你好，欢迎来到第 24 课时，这是我们的最后一节实践课，也是我们的数据挖掘思维与实战的最后一节正课。在这节课中，我将为你讲解数据挖掘在自然语言处理领域最典型的应用——文本分类，并带领你一步步解决文本分类的问题。话不多说，让我们开始课程吧。

fastText 算法

这里我们先简单介绍一下 fastText，因为我们在前面没有提到过这个算法，你可能有点疑惑这是个什么东西。fastText 与我们上一课时介绍的 Word2Vec 的 CBOW 方法十分类似，可以说是在 Word2Vec 的基础上探索出来的。它的结构与 Word2Vec 非常类似，也是一个浅层神经网络模型，有一个输入层、一个隐藏层和一个输出层，其中输入的也是文本的向量。

当然了，它们也有很多区别。首先，fastText 是一个分类算法，是有监督的，输出层输出的是一个分类标签。另外，在输入层，除了词本身的向量，还增加了 n-gram 方法来把词划分成若干个子词。比如说设置 n-gram 为 2，“拉勾教育” 可以分成 “拉勾”“勾教”“教育”，这些子词的向量也加入输入向量中，这样对于低频词更友好一些，它们的字符级别 n-gram 可以与其他词共享信息量，并且在新的数据集上面，对于词库中没出现过的词，可以使用 n-gram 来为它们叠加向量。

除了这些，fastText 的训练非常迅速，虽然是浅层神经网络，但是分类效果可以接近深度神经网络。使用 BERT（流行的深度神经网络自然语言处理算法）要花几十个小时才能训练完的任务，使用 fastText 可能只需要几分钟。

关于 fastText 的简单介绍就先说到这里，接下来我们进入数据挖掘的环节。

理解业务与理解数据

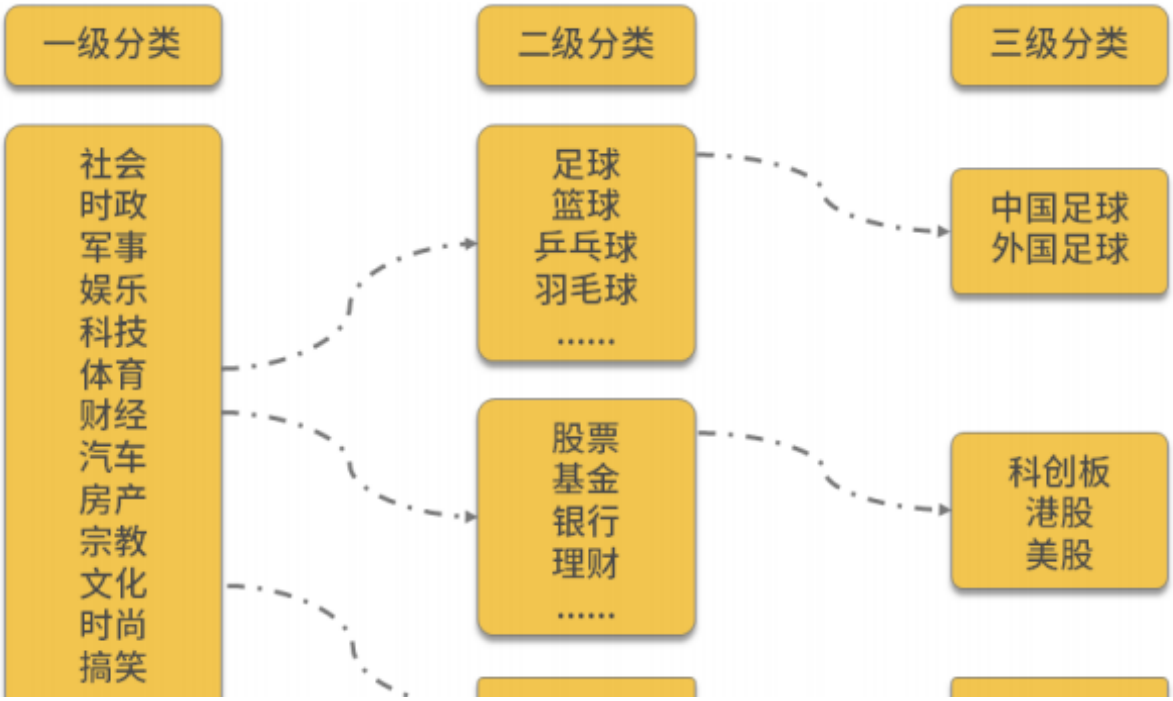
新闻会涉及方方面面的内容，是一种泛领域的内容信息。要给新闻分类，我们首先要有一套分类体系。我们期望能够构建一套完整的、可以覆盖所有新闻内容的分类体系。这个分类体系有着**自上而下的结构**，由粗粒度到细粒度一层一层逐渐展开。

