MESSAGENET：基于自然语言处理和元数据的消息分类

Adar Kahana,Oren Elisha微软研发中心，ILDC
{adarkahana, orelisha}@microsoft.com

arXiv:2301.01808v1 [cs.LG] 4 Jan 2023

摘要

本文提出了一种用于消息分类的新型深度学习（DL）方法。我们的方法基于最先进的自然语言处理（NLP）基础模块，并结合了一种新颖的技术，用于融入消息中通常可获取的元数据输入，例如发件人信息、时间戳、附加图像、音频、隶属关系等。正如本文所展示的，超越单纯的文本，利用消息中所有可用的渠道，可以获得更优的表示和更高的分类准确率。为实现消息表示，神经网络架构中会针对每种输入类型设置一个适合其数据类型的专用模块进行处理。这种实现方式使所有模块能够同时联合训练，并在网络中形成跨渠道特征。我们在实验部分展示，在某些情况下，消息的元数据包含仅凭文本无法提取的额外信息；利用这些信息能够取得更好的性能。此外，我们证明，多模态模块方法在将元数据注入文本分类器方面优于其他方法。

关键词 消息分类 · 元数据注入 · 深度学习 · 自然语言处理

1 引言

许多现实应用都需要进行消息分类和回归，例如处理垃圾邮件 [1],工单路由 [2], 文章情感分析 [3]等。准确的消息分类能够改善呼叫中心（根据主题路由工单） [2], 警报系统（标记重要性极高的警报消息） [4],以及对收到的消息进行分类（自动整理杂乱的电子邮件） [1, 5]等关键场景。文本分类与消息分类之间的主要区别在于，后者可以利用额外属性，例如发件人信息、时间戳、附加图像、音频、隶属关系等。新的消息分类竞赛常常出现在知名平台上（例如 Kaggle [6]），可见这一主题备受关注。该领域已有许多可供探索的数据集，但尚无一种能够适用于所有场景、兼具高准确率、高效率和简洁性的算法（无论从实现还是解释的角度来看）。

自然语言处理领域的一项显著进展是基于注意力机制的Transformer架构 [7]。这类方法擅长发现词语之间的局部关联，并更好地理解句子的含义。一个领先的例子是来自Transformer的双向编码器表示（BERT） [8] 及其变体 [9, 10, 11],，它们在某些基准测试中取得了优异成绩 [12, 13]。Huggingface Transformers [14],等多个软件包让这类模型易于获取和使用，并提供预训练版本。此外，还可以利用迁移学习 [15] 继续使用自己的数据训练 BERT，从而为当前特定任务创建定制模型。

BERT 以及其他许多基于 Transformer 的模型，都是为处理文本而设计的。它们通过将给定文本编码为词元来处理其中的词语，并根据词元之间的联系学习句子的上下文。这种方法有所局限，因为有时还可以提取和利用更多信息，而这些信息不一定是文本信息。在本文中，我们将这类信息称为元数据，以便将其与文本内容主流区分开来（不过，视具体应用而定，也可以将其视为核心数据）。例如，元数据可以是文本的撰写、发送、发布等时间戳。另一个例子是文本的作者，例如在作者名单较短的情况下，

语料库的作者名单。有些尝试将这些信息纳入 BERT 模型，例如为作者或时间段（如每月一个词元）分配人工词元 [16]。这种方法有所局限，因为并非所有元数据条目都适合通过分词进行编码。以时间段为例，划分的时间段越多，需要的词元就越多，导致计算资源消耗过大；时间段越少，则会造成信息损失。另一种方法是将 Transformer 模块生成的嵌入与元数据嵌入模块的输出拼接起来。在这种方法中，先使用文本训练一个处理文本的 Transformer（采用直接学习或迁移学习），再使用其他独立模块（如时间序列嵌入、发送者嵌入等）来嵌入元数据。随后将所有嵌入拼接起来，作为分类网络的输入。这种方法的缺点是，网络内部特征并非通过组合不同的输入流进行训练，因此无法捕捉交叉依赖特征（例如，电子邮件的重要性不仅取决于其内容，还取决于发送者、发送时间、收件人以及附件等）。

