用于实体解析的预训练嵌入：实验分析 [实验、分析与基准测试]

Alexandros Zeakis[1,2], George Papadakis[1], Dimitrios Skoutas[2], Manolis Koubarakis[1]1雅典国立卡波季斯特里亚大学，希腊 {alzeakis,gpapadis,koubarak}@di.uoa.gr2雅典娜研究中心，希腊 {azeakis,dskoutas}@athenarc.gr

arXiv:2304.12329v1 [cs.DB] 24 Apr 2023

摘要

近期许多关于实体解析（ER）的研究都利用涉及语言模型的深度学习技术来提高有效性。这些技术应用于实体解析的两个主要步骤，即阻塞和匹配。研究人员已经测试了多种预训练嵌入，其中最常用的是 FastText 和 BERT 模型的变体。然而，目前尚无对其优缺点的详细分析。为弥补这一空白，我们对 17 个成熟的基准测试数据集上的 12 种流行语言模型进行了全面的实验分析。首先，我们评估将所有输入实体转换为稠密嵌入向量时产生的向量化开销。其次，我们研究它们的阻塞性能，开展详细的可扩展性分析，并将其与最先进的基于深度学习的阻塞方法进行比较。第三，我们总结它们在有监督匹配和无监督匹配中的相对性能。实验结果为主要语言模型的优势与不足提供了新的见解，有助于研究人员和从业者在实践中选择最合适的模型。

PVLDB参考格式：
Alexandros Zeakis[3]、George Papadakis[1],Dimitrios Skoutas[2], Manolis Koubarakis[1]。用于实体解析的预训练嵌入：实验分析 [实验、分析与基准测试]。PVLDB，15(9)：XXX-XXX，2022年。doi:XX.XX/XXX.XX

PVLDB工件可用性：
源代码、数据和/或其他工件已在 https://github.com/alexZeakis/Embedings4ER 上提供。

1 引言

实体解析（ER）是数据集成中的一项关键且具有挑战性的任务，旨在检测属于同一现实世界对象的不同实体档案 [10], 。通过对实体集合进行去重，实体解析可支持广泛的应用，从常见的数据分析任务到高级问答 [9] [12]。

通常，实体解析解决方案分两个步骤进行 [7, 15]。首先，分块通过粗粒度方法将搜索空间限制在最有可能的匹配项，即候选对， [43]。这是控制实体解析固有的二次复杂度并使其能够扩展到大量数据所必需的。然后，匹配执行细粒度处理，检查每个候选对，以确定其是否构成匹配 [42]。

将文本嵌入数值向量已成为自然语言处理及相关任务中一种非常常见的方法 [47]。早期模型采用基于词袋或词频-逆文档频率的稀疏表示 [30]。然而，这些方法只能捕捉句法相似度，而无法捕捉语义相似度。预训练嵌入模型克服了这一局限。

受此启发，实体解析领域的最新突破利用语言模型进行阻塞 [55] 和匹配[3, 33]。这一过程通常包含以下步骤[13]。首先，将每个给定实体转换为一个稠密嵌入向量。为执行阻塞，需要对生成的向量建立索引，并针对每个实体发起一个 K-近邻查询，以识别候选对。随后，由匹配算法处理这些候选对；该算法可以在无监督模式或有监督模式下运行。在前一种情况下，计算嵌入向量之间的相似度，并将其用作二分图中的边权重；图中的节点对应实体，边则连接候选对。然后将该图划分为互不相交的节点集合，使每个集合都包含描述同一现实世界对象的所有实体。在有监督匹配中，将候选对的嵌入向量作为输入送入二元分类器（通常是深度神经网络），由其判断每一对是否为重复项。

例如，考虑图1中的两个实体集合。它们包含来自购物网站的智能手表数据。我们可以看到， e_1 与 $_k$匹配， e_2 与e_j匹配，e_3 与 e_k匹配，尽管它们的描述存在显著差异。要在实体解析中使用所考虑的语言模型，第一步是向量化。在模式无关设置下，该过程会将每个实体配置文件转换为具有固定维度的单个数值稠密向量。实体 e_1 被转换为向量 v_1，实体 e_2 被转换为向量 v_2，依此类推。成功的语言模型会将语义相似的实体分配到距离较小的向量，这正如我们的示例所示。接下来，FAISS(HNSW)会为第一个实体集合的向量建立索引，将其转换为一种称为分层可导航小世界图的图结构，从而实现高效的最近邻搜索。在下一步，即查询阶段，第二个实体集合中的实体实际上会作为查询被提交到FAISS(HNSW)索引中，该索引会为每个实体从另一个实体集合中检索 k 个最相似的向量。由相似向量组成的结果对构成候选对集合。对于每一对候选项，我们估计相应向量之间的（欧几里得）相似度，并为其关联一个相似度分数。分数高于阈值（在我们的示例中为0.5）的候选对随后会由唯一映射聚类算法进行处理。该算法的本质是将每个实体与其最相似的候选项关联起来，并忽略所有其他候选项。这些匹配的实体构成整个过程的输出。

本作品依据知识共享署名-非商业性使用-禁止演绎 4.0 国际许可协议授权。访问 https://creativecommons.org/licenses/by-nc-nd/4.0/ 查看该许可协议副本。如需进行许可协议未涵盖的任何用途，请发送电子邮件至 info@vldb.org 获取许可。版权归所有者/作者所有。出版权已授权给 VLDB基金会。《VLDB基金会会刊》，第 15 卷，第 9 期，ISSN 2150-8097。doi:XX.XX/XXX.XX

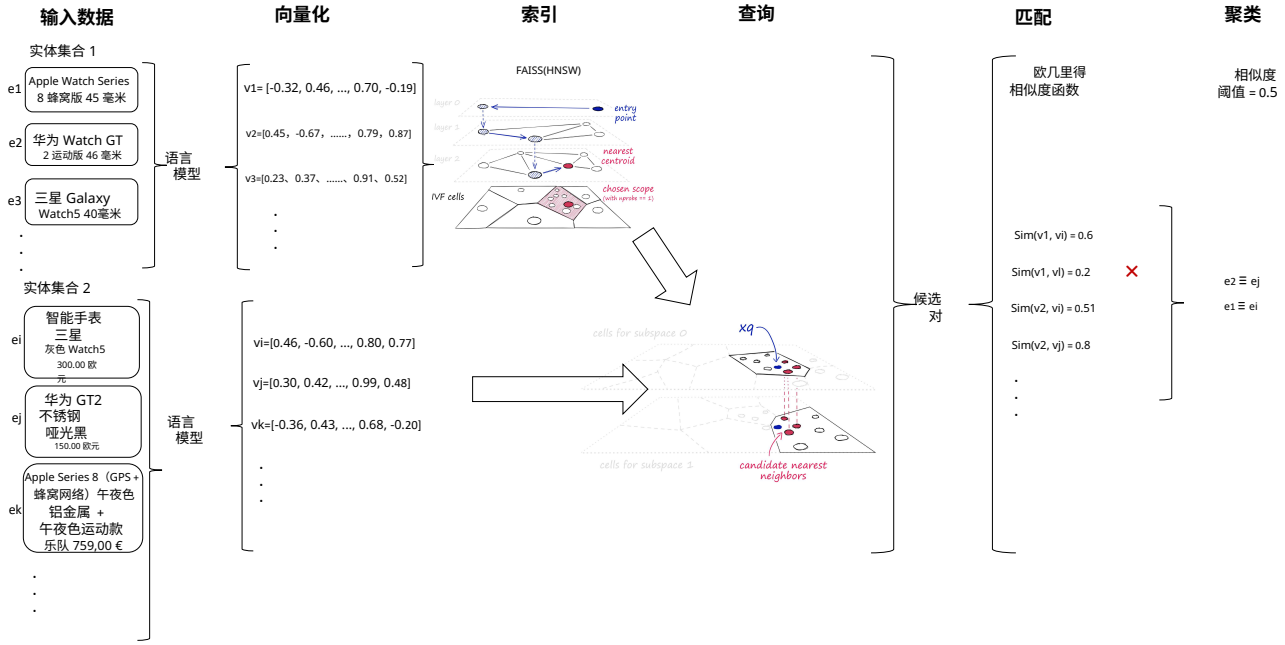


图1：展示　　基于语言模型的实体解析端到端无监督方法　　.