就如下图展示的，我们需要构建一个有三个层级的**分类体系**。

一级分类粒度最粗，覆盖面也最广，每一个分类标签都囊括了一大波内容，几乎所有的内容都可以被分为其中的一个或多个类别。

二级分类标签相对较细，是在一级标签的基础上进行的扩充，比如说在体育下面又细分成足球、篮球、乒乓球、羽毛球等二级分类标签。二级标签开始，就可能存在覆盖不全面的问题，有些新闻可能无法找到合适的二级分类标签为其标注，所以我们在每一个组二级分类下面增设一个 “其他” 类别。

三级分类又在二级分类的基础上进行了扩充，粒度更加细致。相应的，三级分类也都增设 “其他” 类别。



很明显，我们的分类体系是多层级、多标签、多分类的，在具体的实现上，很难依靠单一的算法直接给出结果。当然，在动手去做之前我们还需要对数据有一些了解。

对于新闻，我们多多少少都有些了解。通常来说，一篇新闻都会有一个标题、一篇正文，还有发布时间和发布源。在正文中，可能会有一些配图，这些是新闻的基本内容。对于新闻最重要的正文部分，一般也都会有相对比较固定的写作格式，以及相对标准的新闻内容。比如事件的起因、经过、结果、重要人物、发生的时间、地点等要素。这些信息在新闻中占据了很重要的信息量，在做分类的时候，如果能够恰当地提取和使用这些信息，将会使你的准确率大大提升。

接下来我们就看一下，如何去构建一套完整的数据处理和模型预测体系。

准备数据与模型训练

要想使用 fastText 对新闻文本进行分类，我们需要对数据做一些预处理。让我们先来看一个简单版本的分类。

这是 fastText 的官方地址：https://github.com/facebookresearch/fastText

这里讲解了如何使用 fastText 的基本内容。需要注意的是，fastText 需要用到 C++ lib 的支持，如果你的电脑上没有安装 C++ lib 库，可能会报错。

下面是官网介绍的安装方法：

```
$ wget https://github.com/facebookresearch/fastText/archive/v0.9.2.zip

$ unzip v0.9.2.zip

$ cd fastText-0.9.2

$ make
```

完成了 fastText 的安装之后，可以直接使用命令来进行模型的训练。我们前面说了，fastText 除了可以进行文本分类以外，还可以用来获取中间产物 Word2Vec 的词向量表。
如果要使用 fastText 获得词向量，可以使用下面的语句，其中的 data.txt 是我们需要输入的训练语料。训练完成后会获得两个文件 “model.bin” 和“model.vec”。其中 bin 文件是用来存储模型参数以用来追加训练的，vec 文件是词向量文件。

```
$ ./fasttext skipgram -input data.txt -output model
```

如果要追加训练，可以使用下面的语句，queries.txt 是新的文本文件，其中可能包含了模型中不存在的单词。

```
$ ./fasttext print-word-vectors model.bin < queries.txt
```

