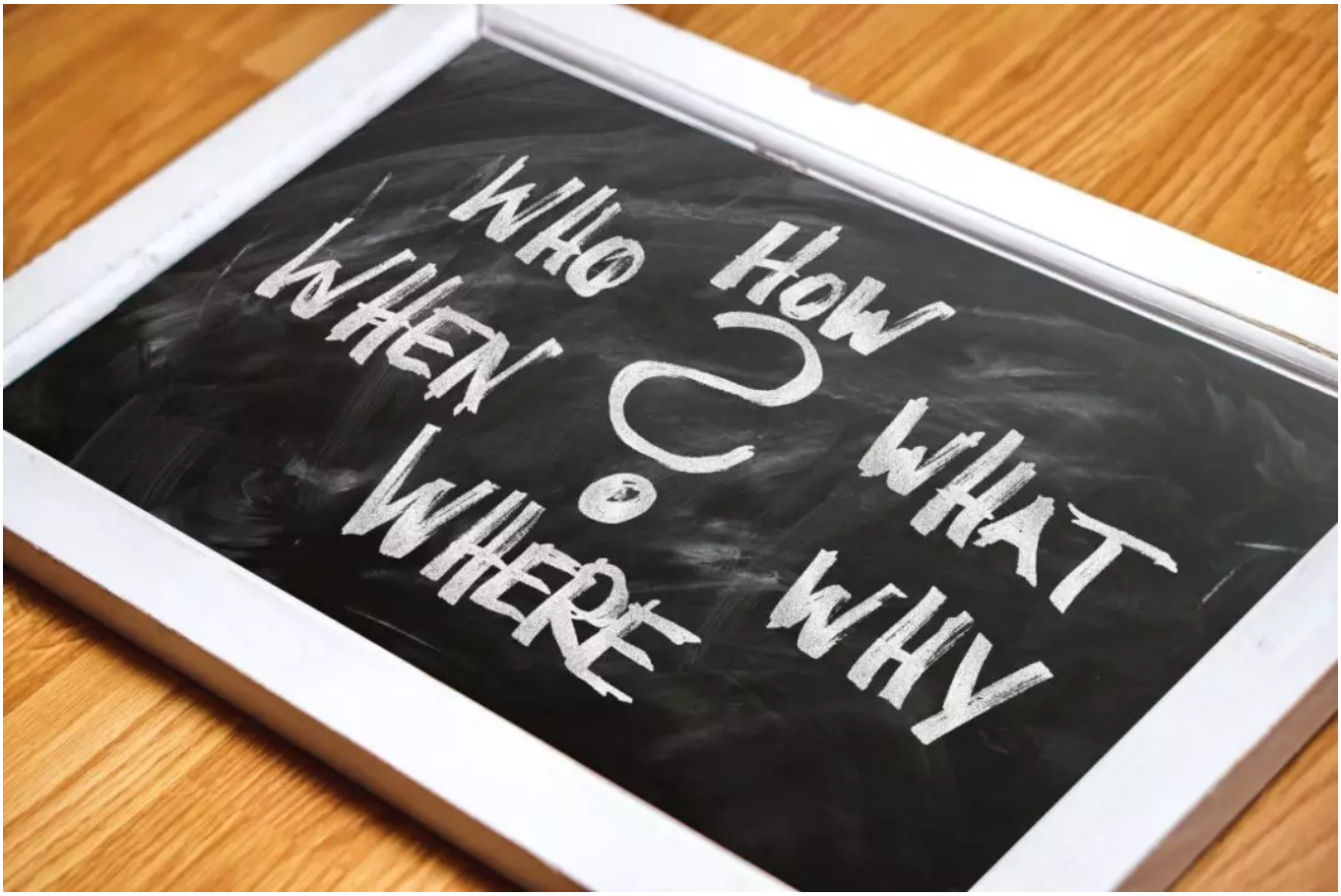
机器学习|八大步骤解决90%的NLP问题

Emmanuel Ameisen 大邓和他的Python 2018-05-13



在现实生活中，文本信息无处不在。理解并学习文本数据的内在涵义一直是一个非常活跃的研究课题，这就是自然语言处理。

对于企业而言，利用文本数据可以对新产品的功能进行验证、改进并扩展。在这样的实际应用中，有三大类自然语言处理任务最为常见：

- 识别不同的用户/客户群（如预测客户流失量、生命周期价值、产品偏好）
- 准确地检测和提取不同类别的反馈信息（如正面和负面的评论/意见、衣服尺寸等特定属性的提及频率）
- 根据用户的意图对文本信息进行分类（如请求基本帮助、紧急问题）

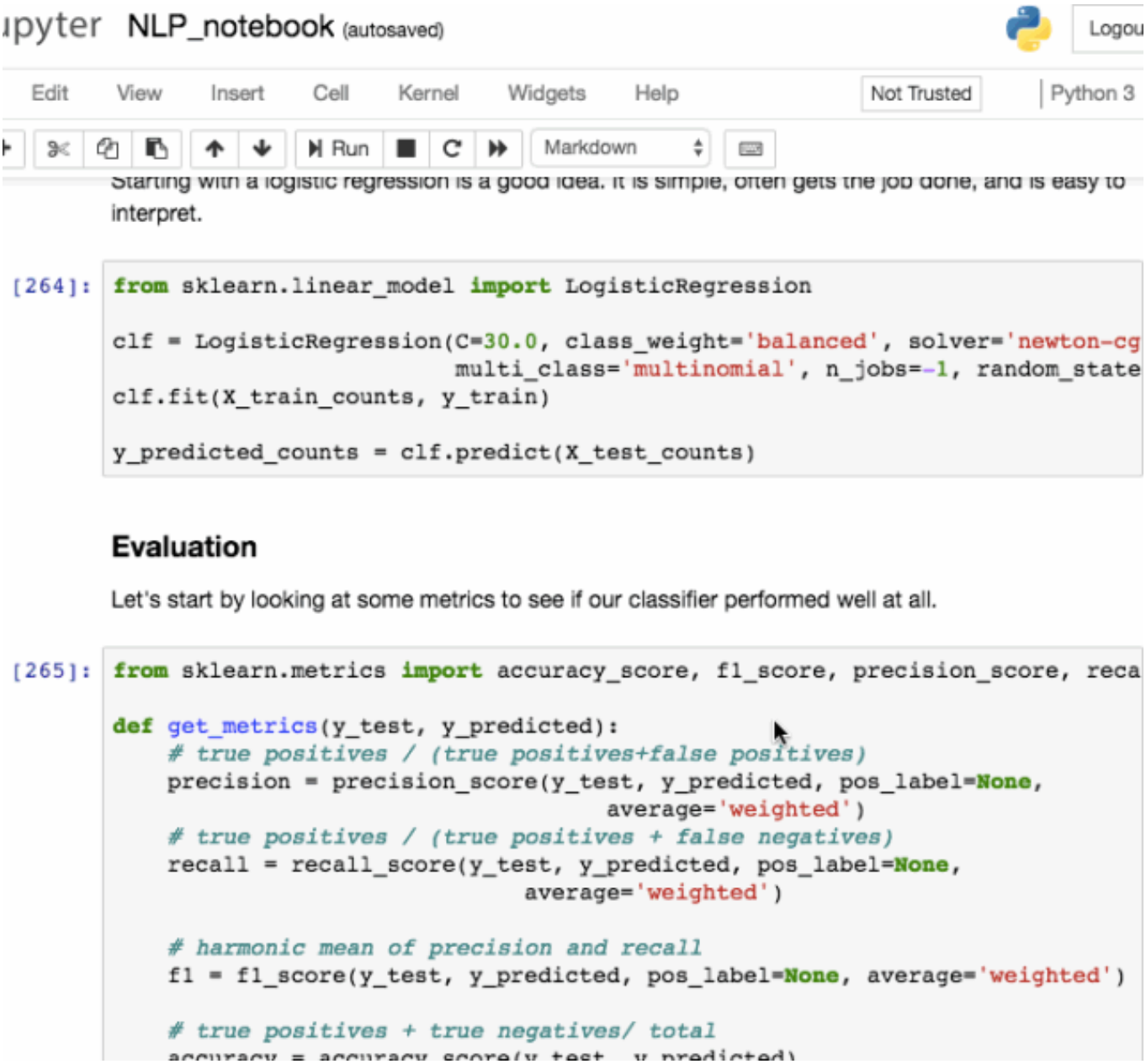
尽管自然语言处理领域有很多在线的论文和教程资源，但很少有一些比较高效的指引和提示，以方便我们快速上手并解决这里的问题。这正是我们整理出本文的目的。