创新点

为弥合这些差距，我们实现了一种基于 Transformer 的模型，能够同时利用文本（Transformer架构）和元数据进行训练。我们创建了一种新型的基于模块的网络架构。每个模块负责处理不同类型的输入。划分为多个模块后，模型便能灵活处理不同类型的输入。此外，与标准做法建议分别进行训练，并采用“分类器投票”方法不同，所提出的方法会同时对文本和元数据进行训练，使语言模型（Transformer）模块的权重能够根据元数据分类模块权重传递的信息进行调整，反之亦然。我们展示了该方法的实验结果：主模块基于 Transformer，用于处理文本；另设一个模块，分别处理预处理后的元数据输入。该方法还可扩展以支持更复杂的模块，例如用于图像的高级深度学习模型 [17],、用于从时间元数据中提取信息的时间分析模块 [18], 、用于多个文本输入（例如电子邮件的主题和正文）的额外 Transformer 模块、类别数据等。为展示该方法的性能，我们在多个公开数据集上进行了实验（亚马逊 [19], 、Yelp [20], 、Reddit [21] 和安然 [5]），以证明采用模块架构的优势，并将其与文献中的基准方法进行比较（相关工作见第 2 节）。这些基准方法基于 Transformer 基准模型（BERT）、随机森林（RF）分类器和多层感知机（MLP）网络。我们取得了有竞争力的结果，并在大多数情况下优于这些基准，表明相较于仅使用文本进行分类任务，元数据中还有大量有待提取的信息。

2 相关工作

自然语言处理任务。 BERT [8] 的发表是文本分类领域的一个转折点。作者展示了该模型在问答、命名实体识别和文本蕴含等复杂任务上的高准确率 [13]。此后，许多作者研究了改进的架构和变体，例如 RoBERTa [9], 、ALBERT [10], 、DistilBERT [11], 等。有些研究致力于提升模型在基准任务上的表现，另一些则开发了更轻量的模型，在保持竞争力的准确率的同时降低计算需求。其他方案，如 XLNet [22] 和 GPT-3 [23], , 则提出了可与 BERT 竞争的架构（同样采用 Transformer）。这些模型常用的基准包括 GLUE、SuperGLUE [13], 、SQuAD 2.0 [12], 等。文本分类作为基准任务并不常见，但正如本文所示，这些模型也可用于该任务。

Transformer 的易用性。 Transformer 日益普及的另一个因素，是各种开源代码库让数据科学家能够轻松尝试不同的架构，并将其用于自己的应用中。Huggingface Transformers 软件包 [24] 是一个 Python 库，可用于训练和微调模型；它提供了丰富的基础模型供选择，且易于实现。GPT-3[23] 已作为开源项目发布，与其他若干实现类似，它还提供了便捷的应用程序编程接口（API）。值得一提的是，许多不使用机器学习进行文本分类的库也已存在，例如 NLTK [25], 、spaCy [26], 等。这些库同样易于使用，并提供高级自然语言处理特征提取及其他文本分析工具。

文本分类。 文本分类涉及许多任务，每项任务都可以视为一个研究领域。其中一项广受欢迎的任务是情感分析，旨在将文本分类为正面或负面。 [3] 中的综述介绍了这一领域面临的挑战和最新进展。另一个例子是垃圾邮件或正常邮件任务，即尝试区分相关电子邮件与无关邮件（如广告、网络钓鱼尝试等） [1]。在这项工作中，我们研究消息的多标签分类。例如，根据购买评价对产品进行分类，根据帖子对讨论串进行分类，根据客户评论判断餐厅的烹饪特色，根据购买反馈判断产品类型，对收到的电子邮件进行分类，等等。对于这些任务，每项任务都有公开的

可用数据集，并在这项工作中用于量化所提方法的成效。此外，Kaggle [6]和其他机器学习研究基准网站上也举办了许多使用这些数据集的竞赛。

结合元数据进行消息分类。 使用 Transformer 对消息进行分类时，有两种常用方法可以将元数据与文本信息结合起来。第一种方法是将 Transformer 计算得出的嵌入与其他专门针对元数据构建的嵌入系统的输出拼接起来。在 [27], 中，作者将这种方法用于视觉和音频元数据。在 [28] 中，作者将文本属性作为元数据，并证明这种方法也适用于德语。在 [29] 中，元数据是扫描文档的布局信息，作者提出了一种创新架构，用于提取文本和布局信息中的内容。还有许多其他研究也在探索这种方法。虽然这种方法易于实现，但也有一些缺点。通常，文本和元数据分别独立训练，决策则采用“分类器投票”的方式做出。这可能导致冲突，因为文本对标签的响应可能与元数据对标签的响应不同，从而造成预测置信度极低。我们将在结果部分比较所提出的方法与这种方法的性能。第二种常见方法是为元数据分配词元，并将其添加到 Transformer 的词元化输入中。在 [16], 中，除了标签层级之外，作者还提出了一种方法，将多种类型（网页、参考文献等）的元数据输入作为词元注入嵌入向量。由于使用词元嵌入信息在算法和实现方面都很简单，开发者会在代码中采用这种方法；它也常见于许多线上笔记本和博客（开源代码）中。这些方法的主要缺点是对输入数据的鲁棒性不足。例如，将图像表示为一系列词元，要么通过对图像进行编码来实现，而这通常会导致信息损失；要么使用大量词元，从而使计算成本呈指数级增长。在本文所述的数值实验中，我们没有将这种方法纳入比较，因为我们使用的特征提取会产生数量不固定且可能很多的特征，采用这种方法会导致计算资源耗尽。在下一节中，我们将提出一种能够解决这两种常用方法所存在问题的方法。