多年来，众多语言模型已被用于实体解析的两个步骤。其中少数模型被用于阻塞：GloVe [13] 和FastText [55, 65]。更多种类的模型被用于匹配：GloVe [13, 33], 、FastText [14, 23, 33, 35, 60, 63, 64] ，以及BERT [45] 及其主要变体，即XLNet、DistilBERT、RoBERTa和AlBERT [3, 5, 25, 38]。然而，目前尚无针对这些模型在实体解析任务中相对性能的系统分析。我们弥补了这一空白，并回答以下研究问题：

（Q1）它们的向量化开销有多大？（Q2）它们在阻塞中的相对性能如何？（Q3）它们在匹配中的相对性能如何？在有监督匹配中，微调会对其产生怎样的影响？（Q4）是否可以仅基于预训练语言模型构建高性能的端到端实体解析流水线，而无需为其提供带标签实例？

为回答这些问题，我们开展了全面的实验分析，涉及 12 个成熟的语言模型：Word2Vec [31,32] , GloVe [46], FastText [2], BERT [11], AlBERT [22], RoBERTa[28], DistilBERT [51], XLNet [62] 以及 S-MPNet [52], S-GTR-T5 [48], S-DistilRoBERTa 和 S-MiniLM [59]。其中一些模型首次应用于实体解析。我们将它们应用于 10 个成熟的真实世界数据集和 7 个合成数据集，并作出以下贡献：

• 我们将它们整理成一个分类体系，以便理解它们的相对性能，并讨论其核心特征，例如维度和上下文感知。• 我们考察每个模型的向量化成本。• 我们评估它们的阻塞有效性、效率和可扩展性。

• 我们比较了它们在有监督匹配和无监督匹配中的相对性能。

• 我们证明，在许多具有长文本属性或短文本属性的数据集中，利用最佳语言模型进行阻塞和匹配的端到端解决方案能够实现高性能。

2 相关工作

阻塞。 利用嵌入向量进行阻塞的第一种方法是 DeepER [13], , 它使用 GloVe 对实体进行向量化，并使用超平面局部敏感哈希进行索引和查询。AutoBlock[65] 通过结合 FastText 嵌入与交叉多胞体局部敏感哈希，进一步超越了 DeepER。然而，它将阻塞视为分类任务，因此需要大量带标签实例来训练其双向长短期记忆网络。DeepBlocker [55] 是一个通用框架，用于合成支持任意语言模型的基于深度学习的阻塞方法。大多数经过考察的方法，包括表现最佳的自编码器，都利用 FastText 嵌入，同时也支持 Word2Vec 和 GloVe。该框架还考察了两种基于Transformer的模型，它们利用字节对编码，以运行时间大幅增加为代价实现了略高的召回率。鉴于效率对于阻塞至关重要，在这些方法中，结合自编码器和 FastText 的 DeepBlocker 达到了最先进水平。**匹配。** 在这里，语言模型通常由有监督的基于深度学习的技术使用。第一种方法同样是 DeepER [13], , 它利用 GloVe。作为其泛化版本，DeepMatcher [33],是一个基于深度学习的匹配算法框架，支持主要的预训练静态模型，即 GloVe 和 FastText，其中 FastText 是预定义选项。类似地，GraphER[23], Seq2SeqMatcher [35],CorDEL [60],MCAN [64], Hier Matcher[14] 和 HIF-KAT [63] 构成了同样利用 FastText 嵌入的独立方法。其原因在于，FastText 具有字符级

操作，可以处理拼写错误形式的噪声，同时还支持未见术语。

较新的研究聚焦于基于BERT的模型，这是因为它们具有动态、上下文感知的特性。首项此类研究是 EMTrans- former [3], ，该研究考察了 BERT 及其三个主要变体：XLNet、RoBERTa 和 DistilBERT。GNEM [5], 使用了相同的模型，并通过构建一个图来扩展 EMTransformer，该图捕获了所有作为匹配输入的候选对之间的关系。GNEM 还将这一思路应用于 DeepMatcher，并结合 FastText 嵌入。DITTO [25] 通过将基于BERT的语言模型与外部领域特定信息（例如词性标注）和数据增强相结合，扩展了 EM- Transformer，从而获得更多（合成的）训练实例。

JointBERT [45] 突破了传统的匹配二分类定义，同时支持多分类。 [38], 研究了如何自动调整基于深度学习的匹配算法的配置参数，并在第 3.2 节中考察了所有基于BERT的模型。

不足之处。 我们注意到，这些工作中没有一项考察主要的 SentenceBERT模型（参见第 3.3 节）。此外，也没有研究考察 BERT 模型（参见第 3.2 节）在阻塞任务上的表现。据我们所知，还没有工作以系统的方式研究主要语言模型之间相对的实体解析性能。与我们工作最接近的是一项对 BERT 在匹配场景中工作方式的深入分析 [37], ，但该研究同样忽略了其他所有模型以及阻塞任务。关于语言模型的综述 [27], 、书籍 [47] 和教程 [56] 都过于宽泛，没有重点关注实体解析，因此与我们的工作属于正交关系。本文旨在弥补文献中的这些重要不足。

自然语言处理中的句子相似度任务。语义文本相似度（STS）是一项重要的自然语言处理任务，通常通过 GLUE 基准中的 STS-B 任务 [4]对 Transformer 模型进行评估 [58]。由于在模式无关实体解析中，记录中的所有属性都会被连接成一个“句子”，因此这些任务似乎具有相关性。然而，输入数据的特征有所不同。例如，常见的 STS 基准包含图像说明或新闻标题等句子，而典型的实体解析基准（例如此处考察的数据集）则包含人物或产品名称、电影标题、地址等属性。连接这些属性并不会形成真正的句子，即使它们被当作句子处理也是如此。此外，STS 的任务是预测一个相似度分数，用于表示两个句子的含义有多相似；而实体解析的任务则是判断两个实体实例是否指向同一个现实世界中的实体。因此，直接或简单地假设语言模型在实体解析和 STS 中会表现出相同的性能，并不稳妥。

3 评估中使用的语言模型

我们在评估中使用了三类语言模型：(1) 静态模型，将每个词元关联到一个固定的嵌入向量；(2) 基于*BERT*的模型，根据上下文对每个词元进行向量化；(3) *Sentence-BERT*模型，将每个词元序列关联到一个上下文感知嵌入向量。

对于每个类别，我们根据以下标准选择了一组具有代表性的语言模型：(i) 在实体解析和自然语言处理文献中的受欢迎程度，(ii) 对英语的支持，以及 (iii)

模型	维度	$\|Seq.\|$	参数	阻塞	匹配
Word2Vec (WC)	300	-	-	[55]	[33]
FastText（FT）	300	-	-	[55, 65]	[23, 33, 35], [14, 60, 63, 64]
GloVe（GE）	300	-	-	[13, 55]	[13, 33]
BERT（BT）	768	100	110M	-	[3, 5, 25, 38, 45]
AlBERT（AT）	768	100	12M	-	[38]
RoBERTa（RA）	768	100	125M	-	[3, 5, 25, 38]
DistilBERT（DT）	768	100	66M	-	[3, 5, 25, 38]
XLNet (XT)	768	100	110M	-	[3, 5, 25, 38]
S-MPNet (ST)	768	384	110M	-	-
S-GTR-T5 (S5)	768	512	110M	-	-
S-DistilRoBERTa（SA）	768	512	-	-	-
S-MiniLM (SM)	384	256	22M	-	-

表1：我们实验中使用的语言模型。

是否有开源实现。理想情况下，该实现应包含便于使用各个模型的文档，而且所有语言模型都应使用同一种语言实现，以便进行运行时比较。因此，我们的实验分析依赖于主要语言模型的开箱即用开源实现，任何从业者都能轻松使用这些实现，即使他们并非该领域的专家。