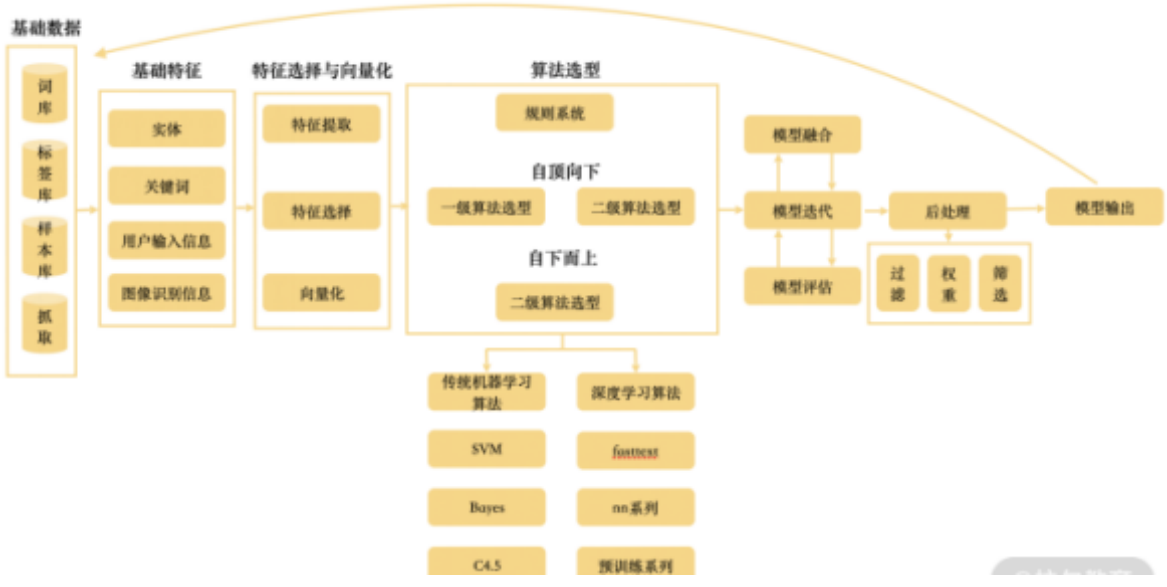
对于文本分类，我们则需要使用下面的语句了，这里需要注意的是，train.txt 文件是我们的训练语料，它需要被处理成固定的格式，其中每一行是一条训练数据，在每一行的开头加入数据的标签，并在标签上加入前缀 “**label**”，多个标签之间使用空格隔开。比如说我们的新闻标签 “科技”“娱乐”，就写成 “**label**科技 **label**娱乐”。

```
$ ./fasttext supervised -input train.txt -output model
```

在训练完成后，我们仍然会得到与上面相同的文件，只不过这次 bin 文件可以用来对新数据进行预测了。除了直接使用语句来进行训练，我们当然也可以在 Python 中使用 fastText 算法包来进行模型的训练。

一个完整的大型分类模型迭代流程

当然了，在我们实际的生产环境中，问题往往没有这么简单，多级多标签多分类要想获得还不错的结果，需要对各个环节进行多种尝试和处理。在下面，我画了一张关于整个数据处理和模型训练的迭代流程图，基本上涵盖了我们整个算法模型的处理过程，让我们来看一下其中都有哪些步骤。



基础数据层

首先，在最左侧的是我们的基础数据，前面我提到过，新闻中有很多特征载有非常重要的信息，比如说人物、地点、时间等，这些都属于词库的范畴，如果我们有比较完善的词库，其实会节约很多时间并取得较好的效果。

其次是我们的标签库，这个不需要过多解释，这个库就是来维护我们需要进行学习和预测的标签。样本库对我们的样本进行管理，在这里就是存储了我们的新闻信息以及我们所需要用来训练的相关数据。