在本文中，我们将分八大步骤来介绍如何用机器学习处理文本数据。我们将从最简单的方法开始，逐一讲解，然后分析更具体的方案细节，如特征工程、词向量和深度学习。你可以把本文看作是标准方法的高度概括。

读完本文后，你将学会：

- 如何收集，准备和检查数据
- 如何建立简单的模型，有必要的话，应构建深度学习模型
- 如何解释并理解你的模型，以确保模型学习的是特征信息而不是噪音

此外，文末还附有一个交互式notebook笔记，让你可以随意运行本文中的代码，来演示和应用其中的技术，特别是一些比较抽象的概念。



Step 1: 收集数据

每一个机器学习问题都始于数据，比如一组邮件、帖子或是推文。文本信息的常见来源包括：

- 电商评论（来自 Amazon、Yelp 以及其他电商平台）
- 用户产出的内容（推文、Facebook 的帖子、StackOverflow 的提问等）
- 问题解决（客户请求、技术支持、聊天记录）

在这篇文章中，我们将使用 CrowdFlower 提供的一个数据集，名为“社交媒体中的灾难（Disasters on Social Media）”。

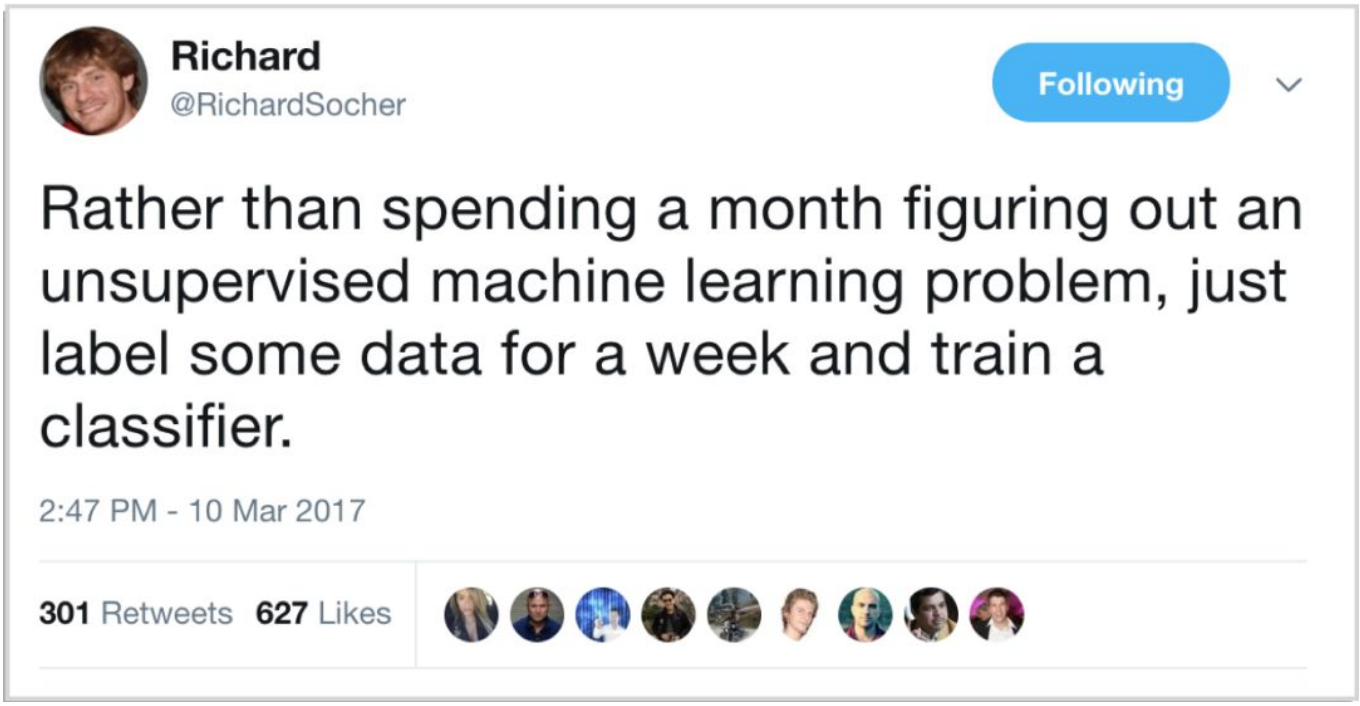
贡献者们查看了超过 10000 条具有类似“着火”、“隔离”、“混乱”等搜索关键词的推文，然后标记这个推文是否和灾难事件有关（与之相反的是一些玩笑、电影点评或是一些非灾难性的事件）。

我们的任务是分辨出哪些推文是真正和灾难事件相关的，而不是一些类似电影描述的不相关话题。为什么呢？一个潜在的应用是针对突发事件对执法人员进行专门的提醒，而不会被其他无关信息，比如 Adam Sandler 新上映的电影所干扰。这项任务中一个特别的挑战是这两种情况在搜索推文的时候都用到了相同的检索词，所以我们只能通过细微的差别去区分他们。