3 方法

我们提出了一种基于模块的方法，用于针对特定的文本分类任务，利用元数据训练语言模型。将每种类型的元数据输入拆分为不同模块后，便可使用最先进的深度学习架构，针对每种元数据类型分别进行处理，从而提高效率。此外，训练时会使用所有模块，并在统一的训练循环中进行；每个优化器步骤都会调整所有模块的全部权重，使来自文本和元数据源的所有信息都能参与学习过程。

3.1 模块架构

Transformer模型（包括BERT）可将输入文本及其对应的输出标签用于文本分类。不过，我们认为，文本的元数据中蕴含着大量信息。如图1所示，我们将Transformer模型用作神经网络中的一个独立模块。随后，还可以添加其他模块来处理元数据输入。

随着深度学习领域的最新发展，出现了许多先进的方法，可从输入信号中提取信息以进行分类。例如，在 [30] 中，作者讨论了如何利用深度学习分析时间序列数据。消息通常具有时间属性，例如电子邮件的到达时间、评论发布的时间、论文发表的时间等。我们提出将这些进展结合起来，以更好地进行模型训练。

图1展示了所提出方法的整体示意图。第一行说明了标准的Transformer架构 [24]。输入为分词后的消息，随后是经过初步预训练的Transformer层。这些层会进一步训练（采用迁移学习），以生成嵌入，并使用分类层预测消息的类别。图中以虚线表示Transformer层，以体现迁移学习过程。第二行展示了一个基于深度神经网络的元数据提取器。绿色层表示经过分类训练的层（例如，全连接网络架构）。第三行展示了另一种迁移学习架构。每一行都表示神经网络中一个不同的模块，用于处理不同的元数据输入。

3.1.1 表示

使用所提出的方法，数据的预期表示与第2节中提到的主流方法所得到的表示大不相同。由于各模块是同时训练的，元数据中的信息可能会影响核心文本模块（Transformer模块）的训练，反之亦然。因此，会产生一种新的