第2节中提到的所有静态预训练模型和基于BERT的模型都满足这些标准，因此被纳入我们的分析。鉴于这些实体解析研究均未考虑已有的SentenceBERT模型，我们根据其分数和总体资源需求，从SBERT库（https://www.sbert.net/docs/pretrained_models.html）中选取了排名前四的模型。所选模型的技术特征汇总于表1中。对于每个模型，我们列出了向量维度、最大序列长度、参数数量，以及将其用于阻塞或匹配的实体解析研究。下面，我们按类别并依时间顺序简要介绍所选模型。大多数模型都有多个版本（例如基础版、大型版），其差异在于学习参数的数量。为确保公平比较，我们采用每个模型的基础版本。我们还使用更大型的版本进行了一些测试，但未观察到结果存在显著差异。

3.1 静态预训练模型

这些模型旨在捕捉文本中的语义相似度，封装来自大型语料库的知识。它们将传统的高维稀疏向量替换为固定大小的低维稠密向量，从而定义一个数学空间；在该空间中，语义相似的词往往具有较小的距离。

Word2Vec [31, 32]是一种浅层双层神经网络，以语料库作为输入，并为每个词生成相应的向量。它采用局部上下文窗口，可以使用连续词袋模型（不考虑词序）或连续跳元模型（考虑词序）。后者能够将句子中表现相似的词联系起来，但无法利用语料库的统计信息。

GloVe [46] 结合了矩阵分解（即全局共现计数）与局部上下文窗口（即词语类比）。它基于 Wikipedia 等大型语料库进行训练，以提供可供通用的预训练向量。由于它基于全局

词典运行，因此只能识别拼写完全一致的词语，无法检测细微的变体。

FastText [2] 将每个词视为一组 n 元语法，而不是单个字符串。它经过训练以将 n 元语法向量化。随后，它将每个词表示为其底层 n 元语法的总和。

3.2 基于BERT的模型

在静态模型中，每个词只有一种表示方式，这种方式存在局限，因为词语往往会根据上下文具有多种含义。Transformer 模型 [57], 引入了上下文感知，这是编码器-解码器 [54]的一种自然演进。后者有许多优点，例如能够处理给定词语周围更大范围的文本。尽管如此，它们仍无法捕捉词语之间任何显著的关系。注意力机制 [1], 的引入解决了这一问题：它通过共享所有相应的隐藏状态，而不只是最后一个状态，促进了各个编码器与解码器之间的通信。Transformer 模型提出的另一项优化是多头注意力机制，它使各个编码器能够并行运行。另一种有用的方法是为每个词元使用位置编码，以处理多义性，也就是同一个词在不同句子中含义各异这一现象。

BERT [11], 即“来自 Transformer 的双向编码器表示”（Bidirectional Encoder Representations from Transformers）的缩写，是 Transformer 模型演进过程中的又一重要进展。它的主要贡献是使用多个 Transformer（仅使用编码器部分，因为它是一种语言表示模型）来预训练可通用的向量。这些向量还可以通过添加输出层进行微调，以适用于各种任务。BERT 使用两项任务进行训练：掩码语言建模（MLM）和下一句预测（NSP）。前者基于词元，因为它会根据句子中未被掩码的词元来预测被掩码的词元。NSP 则基于句子：它接收第一个句子作为输入，并尝试预测第二个句子是否可以接在其后。

AlBERT [22], 即“一个小型 BERT”（A little BERT）的缩写，是一种更轻量的 BERT 版本。BERT-base 包含 12 个编码器和 1.1 亿个参数，隐藏层维度为 768，嵌入层大小相同（参见表1）。AlBERT 仅训练第一个编码器，然后将其所有权重与其余编码器共享。它还通过因式分解将嵌入层缩减至 128 维。这些改动将参数总数降至 1200 万，大幅缩短了训练时间。

RoBERTa [28] 是“鲁棒优化的BERT预训练方法”的缩写。与 BERT 相比：(1) 它使用更多数据以及更多、更大的批次进行训练；(2) 它移除了下一句预测目标；(3) 它会在每个训练轮次中更改掩码词元，从而提高模型的鲁棒性。它的表现通常优于原始 BERT [28]。

DistilBERT [51], 是“蒸馏BERT”的缩写，是一种采用知识蒸馏的轻量版 BERT [16, 50]。原始模型的第二个版本会被构建出来，其中只使用一半的注意力层——每隔一层省略一层——并在训练中使用一种特殊的损失函数，对比教师模型（原始 BERT）与学生模型（DistilBERT）。

XLNet [62] 试图克服 BERT 的一个缺陷：它无法将已预测掩码标记的知识用作另一个掩码标记的输入，因此每次预测都是独立的，也可能是错误的。XLNet 引入了一种变体，称为 MLM 任务，即排列语言建模（PLM）。这项新任务的目标是在不使用任何掩码标记的情况下，以所有可能的方式排列一个句子中的词元。

3.3 SentenceBERT模型

基于BERT的模型大多是为支持基于词元的任务而构建的。需要句子表示的监督任务可以利用特殊词元 [CLS], ，但在回归或无监督任务中，这会带来计算开销，因为所有成对组合都需要输入模型。使用 [CLS] 或对最后一层的输出取平均，效果往往不如 GloVe嵌入 [49]。SBERT [49] 通过提出一种孪生架构解决了这一问题，即使用两个相同的模型，分别以其中一个句子作为输入。这种架构会生成相应的向量，然后根据定义的任务对这两个向量的组合进行评估，例如计算两个句子的余弦相似度。请注意，这种架构与底层的 BERT 模型正交。

S-MPNet扩展了 MPNet [52], ，后者解决了 BERT 和 XLNet 在 MLM 和 PLM 任务中的缺陷。对于前者，它通过打乱句子中的词元，解决了掩码词元预测之间的依赖关系。对于后者，它利用位置信息来减少位置差异。

S-GTR-T5 扩展了 GTR [34], ，后者是一种双编码器，分别将两段文本编码为两个稠密向量。它通常用于对查询和文档进行编码，以计算二者的相似度，进而执行稠密检索。GTR 模型构建于 T5 [48], 之上；T 是一种旨在将所有自然语言处理任务统一到单个模型中的编码器-解码器模型。它没有引入新模型，而是采用了现有技术。文本到文本迁移 Transformer（T5）是一种编码器-解码器模型，其中每个 Transformer 的结构都与 BERT 中的相同。其基本理念是，尽管 BERT 是一种编码器模型，编码器-解码器模型也能取得良好效果，并可用于编码器无法执行的其他任务（例如文本生成）。T5 使用名为“Colossal Clean Crawled Corpus”的数据集进行训练，该数据集包含数百 GB 从网络上收集的干净英文文本。

S-DistilRoBERTa 将知识蒸馏应用于 RoBERTa 模型，以生成更轻量的学生模型。该模型与孪生架构结合，生成最终模型。

S-MiniLM 扩展了 MiniLM [59], ，通过对 BERT 进行知识蒸馏来生成更轻量的学生模型。不同于 DistilBERT 和其他蒸馏策略 [17, 53],，后者受教师层架构的限制，它只模仿自注意力模块，而这些模块是架构中最重要的部分。因此，它可以定义每个 Transformer 中的层数，从而减少所需参数的总数。知识蒸馏发生在预训练模型中，以避免对教师模型进行计算成本高昂的微调。

4 实验设置

4.1 数据集

主要数据集。 大多数实验使用了以下十个用于实体解析的真实且成熟的数据集： D_1，该数据集由 OAEI 20101 提供，包含餐厅描述。 D_2 包含从两家在线零售商 Abt.com 和

[1]http://oaei.ontologymatching.org/2010/im

	D_1	D_2	D_3	D_4	D_5	D_6	D_7	D_8	D_9	D_{10}	D_{s_1}	D_{s_2}	D_{s_3}	D_{s_4}	D_{s_5}	D_{s_6}	D_{s_7}
Dat_1 Dat_2	$Rest_1$ $Rest_2$	Abt Buy	Amz GPr.	DBLP ACM	IMDb TMDb	IMDb TVDB	TMDb TVDB	Wmt Amz	DBLP 学者	IMDb DBP	D_{10K}	D_{50K}	D_{100K}	D_{200K}	D_{300K}	D_1	D_{2M}
$\|V_1\|$ $\|V_2\|$	339 2,256	1,076 1,076	1,354 3,039	2,616 2,294	5,118 6,056	5,118 7,810	6,056 7,810	2,554 22,074	2,516 61,353	27,615 23,182	10K	50K	100K	200K	300K	1M	2M
$\|A_1\|$ $\|A_2\|$	7 7	3 3	4 4	4 4	13 30	13 9	30 9	6 6	4 4	4 7	12	12	12	12	12	12	12
$\|D\|$	89	1,076	1,104	2,224	1,968	1,072	1,095	853	2,308	22,863	8,705	43,071	85,497	172,403	257,034	857,538	1,716,102
$\|S\|$	18.67	198.64	792.43	133.29	81,49	71.48	104.16	103.35	115.57	54.04	84.32	84.21	84.34	84.30	84.30	84.31	84.32