在下面的文章中，我们将把与灾难事件相关的推文称为“灾难”，将其他推文称为“不相关的”。

标签

我们已经标注过数据，所以知道推文是如何分类的。比起优化一个复杂的无监督学习方法，寻找和标记足够多的数据来训练模型会更加快捷、简单和廉价。



Step 2:数据清洗

数据科学家的一个必备技能是知道自己的下一步操作是处理模型还是数据。有一个好的经验法则是先观察数据然后进行数据清洗。一个干净的数据集能使模型学习到有意义的特征而不会被一些不相关的噪声影响。

可以借鉴下方的列表来进行数据清洗：（查看代码获取更多信息）

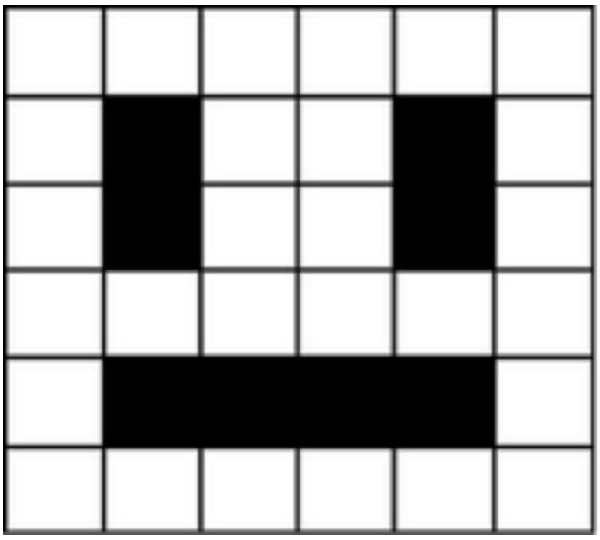
- 去除一切不相关的字符，比如任何非字母数字的字符
- 标记你的文本，将他们拆分为独立的单词
- 去除不相关的词语，比如 @这类提醒或是 url 链接
- 将所有字母转换成小写，这样“hello”，“Hello”，“HELLO”就会被当做同样的单词处理
- 将拼错的单词或是多种拼法的单词与某个特定的表达绑定（比如：“cool”/“kewl”/“cooool”）

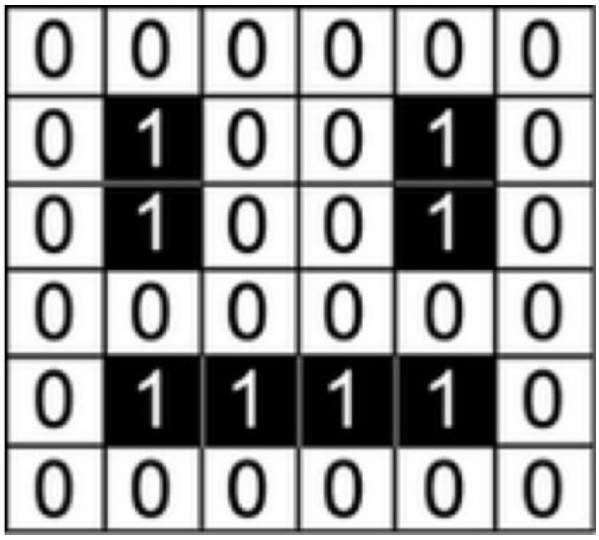
- 考虑词形还原（比如将“am”,“are”,“is”都看做“be”）

完成这些步骤并检查完其他错误后，我们就可以使用这些干净的、标记过的数据进行模型训练了！

Step 3:找到一种好的数据表达方式

机器学习模型通常以数值作为输入。我们这里的数据集是句子列表，为了让模型可以从数据中学到句子的特征模式，我们首先要找到一种方法来把它转换成模型能理解的形式，即数字列表。





One-hot 和 Bag of Words

通常为计算机解释文本的方法是将每一个字符都编为一个独立的数字（例如 ASCII 码），这个过程叫做**特征提取**。如果使用这种简单的表达来做分类器，需要我们的数据从头开始学习词语的结构，这对大多数数据集来说是很难实现的。所以我们需要一种更上层的方法。

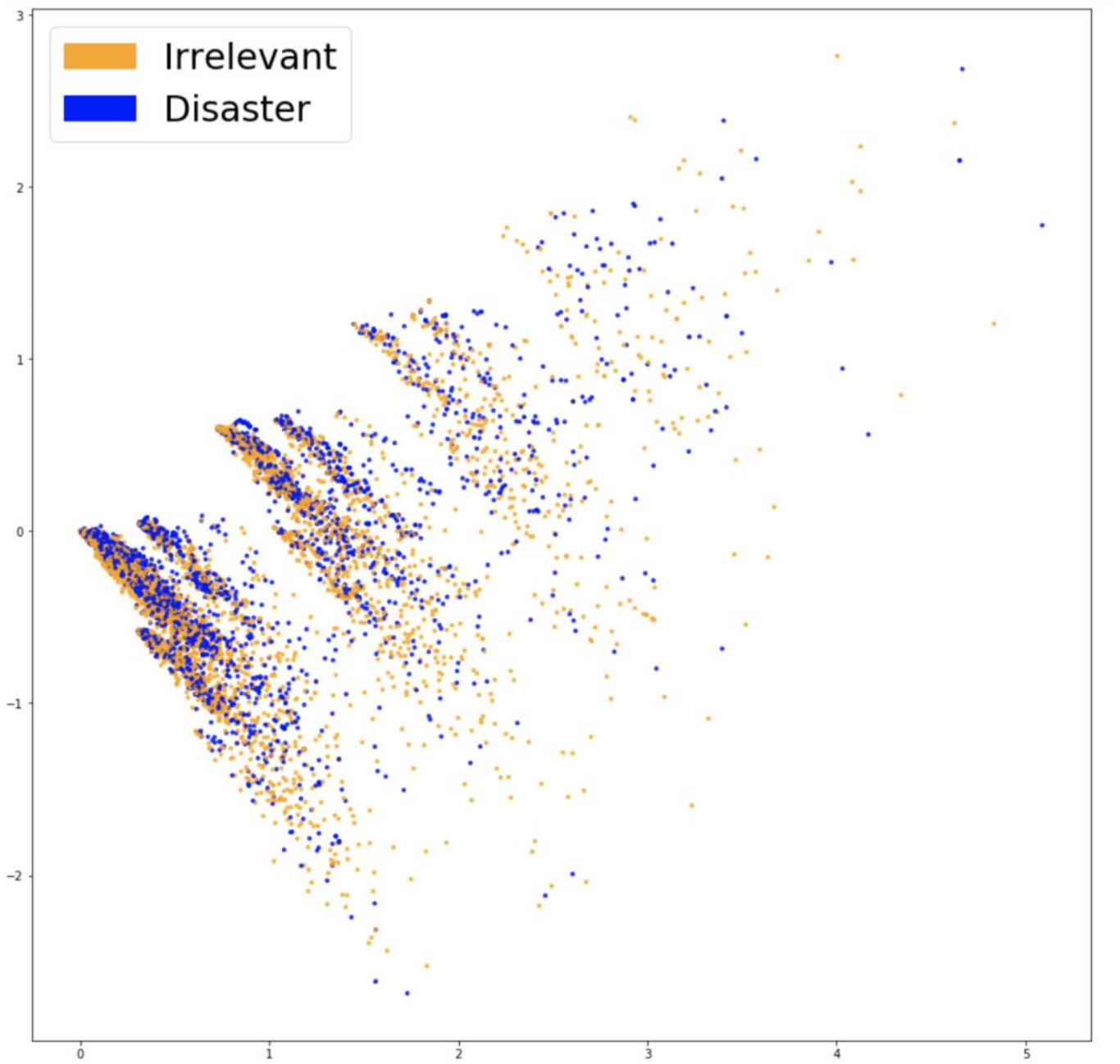
例如，我们可以为数据集中的所有单词制作一张词表，然后将每个单词和一个唯一的索引关联。每个句子都是由一串数字组成，这串数字是词表中的独立单词对应的个数。通过列表中的索引，我们可以统计出句子中某个单词出现的次数。这种方法叫做 词袋模型，它完全忽略了句子中单词的顺序。如下图所示：