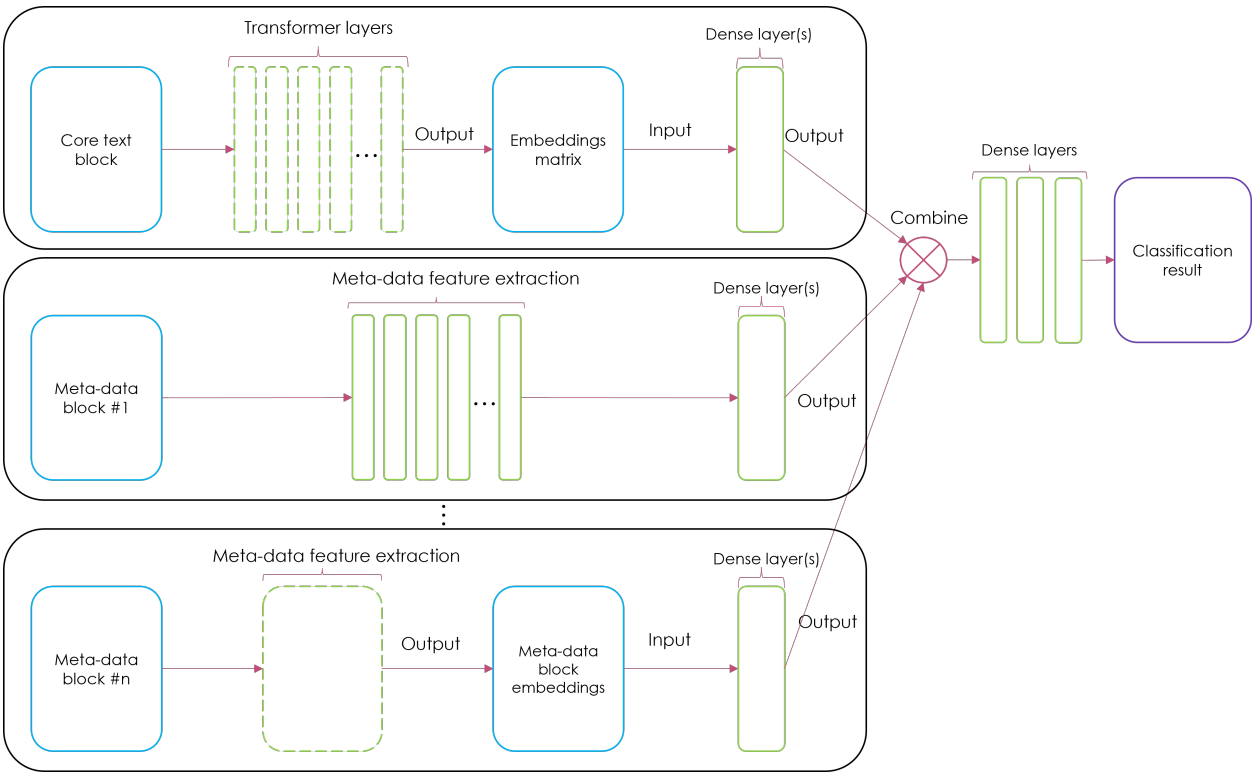


图1：块方法示意图。第一行展示了用于处理文本输入的 Transformer架构，第二行和第三行展示了用于处理特定元数据输入的神经网络。随后将所有模块组合起来，生成统一的预测结果。蓝色方块为静态层，绿色方块为可训练层。绿色虚线方块表示迁移学习层

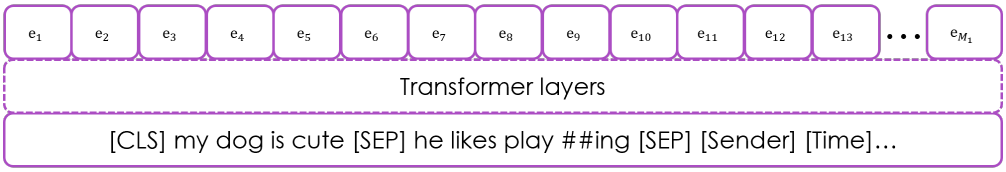
且不同的表示。举一个简单的示例，假设某个样本的文本暗示了一个特定类别，而元数据则暗示了另一个类别，那么文本模块的训练方式就会有所不同（考虑到这一点，文本表示也会有所不同），并且文本模块的权重能够捕捉这种差异。我们在图2中展示了这种差异。图2a展示了如何使用词元将元数据注入Transformer层，从而生成嵌入（e1-eM1）。图2b展示了将Transformer层生成的嵌入（e1-eM2）与元数据的向量表示（v）拼接。图2c展示了所提出的块方法。可训练模块用绿色框出，以强调这些模块是共同训练的，从而生成消息的统一表示；相比之下，在图2b中，Transformer层与全连接层是独立训练的，随后再将输出表示拼接起来。需要强调的是，这些模块可以采用不同的架构，也可以生成不同的向量表示，以获得更好的表示。

3.1.2 组合

在这些模块之后，我们提出对各层进行组合（例如，对输出取平均、求和、拼接等）。这种组合也可以作为训练循环的一部分进行训练。根据实验测试，我们发现，将组合步骤设为对输出进行加权拼接，效果最佳；这样生成的输出大小为类别数乘以模块数。然后，我们使用若干全连接层生成大小为类别数的输出，并应用 Softmax 激活函数进行分类。

4 实验

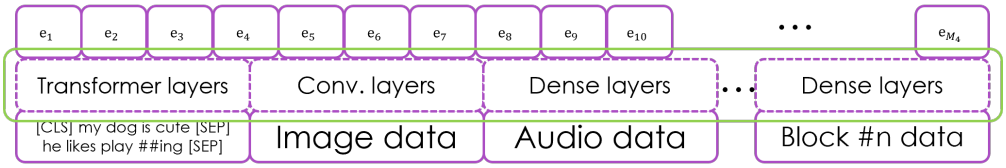
在数值实验中，我们使用两个模块训练所提出方法的网络：a）作用于主文本的 Transformer 模块；b）作用于一维元数据向量的全连接块，即元数据模块。我们强调，可以使用更先进的模块，但即使是这种简单的架构，也取得了能够证明元数据和块方法价值的结果。此外，这种方式也便于区分