(a) 清洁-清洁实体解析 (b) 脏实体解析 表 2：(a) 用于清洁-清洁实体解析的真实数据集，以及 (b) 用于脏实体解析的合成数据集，按总大小递增排列，显示实体数 ($|V_x|$)、属性数 ($|A_x|$)、重复项数 ($|D|$)，以及以字符数计的平均句子长度 ($|S|$)。

	数据集 1	数据集 2	总计配对	测试配对项	重复项	属性-属性
DSM_1	Abt	Buy	9,575	1,917	1,028	3
DSM_2	iTunes	亚马逊	539	110	132	8
DSM_3	DBLP	ACM	12,363	2,474	2,220	4
DSM_4	DBLP	学者	28,707	5,743	5,347	4
DSM_5	沃尔玛	亚马逊	10,242	2,050	962	5

表 3：有监督匹配任务中使用的数据集。

Buy.com 提取的产品 [20]。D_3 来自同一领域，匹配来自 Amazon.com 和 Google Base 数据 API（Google Pr.）的产品[20]。D_4 包含来自两个出版物存储库 DBLP 和 ACM 数字图书馆的书目数据 [20]。D_5、D_6 和 D_7 由三个独立的数据源组成，其中包含来自 imdb.com（IMDb）和 themoviedb.org（TMDb）的电影描述，以及来自 TheTVDB.com（TVDB）的电视节目 [36]。D_8 是产品匹配领域的另一个数据集，包含来自沃尔玛和亚马逊的产品描述 [33]。与 D_4类似，D_9 包含来自 DBLP 和 Google Scholar 的书目数据 [20]。最后，D_{10} 匹配来自 IMDb 和 DBpedia 的电影 [41], , 但与 D_5 和 D_6 中的 IMDb 数据源没有重叠。

所有这些数据集均可通过 Zenodo[2] 以 CSV 格式公开获取。其详细特征如表 2(a) 所示。请注意，所有数据集均对应于清洁-清洁实体解析 任务，也称为记录链接，其中输入包含两个各自不含重复项但可能存在重叠的数据源，目标是检测出它们共有的匹配实体 [10, 42]。

用于阻塞可扩展性评估的数据集。 为了评估阻塞的可扩展性，我们采用了表 2(b) 所示的数据集，这些数据集已广泛用于相关研究 [8, 19, 44]。它们对应于脏实体解析，即去重，其输入是一个包含重复项的单一数据源 [10, 42]。这些数据集由 Febrl [6], 按照以下方式人工生成：首先，从真实人口普查数据的频率表中提取真实姓名（名和姓）以及地址（街道号码、街道名称、邮政编码、郊区和州名），以创建初始的干净实体。随后，根据现实中的错误率和错误类型（例如插入、删除或替换字符或单词），随机生成重复实体。最终，所有实体中有 40% 至少与另一个实体匹配。每条记录最多有 9 个重复项，每个属性值和每条记录分别最多有 3 次和 10 次修改。

用于有监督匹配的数据集。 对于此任务，我们使用了与 [3], 中相同的五个数据集，这些数据集在相关文献中被广泛使用 [25, 33]。它们的技术特征报告于表 3；所有配对中有 60% 构成训练集，其余部分平均分配给验证集和测试集。其中大多数数据集源自表2(a)中的数据集：DSM_1 是 D_2 的一部分，DSM_3 是 D_4 的一部分，DSM_4 是 D_9 的一部分，而 DSM_5 是 D_8 的一部分。唯一的例外是 DSM_2，由于缺少完整的真实值，因此未包含在表2(a)中。另请注意，DSM_2-DSM_5，以增加其难度 [25]。

4.2 设置

在所有实验中，我们考虑每个实体的所有属性值，即每个实体都由其全部属性值的拼接表示。这些模式无关 设置能够天然处理属性值错置问题（例如人名与其职业相关联的情况），同时在阻塞 [39, 44] 和匹配 [3]方面都表现出较高的有效性。

我们在本次实验分析中使用的所有代码和数据集也都公开发布在上述代码库中。对于语言模型，我们使用了两个广受欢迎的 Python 软件包所提供的实现：Gensim[3] 和 Hugging Face[4]。前者提供 Word2Vec 和 FastText，后者则提供其他所有模型。所有实验均在一台配备 Ubuntu 20.04、AMD Ryzen Threadripper 3960X 24 核处理器、256 GB RAM 和 RTX 2080Ti GPU 的服务器上执行。在可能的情况下，我们使用 GPU，即用于动态模型的向量化、匹配和阻塞中的相似度分数计算、阻塞中的最近邻搜索，以及有监督匹配中动态模型的微调。

4.3 评估任务与方法论

我们在以下四项任务中评估这些语言模型。**向量化。** 该过程会将给定的每个文本实体转换为其嵌入向量。我们考虑模式无关实体解析，其中每个实体都表示为一个由其所有文本属性拼接而成的“句子”。对于仅支持词嵌入的 Word2Vec 和 GloVe，我们将该句子切分为词元，并对其向量求平均，以获得单个向量。FastText 会在内部将句子拆分为更小的 n 元语法，并将其嵌入聚合为单个向量。其余模型则会为整个句子生成一个嵌入。在此任务中，我们比较每个模型的执行时间。

阻塞。 给定输入数据集中的向量化实体，阻塞会生成一组候选对。对于每个输入实体，我们

[2]https://zenodo.org/record/6950980

[3]https://radimrehurek.com/gensim [4] https://huggingface.co

执行最近邻搜索（NNS），以查找与其最相似的 k 向量。对于表 2(a) 中的清洁-清洁实体解析数据集，我们执行精确最近邻搜索。对于两个数据集中较小的数据集中的每个实体，我们计算所有相似度分数，并返回 k 最近邻。对于表 2(b) 中的脏实体解析数据集，由于其规模显著更大，我们执行近似最近邻搜索。根据最先进水平，我们采用 [24], HNSW [29] 索引。这是一种基于图的索引，可作为 Delaunay 图的近似，同时还包含长距离链接，以支持小世界导航性质。为避免原始 NSW 中由高阶节点引发的连通性问题，HNSW 根据节点的度数引入多层链接分离，并采用高级启发式方法，以更好地选择每个节点的邻居。最终生成的索引提供了非常出色的查询时间，但代价是构建索引时会产生较大的开销。在我们的实验中，我们使用了 FAISS 提供的实现。[5] 首先，我们将所有输入实体向量化并建立索引。然后，我们使用每个实体查询该索引，e 以根据欧氏距离检索其 k 近似最近邻。

无监督匹配。这被视为一项聚类任务，其中每个实体聚类都对应一个不同的现实世界对象。对于清洁-清洁实体解析中的两个数据集，我们将该任务建模为二分图匹配，其中一个数据集中的每个实体最多与另一个数据集中的一个实体匹配。为解决这一问题，我们采用唯一映射聚类 (UMC) [21], ，它兼具高有效性和高时间效率 [40]。

整个流程如下。首先，我们使用以下公式计算所有实体对之间的相似度分数：$sim(e_i, e_j) = 1/(1 + dist(v_i, v_j))$，其中 $dist$ 表示实体 e_i 和 e_j 的嵌入向量 v_i 和 v_j 之间的欧氏距离。此处我们不执行阻塞，以避免其影响 UMC 的有效性。此处的执行时间仅测量 UMC 的运行时间，即假设所有相似度分数都已计算完成。UMC 按相似度分数从高到低遍历所有实体对，直到最小数据集中的所有实体都已匹配，或不再有实体对的相似度超过给定的相似度阈值 δ。此阈值是 UMC 唯一的配置参数。为对其进行微调，我们考虑所有