用词袋模型表示句子。句子在左边，抽取到的特征在右边。向量中的每一个索引代表了一个特定的单词。

可视化

在“社交媒体中的灾难”样本词表中大概会有 20000 个单词，这意味着每句句子都会用一个长度为 20000 的向量来表示。向量的 大部分会被 0 填充，因为每句话只包含了词表中很小的一个子集。

为了看出工作是否真正抓住了和问题相关的信息（比如推文是否与灾难相关），有一个好方法是将它们可视化，然后观察结果是否有很好的分布。考虑到词表通常很大，而且用 20000 维的数据做可视化是基本不可能的，所以我们使用了 PCA 这种技术将数据降到二维。绘制如下：



两个分类看起来没有很好的分离，这可能是我们选择的嵌入方法的特征或是单纯因为维度的减少引起的。为了了解词袋这种特征提取方式是否会起一些作用，我们可以试着基于它训练一个分类器。

Step 4:分类

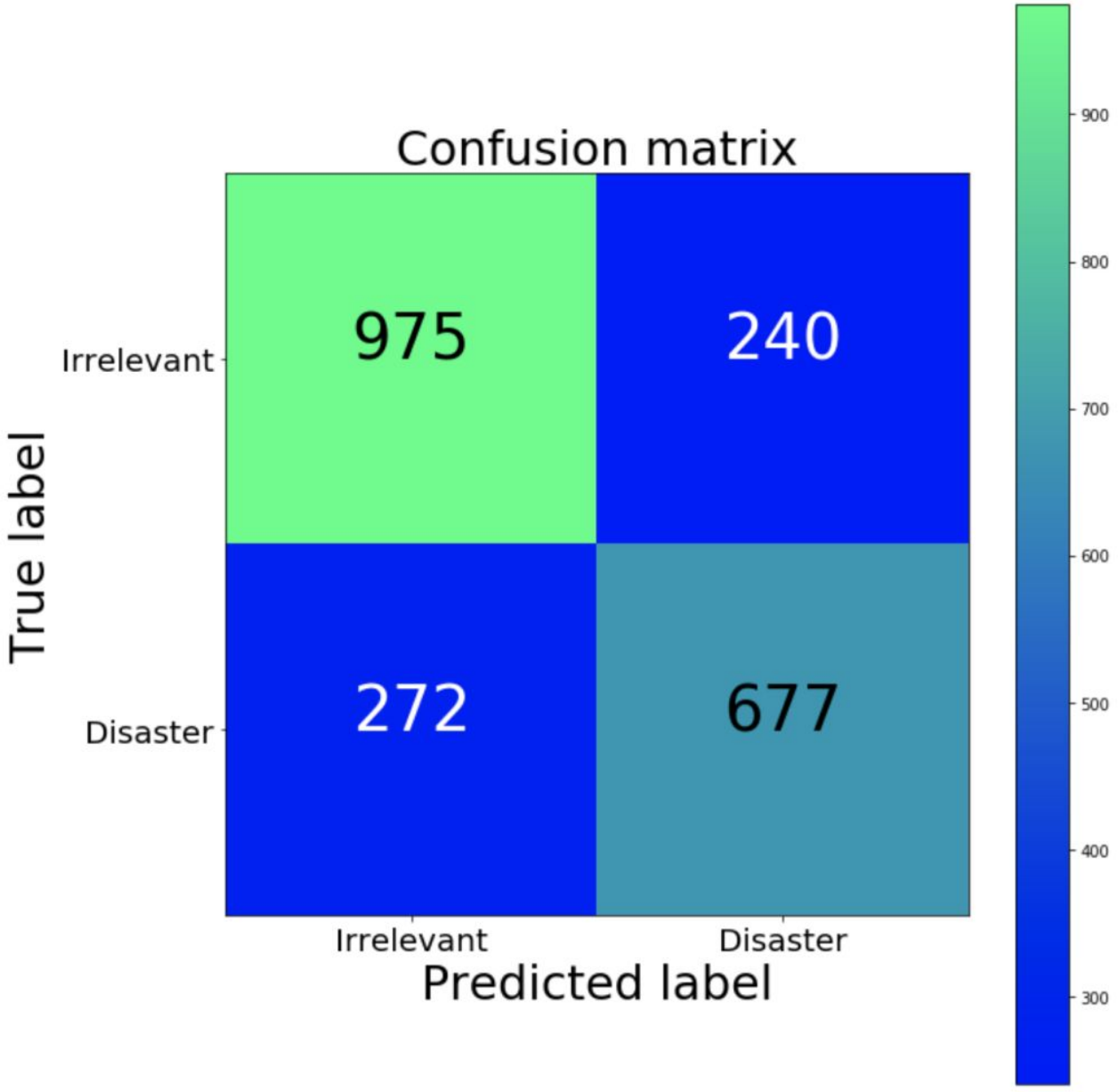
当初次接触一个问题，通常来说最好的方法是先挑选一个能解决问题的最简单的工具。当提到数据分类时，一般最受欢迎的是通用性和可解释性兼具的逻辑回归算法。这种算法很容易训练而且结果也是可解释的，你可以很轻松地从模型中提取出最重要的一些系数。

我们将数据分为两个集合，训练集用于匹配模型，测试集用于观察应用在未知数据上的效果。训练后我们得到了 75.4% 的精确度。结果还不错！推测出现最多的类（“不相关”）只能达到 57%。但是，即使是 75% 的精确度也已经足够好了，我们决不能在还没有理解模型的情况下就开始应用它。