最后一个抓取则是一种手段，有些时候我们自己的数据可能存在着各种各样的问题，需要借助一些外部的数据来支持我们的工作，那么就可能会用到抓取的手段。

基础特征

完成了基础数据的构建，下一步就开始构建一些基础特征，这一环节还可能包括对数据的分析等。

在这个环节，我们也需要用到很多算法或者策略。比如说对于文本进行分析，提取其中的关键词、实体词，对图像信息进行识别；再比如识别其中的人物或者场景，识别图像的质量，等等。

在这一步，我们把很多原始的数据进行了算法加工，从中提炼出一些信息含量更高的特征信息，以方便下一步有针对性的特征选择与向量化。

特征选择与特征向量化

在特征选择与向量化环节，主要是针对后续可能会用到的模型来进行针对性的处理。比如说对于 SVM 算法，那我们可能需要对关键词信息进行加工以适配 SVM 算法；而对于 fastText 算法，相对比较简单，可以只用分词后的结果，当然也可以在其中拼接上重要性更高的实体词等信息，来加强学习的效果。

在这一步，我们要针对后面即将实施的算法实验来针对性地选择特征，并把特征构建成算法可以接收的形式。

算法选型

紧接着，我们就进入了算法选型的步骤。对于多级分类，我们可以采用**自顶向下的方法**，先进行最高级的分类，确定一级分类，再去构建一个模型来处理二级分类，以此类推。

也可以采用**自底向上的方法**，先使用一个大分类器或者若干个二分类器，去预测所有的最底层分类，然后再向上总结。

当然，除了这两个方案，还有很多可以处理模型融合的方法。除此以外，对于有众多分类标签的分类器，我们也可以叠加一层规则系统，来对一些特殊的情况进行修正和处理。

在这一步，我们还需要对各种算法的效果进行对比，并确定我们到底要使用什么样的算法模型来解决其中的每一个问题。

模型评估与优化

与我们的算法选型十分紧密的下一个环节，就是整个评估和迭代环节。其实我们每一个算法模型，都会有相应的准确率、召回率等指标来评估模型的效果。同时，根据效果的反馈，我们来决策该如何融合各个模型，并有针对性地来进行模型迭代。

比如说，如果我们发现 “娱乐” 下面有一个三级分类为 “动漫”，它的分类结果与“文化” 下面的 “漫画” 很难进行区分，这时候就需要考虑是否要把这两个合成为一个标签，或者是否有什么特征可以区分这两个内容。

修饰处理

解决了上面的细节问题，我们就可以对模型的结果来进行一些最终的修饰处理了。一个大型的分类体系可能不是通过一次训练就能解决所有问题，但是不能因为部分问题还没有解决而无限期地进行迭代，我们的结果最终还是要落地到项目上，发挥实际的价值。

所以我们可能需要一个过滤和筛选，暂时屏蔽那些还没有解决的问题。这样当我们在后面的迭代工作中，只要解决了这个问题，再对这个修饰环节进行简单的调整，就可以重新上线了。

模型部署与结果保存

这样，我们的分类模型就可以进行部署应用，来实现算法的输出了。当然，在输出的过程中，我们还需要有一些机制来收集输出的结果以及 badcase，我们把这些东西积累到基础数据中，在经过一段时间之后，就可以有更多的数据来支撑我们的模型迭代了。

总结

至此呢，我们最后一节实践课就讲完了。这一课时，我们以一个多层级的新闻文本分类为背景，先简单介绍了 fastText 模型；然后介绍了一种使用 fastText 来进行分类的简单案例；最后，我们引入了在实际的大型项目中，到底是如何进行多级分类的流程。

可以看到，在这个过程中，算法只是其中的一个环节，要想具体地解决业务中的实际问题，我们还需要做很多很多的工作。

那么我们的课程到这里也就结束了，非常感谢你能够坚持看完这门课程。在写作的过程中，由于各种各样的因素，内容难免存在一些错误和疏漏，如果你发现了问题，欢迎及时批评与指正。

最后希望你将还没有掌握的内容，回过头再详细看一下。我希望这 24 个课时的分享，可以为你解决一些困惑，帮助你在数据挖掘领域走得更长远。再见！