在 $[0.05, 0.95]$ 中以 0.05 为步长取值，并选择使 F度量 最大的值作为最优值。这样，我们的实验分析便考虑了每个语言模型的最大有效性，从而比较它们的潜力。然而，在实际设置中，为 UMC 指定最优相似度阈值是一项并不简单的任务。

为确保结果具有普适性，除 UMC 外，我们还测试了来自 [40] 的另外两种高性能算法：精确聚类，当两个实体互为最佳匹配时将其匹配；以及 *Kiraly* 聚类，它为最大稳定婚姻问题提供了线性时间近似。在这两种情况下，结果与 UMC 的结果均呈现出非常高的（>0.9）皮尔逊相关系数。这一点可以从图 2 中看出。

有监督匹配 这被视为一项二分类任务，将每个候选对分类为匹配或不匹配。通常会使用训练集、验证集和测试集，分别用于学习分类模型、选择其最优配置，以及评估其在新实例上的性能。

[5]https://faiss.ai/cpp_api/struct/structfaiss_1_1IndexHNSW.html

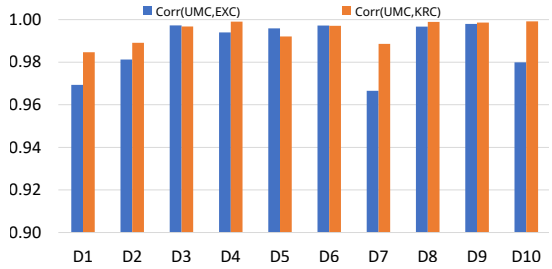


图 2：唯一映射聚类（UMC）、精确聚类（EXC）与 Kiraly 聚类（KRC）之间的皮尔逊相关系数。

为考察 BERT 和 SentenceBERT 模型的性能，我们将它们与 *EMTransformer* [3]结合。我们在开源的基于深度学习的匹配算法中选择了这种方法，因为它在完全依赖嵌入模型的同时，能够实现最先进性能。这与 DITTO [25], 形成对比，后者利用了外部信息，例如词性标注。然而，EMTransformer 与静态模型不兼容。为解决这一问题，我们将它们与此类模型的最先进方法 DeepMatcher [33]结合。

请注意，本分析纳入了 EMTransformer 或 DeepMatcher 支持的模型，并对代码进行了少量调整。因此，S-GTR-T5 和 Word2Vec 不在分析范围内，因为现有实现无法支持它们（EMTransformer 无法处理前者所需的序列到序列输入，而后者的格式不符合 DeepMatcher 的要求）。还请注意，EMTransformer 的原始实现忽略了验证集，直接在测试集上评估每个模型。然而，正如 [26] 中所指出的，这会导致过拟合。我们修改了代码，使其遵循文献中的标准方法：对于每个训练好的模型，使用验证集检查其是否能取得最高 F1 值，然后将该模型用于测试集 [25, 33]。本分析使用了表 3 所示的五个数据集。

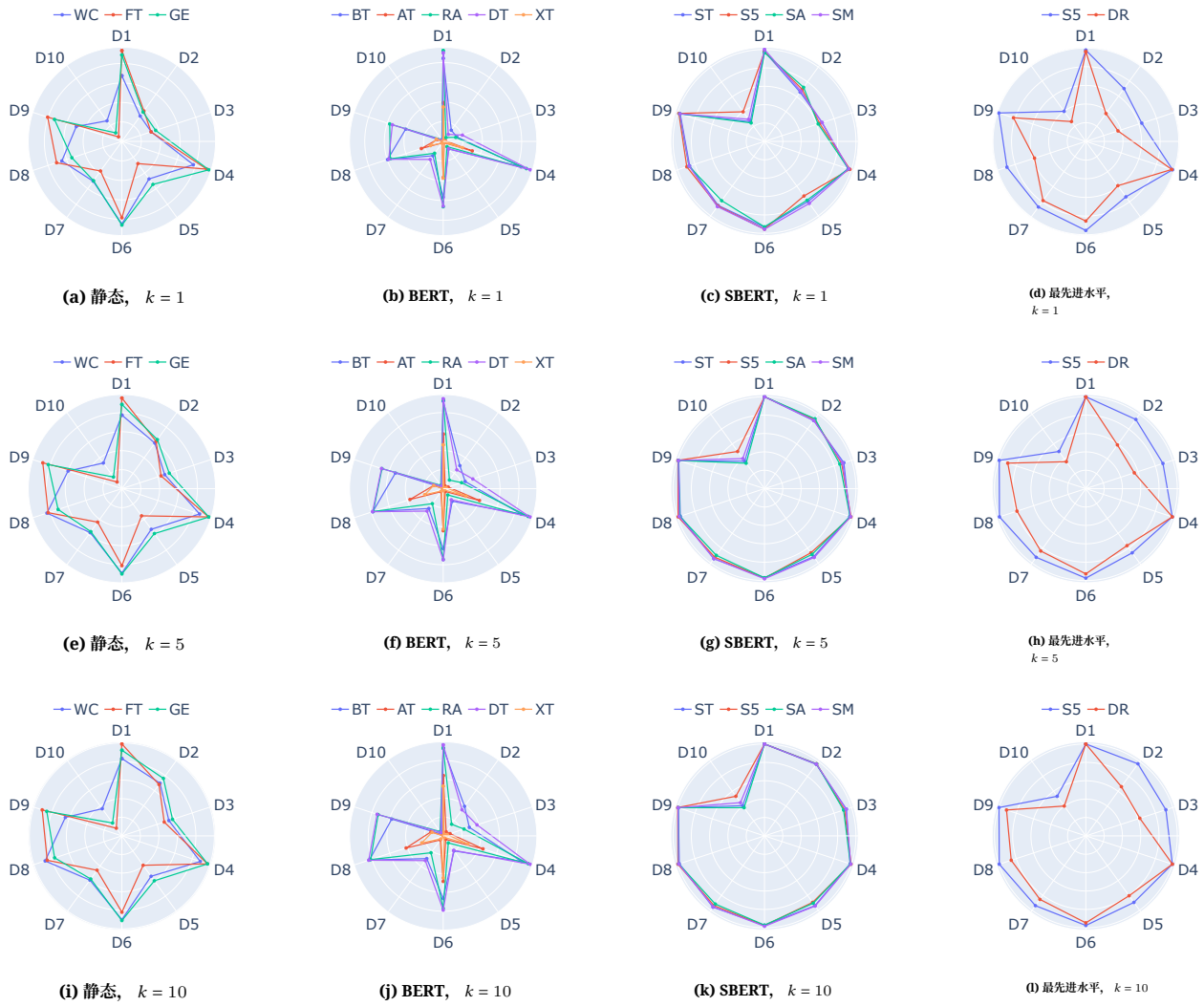
5 有效性比较

5.1 阻塞

我们测量生成的候选对的召回率，也称为候选对完整性 [8, 19, 39]。召回率是阻塞最关键的评估指标，因为它通常决定后续匹配步骤的上限；也就是说，阻塞召回率较低通常会导致更低的匹配召回率，除非采用复杂且耗时的迭代算法 [7, 15]。相比之下，阻塞后的精确率通常较低，这是由大量假阳性造成的，但经过匹配后会显著提高。因此，就精确率而言，所有模型的分母都相同（即候选项总数），可以省略精确率，因为召回率和精确率的表现相同。

各类模型的实验结果如图3所示，$k = 10$。具体而言：

在静态模型中，GloVe 在绝大多数情况下表现最佳，FastText 和 Word2Vec 分列第二和第三位。平均而言，GloVe 比 FastText 高出 19%，



(a) 静态， $k = 1$ **(b) BERT，** $k = 1$ **(c) SBERT，** $k = 1$ **(d) 最先进水平，** $k = 1$

(e) 静态， $k = 5$ **(f) BERT，** $k = 5$ **(g) SBERT，** $k = 5$ **(h) 最先进水平，** $k = 5$