Step 5: 检验分析效果

检验混淆矩阵

理解模型的第一步，是了解模型产生的错误分类，以及最不应该出现的错误。在我们的例子中，“误报”是指将不相关的推文分类为“灾难事件”，“漏报”是指将与灾难有关的推文归类为“与灾难无关的事件”。如果要优先处理潜在的灾难事件，那就要降低“漏报”。而如果资源受限，就要优先降低“误报”，减少错误的提醒。使用混淆矩阵可以很好地可视化这些信息，并将模型预测的结果与数据的真是标签进行比较。理想情况下，模型的预测结果与真实情况（人工标注）完全相符，这时候混淆矩阵是一条从左上角到右下角的对角矩阵。



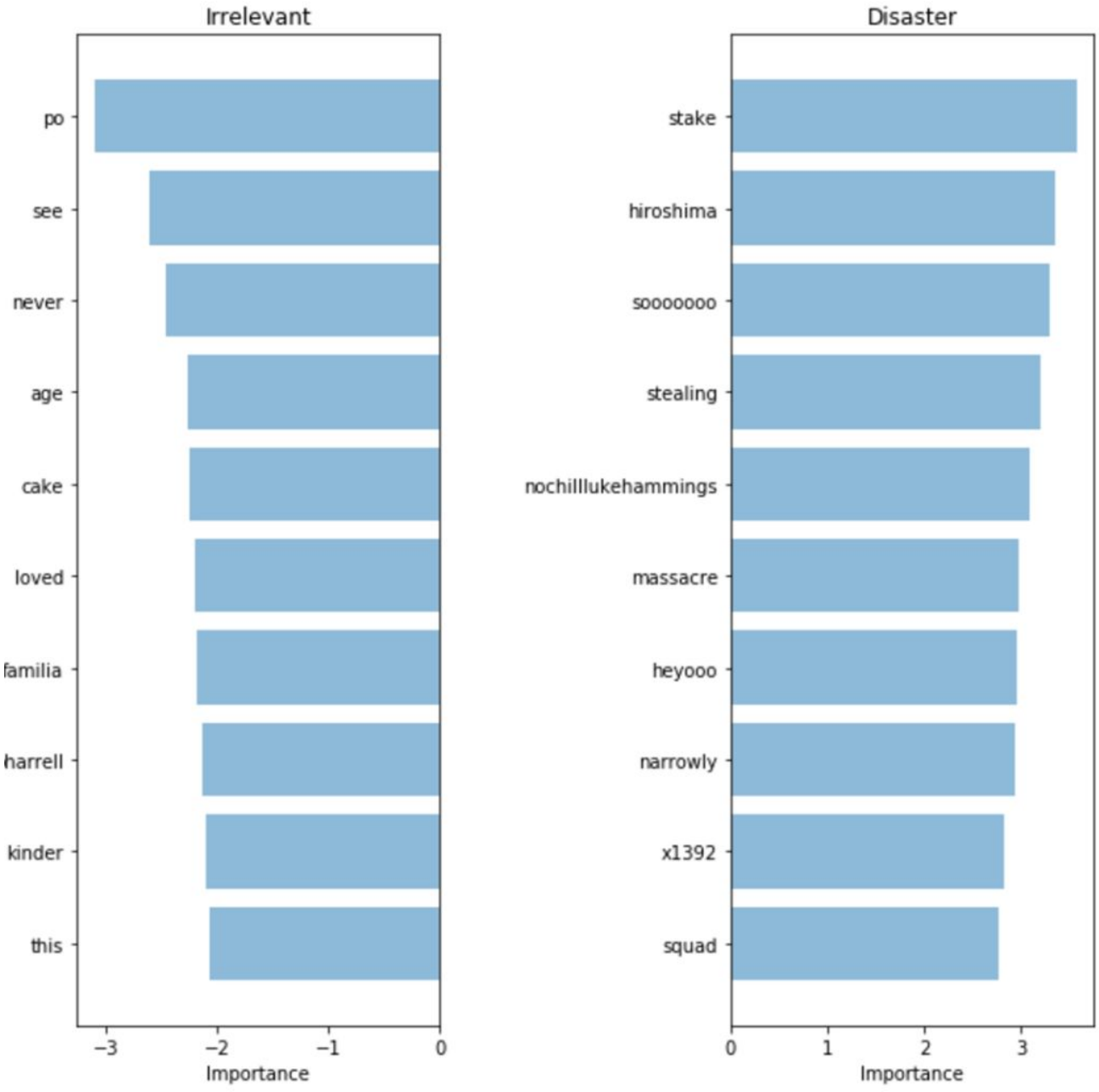
混淆矩阵（绿色部分所占比例较高，蓝色部分的比例较低）

相比假阳性结果，我们的分类器产生了更多的假阴性结果。换句话说，模型中最常见的错误是将灾难性推文错误归类为不相关推文。如果假阳性结果的执法成本很高的话，那么我们分类器的这种偏差就是良性的。

解释和说明模型

为了验证模型并解释它的预测结果，我们需要明确模型用以进行判断的那些词汇。如果我们的数据有偏差，而分类器在样本数据中却能做出准确预测，那这样的模型就无法在现实世界中很好地推广。

在这里，我们可以用图表来表示灾难性推文与不相关推文两类预测中最重要的词汇。由于我们可以对模型的预测系数进行提取和排序，用词袋模型(bag-of-words)和Logistic回归模型很容易就能计算出单词的重要性。