（a）将元数据注入为词元的示意图



（b）元数据拼接示意图



（c）块方法示意图

图2：两种常见方法（a、b）和所提出方法（c）的概览，强调各模块是共同训练的，因此一个模块的信息可能会影响其他模块的训练（不同于例如（b）中各模块分别训练后再进行拼接的方式）

仅使用 Transformer 的架构与同时使用 Transformer 和元数据的架构，从而更清楚地解释元数据的贡献。

4.1 数据集

亚马逊评论。 亚马逊产品评论数据集 [19] 包含大量产品评论，因产品类别（标签）而异。主要文本是评论的描述。我们从亚马逊数据集中提取元数据，包括评论者字段（发送者）、评论创建日期和时间（时间戳）以及总体满意度（枚举）。我们预期存在这样的相关性：某位用户经常评论某一特定类别的产品，或某些产品在一天中的特定时间被更频繁地购买。这些信息包含了文本中无法获得的产品信息，并增强了分类能力。该数据集总计包含 8,283 万条产品评论。我们通过选取每个类别中最前面的 100,000 条评论，再从中选择文本最长的评论，对评论进行子采样。最终，我们保存了 4,687 条评论用于训练、521 条用于验证，以及 2,064 条用于测试（删除无文本的消息）。最终数据集接近平衡状态（每个类别中的评论数量大致相同）。

Yelp 开放数据集。 Yelp 餐厅评论数据集 [20] 包含餐厅评论，这些评论按菜系类别（标签）划分。主要文本是餐厅评论。我们从评论者字段（发送者）、评论创建日期和时间（时间戳）、评论是否有用/有趣/酷（枚举）以及星级数（枚举）中提取元数据。这里预期的相关性包括：评论者会专注于特定餐厅；早餐和晚餐的时间戳不同；评论本身的属性可以提示对用餐体验的评分。我们按照与亚马逊数据集类似的方式清理数据集，并保存了 4,612 条评论用于训练、513 条用于验证、2,050 条用于测试，类别也大致均衡。

Reddit。 Reddit 数据集 [21] 包含大量 Reddit 帖子，每条帖子都属于一个特定的子版块（标签）。具体来说，我们使用的是 2010 年发布的第 2 版数据集。主要文本是帖子内容。我们从发帖者字段（发送者）、发帖日期和时间（时间戳）、帖子是否有评论（枚举）以及帖子是否有附件（枚举）中提取元数据。预期的相关性包括：用户通常会在同一个子版块发帖；不同的发帖时间可能与某些特定事件有关；评论和附件属性通常能反映评论的重要性。我们也对数据集进行了清理，并保留 4,950 条帖子用于训练、550 条用于验证、2,200 条用于测试。我们关注 12 个子版块（12 个不同的标签），数据集大致均衡。

安然电子邮件数据集。 安然电子邮件数据集 [5] 包含一个电子邮件语料库。尽管该数据集可公开获取，但我们获得的是一个有限的内部版本，其中的邮件已根据电子邮件类别（标签）进行标注。主要文本是电子邮件正文。我们从电子邮件发件人字段（发送者）以及电子邮件接收日期和时间（时间戳）中提取元数据。预期的相关性包括：某些发送者会围绕特定主题发送电子邮件（同事会发送有关工作、会议等的邮件，而朋友会发送有关个人事务的邮件），电子邮件的到达时间通常与主题相关。我们保留 4,500 封电子邮件用于训练、500 封用于验证、2,000 封用于测试。该数据集并不均衡。

4.2 特征提取器

为进行数值实验，我们首先介绍一种实际使用的特征提取器，用于从数据集的可用字段中提取元数据信息。对于每种元数据，我们都会提取相应的特征。由于不同数据集具有不同的字段，我们使用通用模块从用途相同的不同字段中提取信息，例如电子邮件数据集中的发件人字段，或餐厅评论数据集中的评论者字段。表1概述了我们从数据集中提取的元数据特征。请注意，不同的数据集具有不同的字段。有些数据集没有枚举字段，有些则有多个枚举字段。