(i) 静态， $k = 10$ **(j) BERT，** $k = 10$ **(k) SBERT，** $k = 10$ **(l) 最先进水平，** $k = 10$

图3：表2(a)中所有数据集上各模型的阻塞召回率。每行图对应于 $k \in \{1,5,10\}$ **的一个取值。**

但 D_1、D_8 和 D_9 除外，在这些情况下 FastText 领先。与 Word2Vec 相比，GloVe 的召回率平均高出 12.5%，但 D_7 D_8 和 D_{10} 除外（其中充斥着噪声和缺失值）。

在 BERT 模型中，我们可以看到，XLNet 和 AlBERT 在几乎所有数据集上的表现都非常差。XLNet 依赖置换语言建模（PLM），试图对词语和短语之间的依赖关系进行建模，而不是采用 BERT 使用的掩码语言建模（MLM）。在我们的任务中，这种方法效果较差，因为输入文本是通过串联多个不同属性构成的，并不构成连贯的句子。AlBERT 只训练一个编码器，以生成 BERT 的轻量版本，并与其余编码器共享权重。这在此处同样表现不佳。因此，这两个模型都存在判别性较差的问题，即它们会为匹配实体对和不匹配实体对都赋予较低的相似度分数。出于同样的原因，尽管程度较轻，同样的情况也适用于其余的 BERT 模型。这些模型的表现中规中矩，其中 DistilBERT 在所有数据集上都是表现最好的模型。

最后，所有 SentenceBERT模型在所有数据集上都实现了非常高的召回率，但极其嘈杂且稀疏的 D_{10}。此类别中最优的模型是S-GTR-T5。这与每个 SentenceBERT模型所依赖的基础模型有关。例如，GTR-T5 在包含超过 20 亿个配对的数据集上进行训练，而其他三个基础模型均在由多个数据集组成的集合上进行训练，这些数据集总计包含 10 亿+ 个配对。

最后，在不同组之间，我们可以看到，[49] 中提出的观点同样适用于我们的任务。未经微调的 BERT 模型在召回率方面的表现不如静态模型（GloVe），而 Sentence-BERT 的总体表现最佳。后者主要有两个原因：它们是为句子而非词嵌入设计的，并且是在规模更大的语料库上训练的。

总结。 上述模式总结于图 4 中，该图报告了每个模型在各数据集上按召回率排序的结果

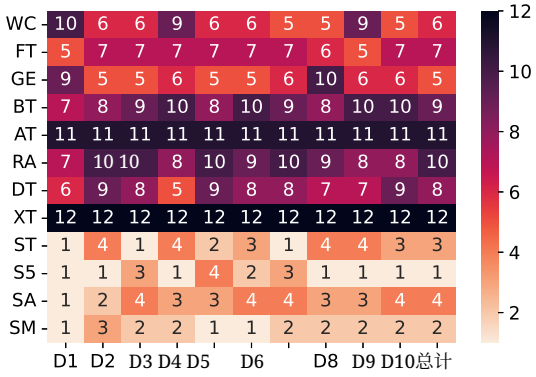


图 4：模型相对于阻塞召回率的排名（越低越好）。

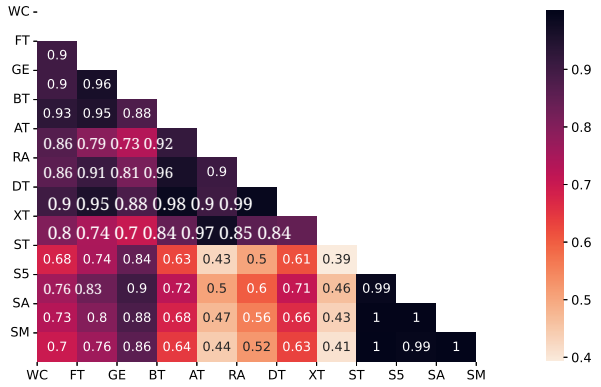
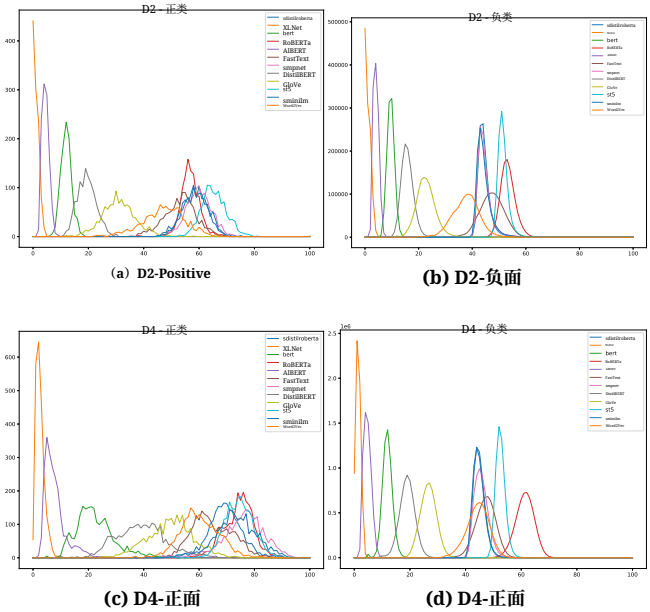


图 5：模型相对于阻塞召回率的皮尔逊相关系数。

对于 k=10，最右侧一列表示每个模型的平均排名。我们观察到，前四名均由 SentenceBERT模型 占据，其中 S-GTR-T5 在大多数数据集中排名第一。有趣的是，这些模型中没有一个低于第四名。SentenceBERT 模型表现优异有两个原因：(a) 它们天生能够将句子转换为嵌入向量，而另外两类模型则是为单个词元的向量化而设计的。(b) 它们封装了来自更广泛语料库的知识，同时其最终层在预训练形式下已经具有合理的权重（不同于 BERT 模型）。

接下来的三个排名位置大多由静态模型占据，其中 GloVe 的平均排名最高。不过，FastText 是最稳定的模型，排名在第 5 至第 7 位之间波动，而另外两个模型则会跌至第 10th 位。最后五个排名位置则由基于 BERT 的模型占据。DistilBERT 表现最佳，平均排名为第 8th 位；AlBERT 和 XLNet 则始终分别排名第 11th 位和第 12th 位。BERT 模型的表现不如静态模型，因为它们缺乏区分能力，这是由于没有进行微调；微调能够确保其具备上下文感知功能。它们最终层中的预设权重会给大多数实体对赋予极低的分数，无论这些实体对是否匹配。静态模型则不同，尽管它们只具备上下文无关功能，且维度较低。



(a) D2-Positive (b) D2-负面

(c) D4-正面 (d) D4-正面

图6：在匹配（正面）和不匹配（负面）类别中，D2 和 D4 情况下所有模型的判别性。

静态模型则不同，尽管它们只具备上下文无关功能，且维度较低。

这些模式表明，属于同一类别的模型在有效性方面具有高度相关性。事实上，如图5所示，就 k=10 的召回率而言，每个类别中模型之间的皮尔逊相关系数相当高（≥ 0.9）。预训练静态模型与 SentenceBERT 模型之间的相关性同样很高，这是由于这两个类别的性能都很出色。相比之下，基于 BERT 的模型与另外两个类别之间的相关性明显较低：对于 SentenceBERT模型，相关系数在 0.58 到 0.85 之间波动；对于静态模型，则在 0.69 到 0.86 之间波动。在后一种情况下，DistilBERT 是一个例外，其相关系数在 0.84 到 0.94 之间波动，因为它是性能最佳的 BERT 模型，因此与静态模型更为接近。

所有基于BERT的模型在阻塞和无监督匹配任务中的判别性或多或少都较差。这可以归因于缺乏能够确保其上下文感知功能的微调。无论实体对是否匹配，其最终层中的预设权重都会为大多数实体对产生非常低的分数。尽管静态模型不具备上下文感知功能且维度较低，但这一点并不适用于静态模型。图 6 展示了这一点，其中描绘了每个语言模型在数据集 D2 和 D4 中的相似度分数分布（每个数据集的正类显示在左列，负类显示在右列）。我们观察到，SentenceBERT模型具有较高的判别性，静态模型的判别性较低，而基于BERT的模型无法将这两种分布区分开来。