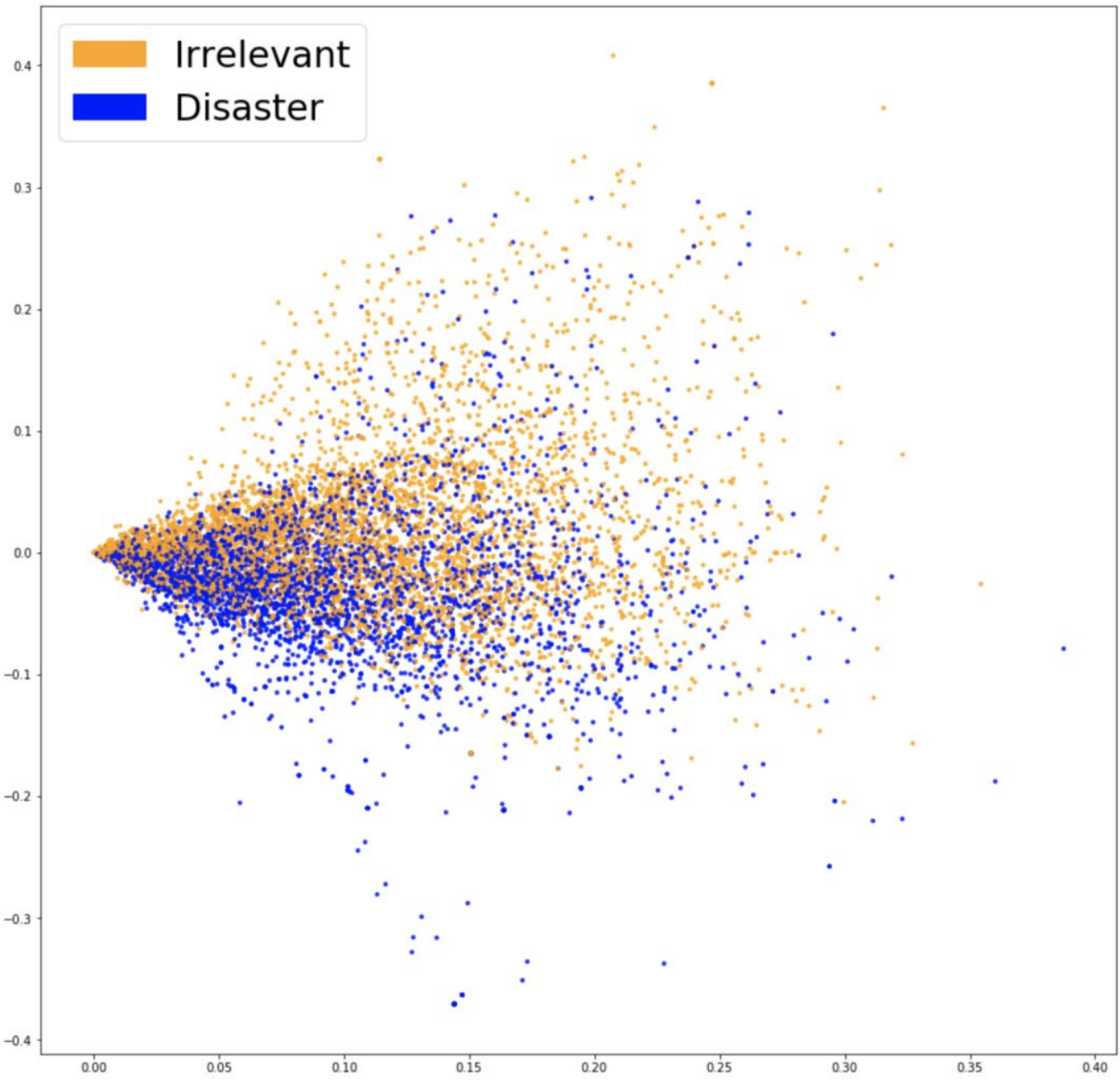
词袋(bag-of-words)：基于单词词频表示重要性

我们的分类器能够正确识别出一些模式（如广岛、大屠杀等），但在一些毫无意义的词汇（如heyoo、x1392等）上还是出现了过拟合。词袋模型（bag-of-words）仅能处理庞大词汇表内的不同词汇，并对所有的词汇分配相同的权重。然而，其中一些词汇出现得非常频繁，但却只是预测结果的噪音数据。接下来，我们将试着找到一种能够表示词汇在句子中出现频率的方法，尽量让模型从数据中获取更多的信号。

Step 6：词汇结构的统计

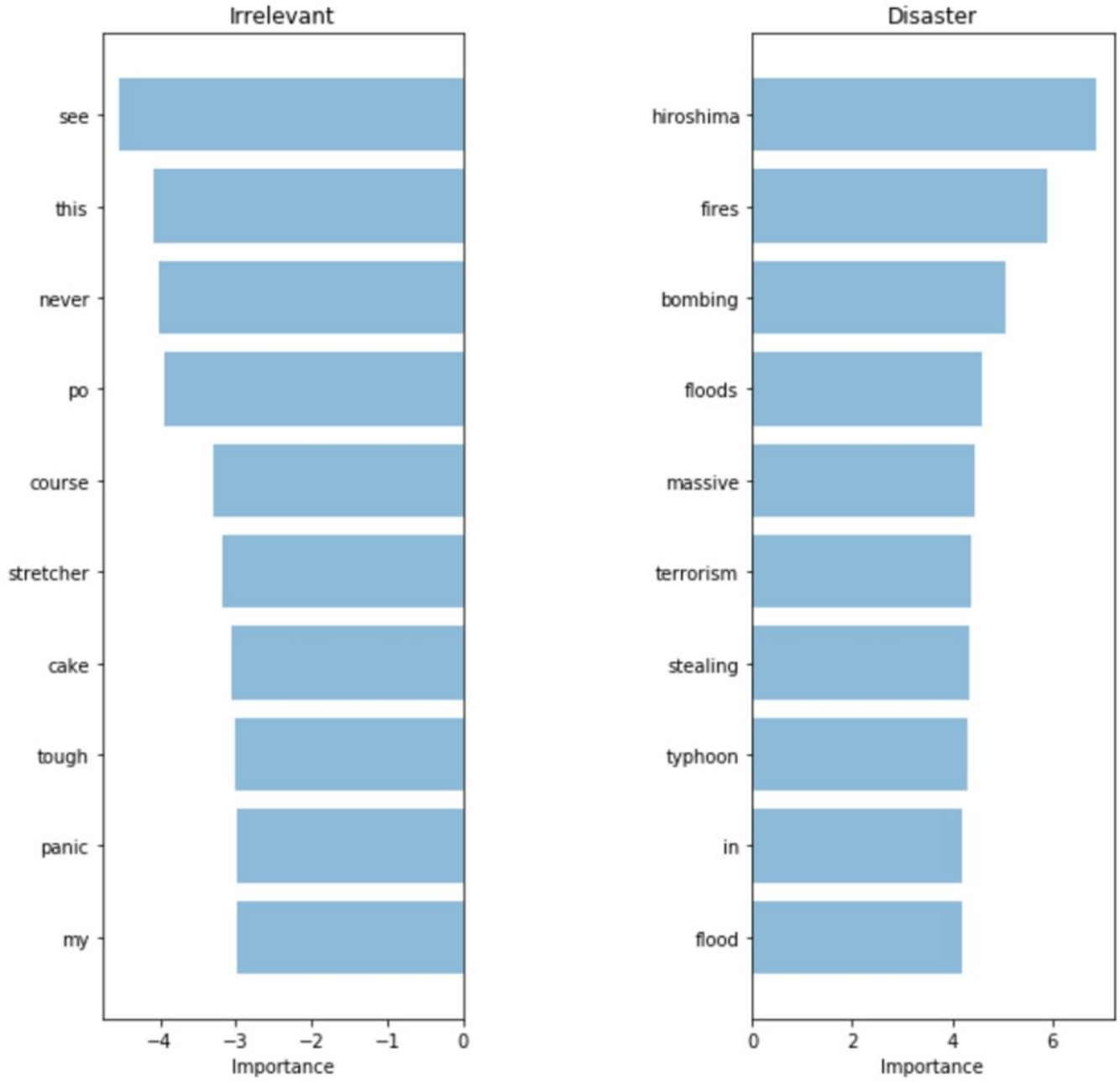
TF-IDF特征抽取方式

为了让模型专注于学习更有意义的词汇，我们可以在词袋模型上面使用TF-IDF特征抽取方式。TF-IDF通过词汇在数据集中的词汇的tfidf值来度量该词语的重要性，适度削弱出现过于频繁的单词。下图是基于TF-IDF特征抽取方式构建的模型，经过PCA降维后可视化：



