的函数就是你脸上的包的大小随时间变化的函数。本来你的包几分钟就可以消肿，可是如果连续打，几个小时也消不了肿了，这难道不是一种平滑过程么？反映到剑桥大学的公式上，f(a)就是第a个巴掌，g(x-a)就是第a个巴掌在x时刻的作用程度，乘起来再叠加就ok了

3.2代码实现

- 修改配置文件application.yml

```
ai:
  #卷积神经网络模型
  cnnModel: F:/workspace/sk/sz/ai/article.cnnmodel
```

- 拷贝卷积神经网络模型工具类CnnUtil 到tensquare_ai
- 创建业务类CnnService

```
@Service
public class CnnService {

    //语料库目录
    @Value("${ai.dataDir}")
    private String dataDir;

    //词向量模型
    @Value("${ai.vecModel}")
    private String vecModel;

    //卷积神经网络模型
    @Value("${ai.cnnModel}")
    private String cnnModel;


    //构建卷积神经网络模型
    public void buildCnn(){
        String[] childPath = new String[]{"ai","web","db"};
        CnnUtil.build(vecModel,dataDir,childPath,cnnModel);
    }

}
```

- 修改任务类TrainTask

```
package com.tensquare.ai.task;

import com.tensquare.ai.service.CnnService;
import com.tensquare.ai.service.Word2VecService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.scheduling.annotation.Scheduled;
import org.springframework.stereotype.Component;

@Component
public class TrainTask {

    @Autowired
    private Word2VecService word2VecService;

    @Autowired
    private CnnService cnnService;

    @Scheduled(cron = "40 31 10 * * ?")
    public void train() {
        //System.out.println("合并词料库开始...");
        //word2VecService.mergeWord();
        //System.out.println("合并词料库结束...");

        //System.out.println("词向量构建开始...");
        //word2VecService.buildVec();
        //System.out.println("词向量构建结束...");

        System.out.println("卷积神经网络模型构建开始...");
        cnnService.buildCnn();
        System.out.println("卷积神经网络模型建结束...");

    }

}
```

第四章-实现智能分类

1.需求分析

传入文本，得到所属分类信息

2.代码实现

- 修改CnnService，增加方法

```
    /**
     * 预测归类
     * @param content
     * @return
     */
```

```
    public Map  predictions(String content){
        Map<String, Double> map = null;
        try {
            content = IKUtil.split(content," ");
            String[] childPaths =  new String[]{"ai","db","web"};
            map = CnnUtil.predictions(vecModel,cnnModel,dataPath,childPaths,content);
        } catch (IOException e) {
            e.printStackTrace();
        }
        return map;
    }
```

- 创建AiController

```
@RestController
@RequestMapping("/ai")
public class AiController {
    @Autowired
    private CnnService cnnService;
    @RequestMapping(value = "/predictions",method = RequestMethod.POST)
    public Map predictions(@RequestBody Map<String,String> map){
        return cnnService.predictions(map.get("content"));
    }

}
```

- 使用postman测试 http://localhost:8080/ai/predictions 提交格式

```
{
    "content":"编写一个网页文件，使用一个表单让用户填写购货订单。填写的信息包括姓名、电话、商品名称、单价、数量和金额。当提交表单时，要求：(1)商品名称和单价只能让用户选择；"
}
```