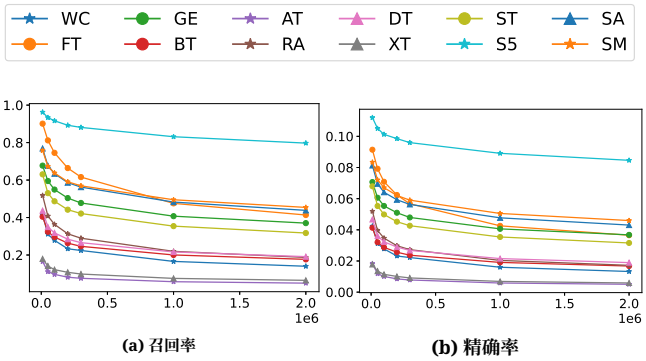


图 7：表 2(b) 中合成数据集上的可扩展性和有效性。横轴表示输入实体的数量。

与最先进水平的比较。 图 3 最右列将表现最佳的语言模型 S-GTR-T5 与一种基于深度学习和嵌入向量的最先进阻塞方法进行了比较：DeepBlocker 的自编码器搭配 FastText 嵌入。S-GTR-T5 的召回率始终显著高于 DeepBlocker。唯一的例外是 D_1 和 D_4，在这两种情况下，两种方法都达到了近乎完美的召回率。原因在于，D_1 中的重复对数量相对于每个数据源的规模而言非常少，而 D_4 包含相对干净且具有较长文本描述的实体，因此易于匹配。在其余 8 个数据集中，S-GTR-T5 的召回率比 DeepBlocker 高 15%。

这似乎可以归因于 DeepBlocker 使用的 FastText 嵌入。然而，DeepBlocker 是一种综合性的阻塞方法，它使用 FastText 的方式比 S-GTR-T5 单纯的最近邻搜索更复杂。例如，DeepBlocker 的一个关键组件是自监督，它会自动为随机抽取的候选对样本打标签，以训练其分类模型。因此，DeepBlocker 是一种随机方法，而 S-GTR-T5 则只执行最近邻搜索。 [55] 中的消融分析表明，DeepBlocker 的所有组件都对最终结果有显著贡献。因此，语言模型的作用有限，DeepBlocker 的性能与 FastText+最近邻搜索的相关性也较低。

5.1.1 **可扩展性。** 阻塞必须能够扩展到大规模数据量，以便将匹配的输入限制在可管理的范围内，即使面对包含数百万个实体的情况也是如此。因此，我们需要评估所选语言模型在输入数据规模增加时的扩展能力。为此，我们使用表 2(b) 中的七个数据集进行分析。结果如图 7 所示。

图 7(a) 显示，从 D_{10K} 到 D_{2M}，所有模型的召回率都持续下降。这是意料之中的，因为候选项数量会随着输入数据规模的增大而呈二次方增长。显然，S-GTR-T5 的表现最佳：它在 D_{2M} 上的召回率相当令人满意，无论从绝对值（0.800）来看，还是与在 D_{10K} 上的初始召回率（0.962）相比都是如此，降幅仅为 17%。S-GTR-T5 的训练语料库规模大得多，且内容也更加丰富。这为 S-GTR-T5 带来的优势，在使用合成数据集进行测试时更加明显；由于候选对数量大得多，这些数据集更具挑战性。在大多数数据集中，排名第二的方法是 FastText，其召回率下降了 54%，从 0.901 降至 0.415。另一端则是 AlBERT 和 XLNet：它们在所有数据集上的召回率都低于 0.18（即使在 D_{10K} 上也是如此），且在 D_{2M} 上的召回率下降了三分之二。其余模型的表现介于这两个极端之间，并可根据性能分为三组。表现最好的一组包括 S-DistilRoberta 和 S-MiniLM，它们在最小的数据集上的初始召回率为 ~0.750，最终降至 0.450，降幅为 ~40%。表现最差的一组包括 Word2Vec、DistilBERT、BERT 和 RoBERTa，它们的召回率下降了 56%–66%，在 D_{2M} 上的召回率远低于 0.2。GloVe 和 S-MPNet 处于前两个极端之间：它们的召回率降幅不到 50%，且在 D_{2M} 上高于 0.32。

关于图7(b)中的精确率，我们观察到几乎相同的模式。这是由于前文所述的两项指标之间存在线性关系。只有在脏实体解析的情况下，由于冗余候选对数量不同，会出现轻微变化；这些冗余候选对只计数一次（如果 e_j 被包含在 e_i 的最近邻中，反之亦然，则候选对 $<e_i, e_j>$ 是冗余的）。因此，随着输入规模增加，所有模型的精确率都会逐渐下降。在大多数情况下，这一降幅比召回率的降幅高出 2-3%，因为召回率的分母（即现有重复项的数量）增长速度低于精确率的分母（即不同候选对的数量）。

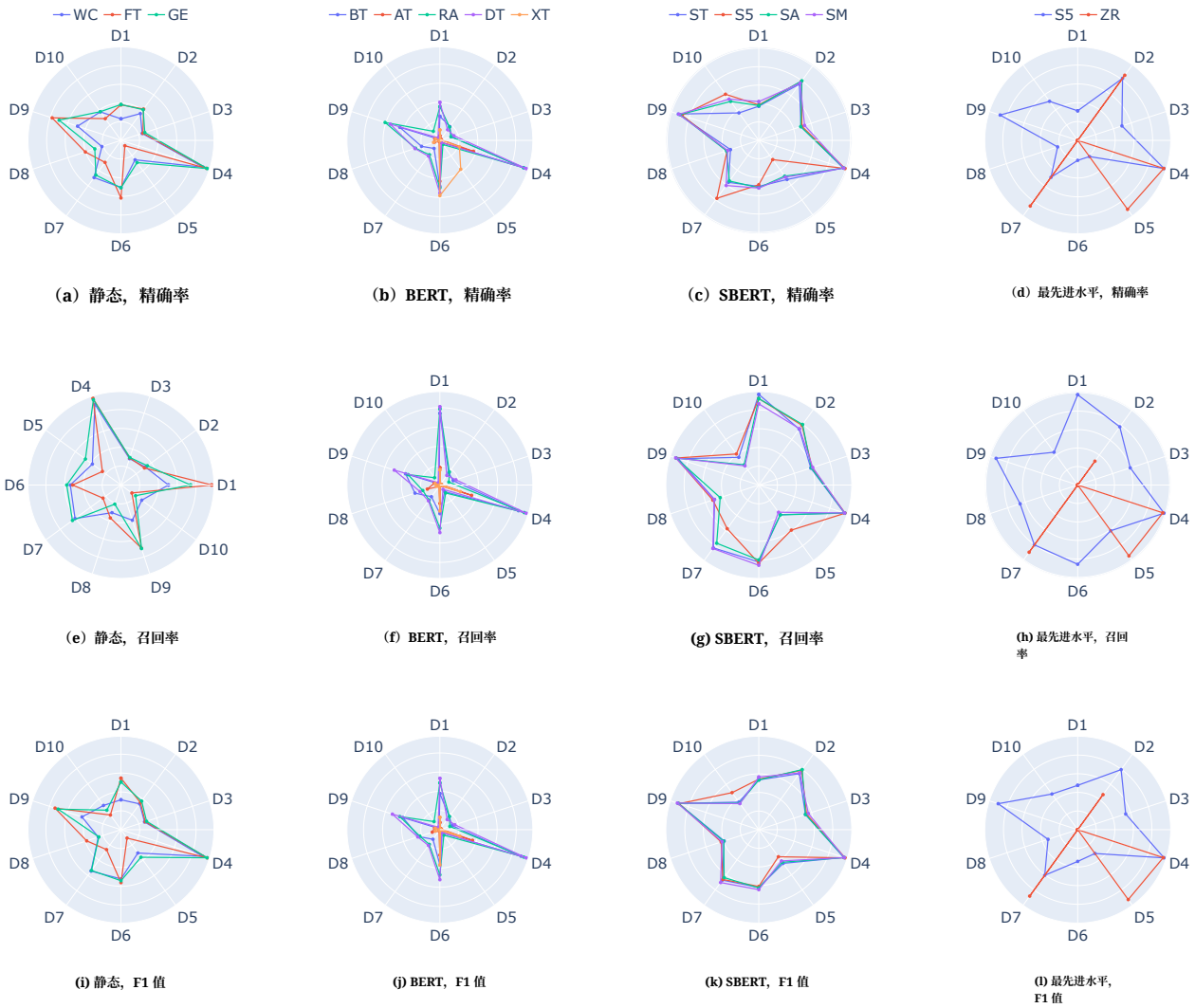
5.2 无监督匹配

图 8 展示了所有模型在此任务中的表现。

在静态模型中，根据与最高 F 度量（F1）的平均距离，GloVe 表现最佳（31.9%），其次是 FastText（37.4%）和 Word2Vec（39.4%）。从绝对值来看，除书目数据集外，它们在所有数据集上的 F1 值都相当低：在 D_4 中，F1 值超过 0.9，与 SentenceBERT 模型非常接近；但在 D_9中，只有 FastText 和 Glove 的 F1 值超过 0.7。原因在于，来自 Google Scholar 的实体噪声更多，涉及的术语也比 D_4多得多。在所有其他情况下，它们的 F1 值都（远）低于 0.57。