字段类型	特征	槽位数量	描述
发送者	主要发件人	120	独热向量，每个位置表示该发送者是否属于前 100 名发送者之一
	主要隶属机构	120	与发送者的主要隶属机构相同
	发送者频率	1	计算来自该发送者的消息的频率
时间戳	Day	7	独热向量，每个时段表示消息是否到达在当天
	工作时间	1	指示消息是否在工作时间内到达或没有
	高峰时段	50	创建一个包含 50 个分箱的直方图以及一个对应的、包含 50 个槽位的独热向量，其中每个槽位表示消息是否已到达在对应的时间段内
枚举	枚举值	#选项	值为 1 的独热向量在表示该选项的槽位中，否则为 0
数值搭配	数值	1	该数值

表1：各字段元数据特征提取概述

4.3 训练细节

我们采用简单的网络架构，以便清楚地展示该方法的优势。需要强调的是，通过更先进的元数据处理方式并引入更多模块，预计可以获得更高的准确率。用于文本嵌入的模型是来自 HuggingFace [14] library 的预训练 BERT 模型（bert-base-uncased）。针对新任务（新数据），我们使用迁移学习，利用元数据进一步训练语言模型。此外，我们使用了 ktrain 软件包，具体来说是其中的文本分类模块，并对其进行了重写，使其包含一个消息分类类。对于元数据分类模块，我们使用两个全连接层：第一层的大小与输入元数据的维度相同，第二层的大小则等于类别数；两层均使用修正线性单元（ReLU）激活函数。模块之后，我们根据测试场景，选择使用随机森林分类器（包含 250 棵树，深度不受限制，每个叶节点最多包含一个样本，分裂准则基于基尼指数），或将各项输出合并为一个输出（采用平均或可训练权重）。对于输出加权，我们在末尾使用一个全连接层，其大小等于类别数，后接 Softmax 激活函数（用于分类）。

4.4 结果

We 使用第 4.1 节中提到的数据集。我们从准确率方面比较多种方法 :

1. 预训练 BERT 与额外的全连接层。2. 预训练 BERT 与随机森林分类器。3. 预训练 BERT 与拼接的元数据及额外的全连接层。4. 预训练 BERT 与拼接的元数据及随机森林分类器。5. 采用迁移学习的 BERT 与额外的全连接层。6. 采用迁移学习的 BERT 与随机森林分类器。7. 采用迁移学习的 BERT 与拼接的元数据及额外的全连接层。8. 采用迁移学习的 BERT 与拼接的元数据及随机森林分类器。9. 所提出的方法（BERT 和元数据）结合输出平均。10. 所提出的方法（BERT 和元数据）结合输出加权。

方法 1-4 使用预训练 BERT 模型（不进行迁移学习）生成的嵌入，并训练额外的全连接层或随机森林分类器 [31] 进行分类。按照文献中的常见方法，我们分别采用不将元数据与嵌入拼接（方法 1-2）或将二者拼接（方法 3-4）的方式。方法 5-8 与此类似，但会进一步使用特定任务数据训练 BERT Transformer 层。方法 3、4、7、8 是文献中将 BERT 模型嵌入与元数据嵌入拼接的基准方法[16]。方法 9-10 是本文提出的方法，旨在探究将 Transformer 输出与元数据模块输出相结合的效果：一种方法对模块输出取平均值，另一种方法则在拼接各模块的输出表示后，使用全连接层学习权重。后一种方法需要通过训练来学习这些权重，并且其实现方式确保了这些权重与网络中的其他所有权重在同一训练过程中学习。结果见表 2。观察结果可知，本文提出的方法与其他所有方法相比均具有竞争力，甚至优于其中大多数方法。

方法#	亚马逊评论	Yelp 开放数据集	Reddit	安然电子邮件
1	0.66	0.29	0.56	0.49
2	0.61	0.22	0.52	0.49
3	0.65	0.24	0.46	0.48
4	0.61	0.22	0.5	0.5
5	0.74	0.39	0.61	0.47
6	0.73	0.38	0.6	0.47
7	0.71	0.38	**0.62**	0.47
8	0.73	0.39	0.6	0.47
9	0.7	0.3	**0.62**	0.47
10	**0.77**	**0.4**	**0.62**	**0.53**

表 2：结果表

在安然电子邮件数据集上观察到一个有趣的结果。我们发现，预训练方法的表现优于基于迁移学习的方法。这可以用文本信息与元数据信息之间的不匹配来解释。当文本指向一个类别时，元数据可能指向另一个类别。在这种情况下，方法 1-8 相当于分类器投票。然而，如第 3.1.1 节所述，所提出的方法同时受到文本和元数据的影响，并学习到一种能够考虑这些差异的更优表示，因此表现优于所有其他方法。