从中可以看出，两种颜色之间有了更清晰的区分，使这两类数据更易于被分类器分开。在新模型上训练Logistic回归，我们得到了76.2%的准确度，说明TF-IDF特征提取方式确实有助于提高识别性能。

尽管只是非常微小的改进，但我们的模型能否就此学到更重要的词汇呢？如果能得到更好的结果，同时还能避免模型在无关词汇上的过拟合，使用TF-IDF特征抽取方式可以被认为是对学习的效果有了显著的改善。



可以看到，新模型学到的词汇看起来相关度更高！尽管测试集的指标只是略有增加，但是我们对模型的识别性能更有把握，因此部署新模型的交互系统会让用户体验更为舒适。

Step 7:使用语义

Word2Vec特征表示方式

之前的词袋、one-hot和tfidf，都是基于待分析的语料库构建自己的特征集，并用该特征集去抽取特征，将待分析语料转化为计算机可分析的数据。

然而，如果部署该模型后，我们很可能会遇到一些特征集中从未出现过的词汇。先前的模型均无法正确分类这样的新数据，即便其中的词汇与语料库非常相似。

要解决这个问题，我们就要捕捉词汇的语义，这就意味着模型需要理解“好”与“积极”在语义上的距离要比“杏”和“大陆”更接近。这里的工具就是Word2Vec。

使用预训练好的word2vec数据

Word2Vec是一种为单词查找连续嵌入的技术，这里嵌入很难说清楚。你可以简单的认为计算机随机的给每一个单词分配一个多维向量（word2vec优点是任何两个单词可以通过词向量进行相似度计算）。通过阅读大量的文字，它能够学习并记忆那些倾向于在相似语境中出现的词汇。经过足够的数据训练之后，它会为词汇表中的每个单词都生成一个300维的向量，用以记录语义相近的词汇。

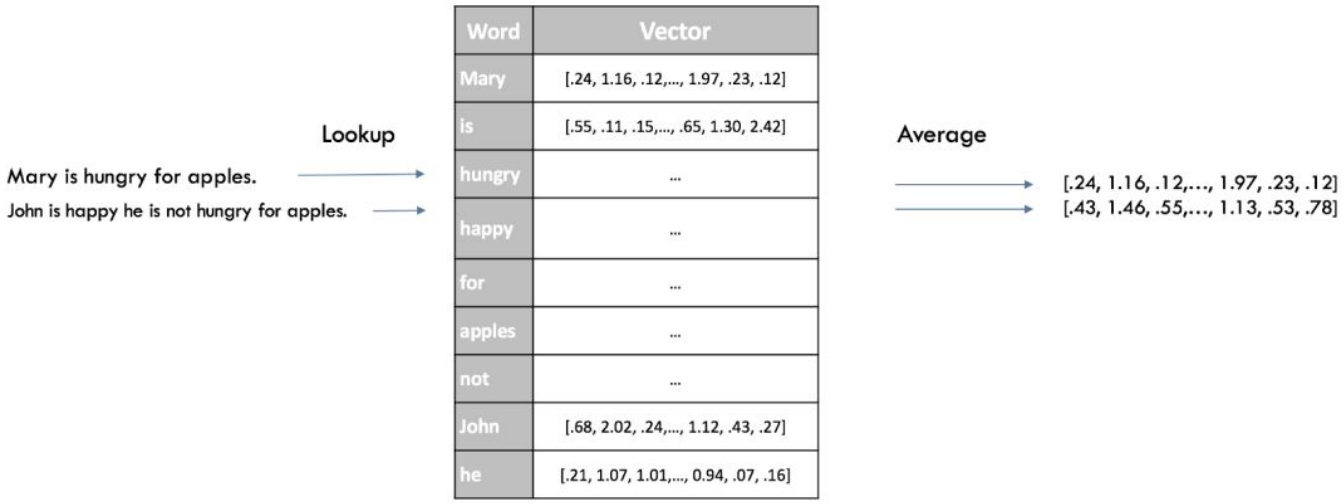
Word2Vec作者在一个非常大的语料库上预训练并开源了该模型。利用这一语料库，我们可以将一些语义知识纳入到我们的模型内。预训练好的词向量可以在Word2vec中找到并下载。