5 结论

本文提出了一种用于训练分类模型的新框架。该方案依赖于额外数据通道的可用性，这些数据不一定是文本数据，例如附加图像、音频、发送者和时间戳等。我们提出了一种架构，可以利用上述额外信息的不同模块，并结合文本数据，训练神经网络，从而更准确地执行消息分类。我们通过一组示例展示了该方法的优势，这些示例在数据集和分类算法方面各不相同，并表明所提出的方法优于相关参考工作。

参考文献

[1] Asif Karim、Sami Azam、Bharanidharan Shanmugam 和 Krishnan Kannoorpatti。将电子邮件高效聚类为垃圾邮件和正常邮件：综合无监督框架的奠基性研究。*IEEE Access*，8：154759–154788，2020年。[2] Jianglei Han、Jing Li 和 Aixin Sun。Uftr：用于工单路由的统一框架，2020年。[3] Walaa Medhat、Ahmed Hassan 和 Hoda Korashy。情感分析算法及其应用：综述。*AinShams engineeringjournal*，5(4)：1093–1113，2014年。[4] Dheeraj Gupta、P.S. Joshi、A.K. Bhattacharjee 和 R.S. Mundada。使用基于知识的评估进行入侵检测系统警报分类。载于 *2012 Fourth International Conference on Communication Systems and Networks(COMSNETS2012)*，第 1–8 页，2012年。[5] Bryan Klimt 和 Yiming Yang。安然语料库：用于电子邮件分类研究的新数据集。载于 Jean-François Boulicaut、Floriana Esposito、Fosca Giannotti 和 Dino Pedreschi（编），*Machine Learning: ECML 2004*，第 217–226 页，柏林、海德堡，2004年。Springer Berlin Heidelberg。[6] Kaggle。https://www.kaggle.com/。[7] Ashish Vaswani、Noam Shazeer、Niki Parmar、Jakob Uszkoreit、Llion Jones、Aidan N Gomez、Łukasz Kaiser 和 Illia Polosukhin。Attention is all you need。神经信息处理系统进展，30，2017年。[8] Jacob Devlin、Ming-Wei Chang、Kenton Lee 和 Kristina Toutanova。Bert：用于语言理解的深度双向 Transformer 预训练，2018年。[9] Yinhan Liu、Myle Ott、Naman Goyal、Jingfei Du、Mandar Joshi、Danqi Chen、Omer Levy、Mike Lewis、Luke Zettlemoyer 和 Veselin Stoyanov。Roberta：一种经过稳健优化的 BERT 预训练方法，2019年。[10] Zhenzhong Lan、Mingda Chen、Sebastian Goodman、Kevin Gimpel、Piyush Sharma 和 Radu Soricut。Albert：用于语言表示自监督学习的轻量级 BERT，2019年。[11] Victor Sanh、Lysandre Debut、Julien Chaumond 和 Thomas Wolf。Distilbert，BERT 的蒸馏版本：更小、更快、更经济且更轻量，2019年。[12] Pranav Rajpurkar、Robin Jia 和 Percy Liang。了解你不知道的内容：SQuAD 中无法回答的问题，2018年。[13] Alex Wang、Amanpreet Singh、Julian Michael、Felix Hill、Omer Levy 和 Samuel R. Bowman。GLUE：用于自然语言理解的多任务基准与分析平台。载于 *In the Proceedings of ICLR.*，2019年。[14] Thomas Wolf、Lysandre Debut、Victor Sanh、Julien Chaumond、Clement Delangue、Anthony Moi、Pierric Cistac、Tim Rault、Rémi Louf、Morgan Funtowicz、Joe Davison、Sam Shleifer、Patrick von Platen、Clara Ma、Yacine Jernite、Julien Plu、Canwen Xu、Teven Le Scao、Sylvain Gugger、Mariama Drame、Quentin Lhoest 和 Alexander M. Rush。HuggingFace 的 Transformer：最先进的自然语言处理，2019年。[15] Sinno Jialin Pan 和 Qiang Yang。迁移学习综述。*IEEE Transactions on knowledge and dataengineering*，22(10)：1345–1359，2009年。[16] Yu Zhang、Zhihong Shen、Yuxiao Dong、Kuansan Wang 和 Jiawei Han。MATCH：大规模层级结构中的元数据感知文本分类。载于*Proceedings of the Web Conference 2021*，第 3246–3257 页，2021年。[17] Fei Wang、Mengqing Jiang、Chen Qian、Shuo Yang、Cheng Li、Honggang Zhang、Xiaogang Wang 和 Xiaoou Tang。用于图像分类的残差注意力网络。载于 *Proceedings oftheIEEEconferenceoncomputervision and pattern recognition*，第 3156–3164 页，2017年。[18] Dino Ienco 和 Roberto Interdonato。通过注意力门控自编码器进行深度多变量时间序列嵌入聚类。载于亚太知识发现与数据挖掘会议，第 318–329 页。Springer，2020年。[19] Ruining He 和 Julian McAuley。起伏变化：使用单类协同过滤对时尚趋势的视觉演变进行建模。载于 *proceedings of the 25th international conference on worldwide web*，第 507–517 页，2016年。[20] Yelp。Yelp 开放数据集，https://www.yelp.com/dataset，2014年。[21] Jason Baumgartner、Savvas Zannettou、Brian Keegan、Megan Squire 和 Jeremy Blackburn。Pushshift Reddit 数据集，2020年。