> https://code.google.com/archive/p/word2vec/

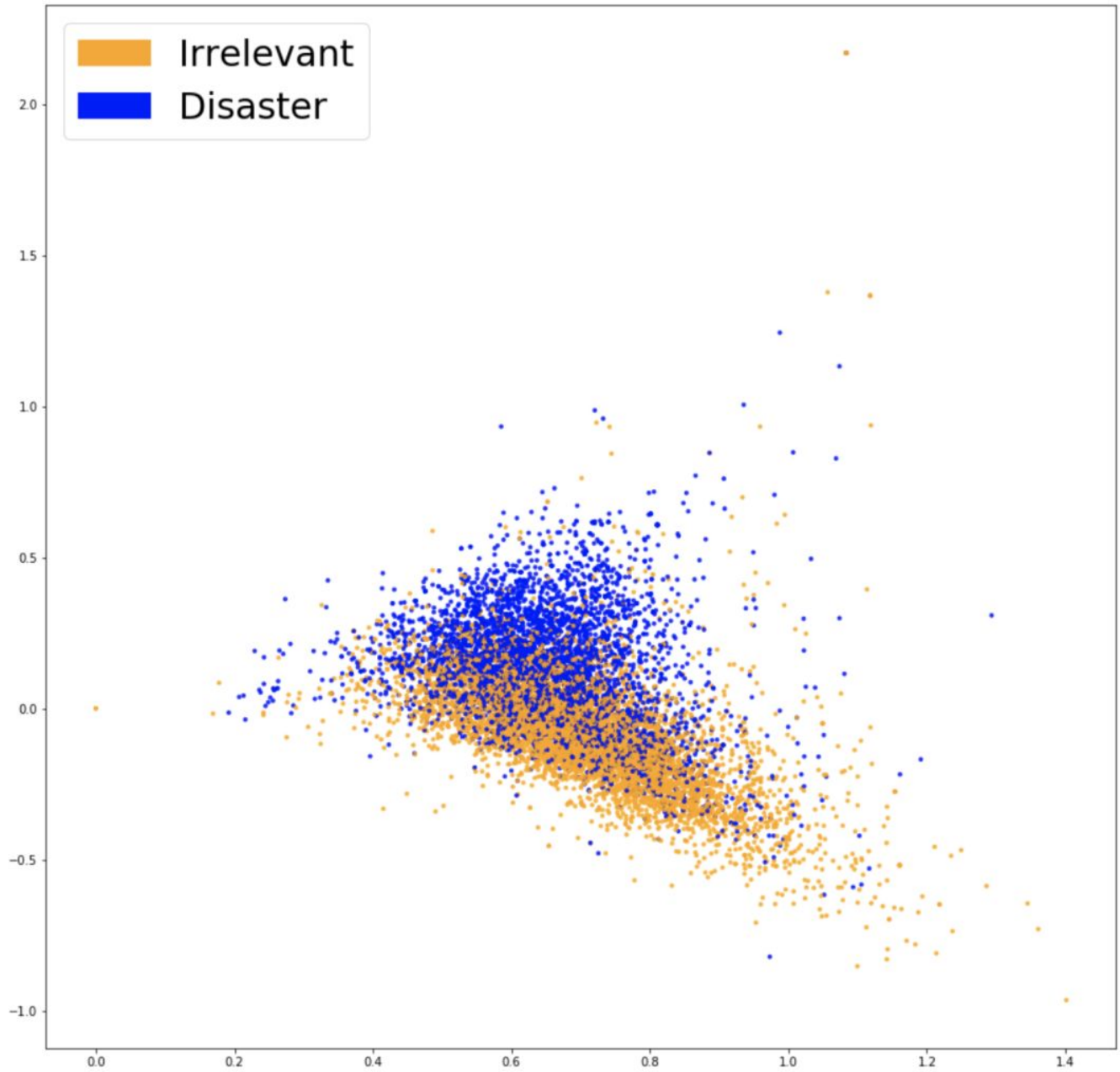
句子分级特征表示

让分类器快速得到句子嵌入的方法，是先将句中所有词汇Word2Vec得分的平均化。这与此前词袋模型的做法类似，但这里我们在保留语义信息的同时只丢弃句法。

Word2vec模型的句子特征表示



Word2Vec句子特征表示模型分析的可视化结果，结果如下：



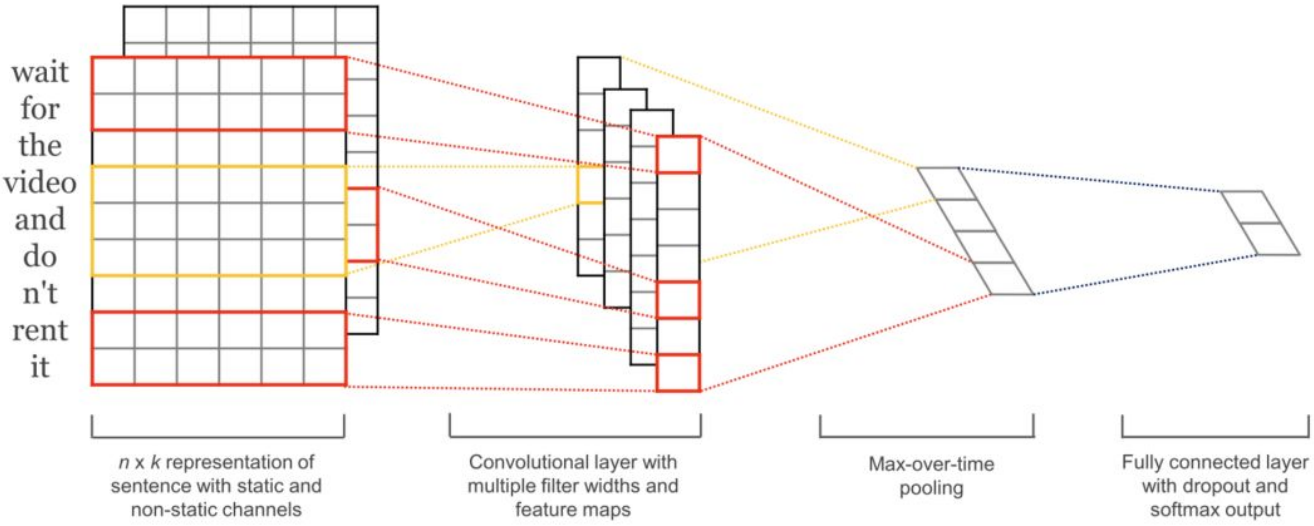
在这里，两组颜色的分离程度更大一些，这就意味着Word2Vec能够帮助分类器更好地分离这两种类别。再一次使用Logistic回归，得到77.7%的准确率，是我们迄今最好的结果！

复杂性/可解释性权衡取舍

与先前的模型不同，新模型无法将每个单词都表示成一维向量，因此很难看出哪些词汇与我们的分类结果相关度最高。尽管我们仍可使用Logistic回归的系数，但它们仅与嵌入的300个维度相关，而与词汇索引值并不相关。

Step 8:使用端到端的方式训练语法特征

我们已经介绍过如何用快速有效的办法来生成紧凑的句子嵌入。然而，通过省略词汇的顺序，我们也放弃了语句的所有句法信息。如果简单的方法给不出令人满意的结果，那我们就用更为复杂的模型：将整个句子作为输入并预测标签，同时无需建立中间表示。一种常见的做法是把句子视为词向量的序列，如使用Word2Vec，或是GloVe、CoVe等更先进的方法。接下来我们详细讨论。



训练速度很快。它作为一种入门级的深度学习架构，能够很好地解决分类问题。尽管CNN声名主要源自它在图像处理方面的出色能力，但在文本相关任务上，它所提供的结果也相当优异。且相比多数复杂的NLP方法（如LSTM、Encoder/Decoder架构等），CNN训练速度也更快。它能够保留单词的顺序，很好地学习单词的序列特征以及其他有用信息。相对于先前的模型，它可以区分出“Alex eats plants”与“Plants eat Alex”之间差异。

相比先前的方法，该模型的训练不需更多的工作，但效果却好得多，准确率高达79.5％！与前面的步骤一样，下一步也要继续探索并可视化该模型的预测结果，以验证它是否为最佳模型。做到这一步，你应该能自己完成这里的操作。

写在最后

简单回顾一下，我们在各个步骤中所用的方法是这样的：

- 从一个简单的模型快速开始
- 解释模型的预测
- 理解模型分类中的错误样本
- 使用这些知识来决定下一步的部署。

上述八大步骤所用的模型是我们处理短文本时的几个特定实例，但其背后的解决方法已经广泛被用在各类NLP问题的实际处理上。

往期文章

100G Python学习资料：从入门到精通! 免费下载

为什么你要为2019，而不是2018做计划?

我分享文章的不良动机就是为了涨粉!

2017年度15个最好的数据科学领域Python库

如何从文本中提取特征信息?

初识K-means算法

对于中文，nltk能做哪些事情

留在网上的每个字，都在泄露你的身份

优雅简洁的列表推导式

Get小技巧等分列表

如何对数据进行各种排序?

【视频讲解】Scrapy递归抓取简书用户信息

美团商家信息采集神器

用chardect库解决网页乱码问题

gevent：异步理论与实战

轻盈高效的异步访问库grequests库

selenium驱动器配置详解

爬虫神器PyQuery的使用方法

简易SQLite3数据库学习

Python通过字符串调用函数

Python圈中的符号计算库-Sympy

Python中处理日期时间库的使用方法

阅读原文