[22] Zhilin Yang、Zihang Dai、Yiming Yang、Jaime Carbonell、Russ R Salakhutdinov 和 Quoc V Le。XLNet：面向语言理解的广义自回归预训练。载于 H. Wallach、H. Larochelle、A. Beygelzimer、F. d'Alché-Buc、E. Fox 和 R. Garnett（编），神经信息处理系统进展，第 32 卷。Curran Associates, Inc.，2019年。[23] Tom Brown、Benjamin Mann、Nick Ryder、Melanie Subbiah、Jared D Kaplan、Prafulla Dhariwal、Arvind Nee- lakantan、Pranav Shyam、Girish Sastry、Amanda Askell、Sandhini Agarwal、Ariel Herbert-Voss、Gretchen Krueger、Tom Henighan、Rewon Child、Aditya Ramesh、Daniel Ziegler、Jeffrey Wu、Clemens Winter、Chris Hesse、Mark Chen、Eric Sigler、Mateusz Litwin、Scott Gray、Benjamin Chess、Jack Clark、Christopher Berner、Sam McCandlish、Alec Radford、Ilya Sutskever 和 Dario Amodei。语言模型是少样本学习者。载于 H. Larochelle、M. Ranzato、R. Hadsell、M.F. Balcan 和 H. Lin（编），神经信息处理系统进展，第 33 卷，第 1877–1901 页。Curran Associates, Inc.，2020年。[24] Thomas Wolf、Lysandre Debut、Victor Sanh、Julien Chaumond、Clement Delangue、Anthony Moi、Pierric Cistac、Tim Rault、Rémi Louf、Morgan Funtowicz、Joe Davison、Sam Shleifer、Patrick von Platen、Clara Ma、Yacine Jernite、Julien Plu、Canwen Xu、Teven Le Scao、Sylvain Gugger、Mariama Drame、Quentin Lhoest 和 Alexander M. Rush。Transformer：最先进的自然语言处理技术。载于*2020*年自然语言处理实证方法会议论文集：系统演示，第 38–45 页，线上，2020年10月。计算语言学协会。[25] Steven Bird、Ewan Klein 和 Edward Loper。使用 *Python* 进行自然语言处理：利用自然语言工具包分析文本。" O'Reilly Media, Inc."，2009年。[26] Matthew Honnibal 和 Ines Montani。spaCy 2：利用 Bloom嵌入、卷积神经网络和增量解析实现自然语言理解。即将出版，2017年。[27] Tanzila Rahman、Mengyu Yang 和 Leonid Sigal。TriBERT：面向视觉声音分离的全身人体中心化音视表征学习。*arXiv* 预印本 *arXiv:2110.13412*，2021年。[28] Malte Ostendorff、Peter Bourgonje、Maria Berger、Julian Moreno-Schneider、Georg Rehm 和 Bela Gipp。利用知识图谱嵌入增强 BERT，实现文档分类。*arXiv* 预印本 *arXiv:1909.08402*，2019年。[29] Yiheng Xu、Minghao Li、Lei Cui、Shaohan Huang、Furu Wei 和 Ming Zhou。LayoutLM：面向文档图像理解的文本与版面预训练。载于第 *26* 届 *ACM SIGKDD*知识发现与数据挖掘国际会议论文集，第 1192–1200 页，2020年。[30] John-Syin Ang、Kok-Why Ng 和 Fang-Fang Chua。利用深度学习建模时间序列数据：评论、分析、评估与未来趋势。载于*2020*年第八届信息技术与多媒体国际会议（*ICIMU*），第 32–37 页，2020年。[31] Leo Breiman。随机森林。机器学习，45(1)：5–32，2001年。