在基于BERT的模型中，XLNet 和 AlBERT 的表现始终最差，它们在任何数据集上的 F1 值都不超过 0.37。平均而言，它们的 F1 值比表现最佳的模型低近一个数量级（~87%）。原因与阻塞中的情况相同：两者都是针对不同任务训练的，若不进一步微调，就无法在匹配任务中取得良好表现。与之相反，DistilBERT 和 RoBERTa 的平均距离为 ~55%。最后，BERT 的表现介于这两个极端之间，与最佳模型的平均距离为 62%。这三个模型在干净且简单的 D_4上取得了尚可的 F1 值（~0.9），但在所有其他情况下都（远）低于 0.54。

在 SentenceBERT模型中，表现最佳的是 S-GTR-T5，它在 10 个数据集中的 7 个上取得了最高的 F1 值。在其余数据集中，它排名第二或第三，与表现最佳的方法非常接近。平均而言，它与最高 F1 值的距离仅为 0.7%，在所有模型中最低。最差的情况对应于 D_6，其结果比最佳方法低 4.9%。第二佳模型是 S-MiniLM，它在 3 个数据集上取得了最高的 F1 值，并且



（a）静态，精确率　（b）BERT，精确率　（c）SBERT，精确率　（d）最先进水平，精确率

（e）静态，召回率　（f）BERT，召回率　(g) SBERT，召回率　(h) 最先进水平，召回率

(i) 静态，F1 值　(j) BERT，F1 值　(k) SBERT，F1 值　(l) 最先进水平，F1 值

图 8：表2(a)中所有数据集上各模型的精确率、召回率和 F1 值。

与最高 F1 值的平均距离第二低（6.5%）。接下来表现最好的两个模型是 S-MPNet 和 S-DistilRoBERTa，它们与最高水平的平均距离为 10.5 %。

从绝对值来看，它们在书目数据集 D_4 和 D_9 上的表现最佳，其 F1 值始终远高于 0.9。原因在于，较长的标题和作者列表有助于区分匹配实体与不匹配实体。此外，它们在 D_2 上的 F1 值也很高（~0.8）；该数据集将较长的文本描述与两个数据源之间的 1-1 匹配结合起来（即，一个数据源中的每个实体都与另一个数据源中的一个实体相匹配）。在其他所有数据集上，由于其中包含大量噪声，它们的 F1 值介于 0.35（D_{10}）和 0.77（D_7）之间。

最后需要强调的是，与阻塞阶段一样，S-GTR-T5 的表现优于另外两类中最优秀的模型，因为其他所有模型的 F1 值都没有超过 0.5——相对干净且容易的 D_4 除外。不过，与阻塞不同，在匹配阶段我们可以进行微调，以检验 BERT 模型是否能够比另外两类模型表现得更好。更多详情请参见第 5.3 节。

总结。 图 9 总结了每个数据集上各模型的排名位置。SentenceBERT模型通常在第 1 至第 4 位之间波动，静态模型位于第 5 至第 7 位之间，而基于 BERT 的模型位于第 8 至第 12 位之间。此外，图 10 中它们的 F1 值的皮尔逊相关系数显示，SentenceBERT模型之间几乎完全相关，静态模型组和基于 BERT 的模型组内部也具有非常高的相关性。后两类模型与 SentenceBERT模型的相关性较弱。静态模型与另外两组模型具有中等程度的相关性（$0.7 - 0.9$）。

总体而言，三种模型类型的相对性能遵循与阻塞相同的模式，原因也相同。SentenceBERT模型的表现优于所有其他模型，因为它们本来就是为整个“句子”向量化而设计的，而且训练所用的语料库规模大得多。处于另一极端的是

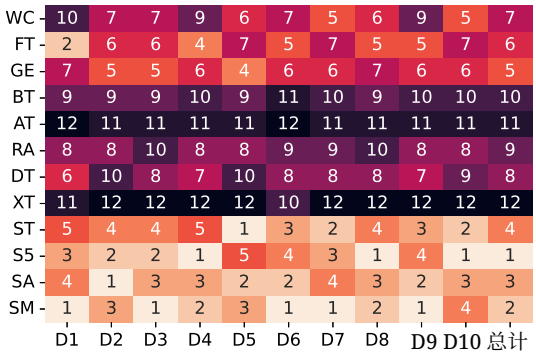


图 9：各数据集上每个模型相对于无监督匹配 F1 的排名位置。排名越靠前越好。

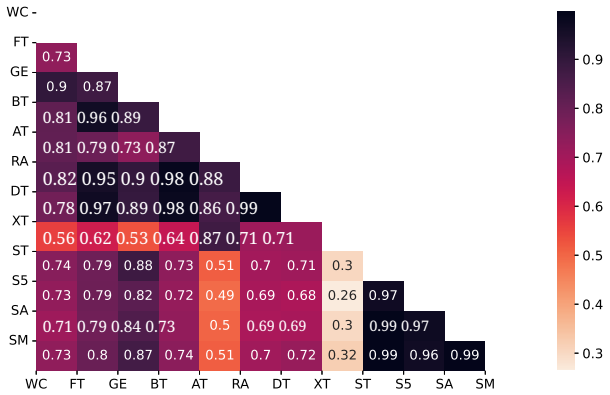


图 10：语言模型与无监督匹配 F1 的皮尔逊相关系数。

缺乏微调的 BERT 模型，其最终层中的预定义权重导致匹配实体对和非匹配实体对的相似度都很低。静态模型则处于这两个极端之间，这是因为它们采用了上下文无关的词级嵌入。

与最先进水平的比较。无监督匹配领域最先进的方法是 ZeroER [61], , 它将每一对实体转换为一个特征向量，其维度对应于相似度函数。其核心假设是，生成的特征向量服从含有两个混合成分的高斯混合模型（每个匹配类别对应一个成分）。该方法利用自适应特征正则化来避免过拟合，并通过传递性提高准确率。ZeroER 使用 Magellan 的重叠阻塞，将搜索空间缩减为一小组候选对。

我们将 ZeroER 与一个基于用于阻塞和匹配的最佳语言模型 S-GTR-T5 的端到端框架进行比较。实际上，我们使用上述匹配算法，默认将相似度阈值设为 0.5；但不同于使用所有实体对，我们仅允许最小实体集合中的每个实体拥有 k=10 个候选项，这些候选项由精确最近邻搜索执行的阻塞生成。

两种方法的相对性能如图 8(d)所示。对于其中一半的数据集，ZeroER 没有可估算的性能，因为它在运行 6 小时后仍未终止，而 S-GTR-T5 始终耗时不到 1 分钟，如表5(b)所示。在 D_4 中，由于数据相当干净且任务相对简单，两种方法的性能几乎同样出色。在 D_1 和 D_2 中，S-GTR-T5 的性能显著优于 ZeroER。ZeroER 在 D_1 上实际上得到 F_1=0，因为 D_1 包含许多缺失值和错置值，而 ZeroER 的基于模式的功能无法支持这些值（不同于 S-GTR-T5 的模式无关设置）。 D_2 包含大量文本值，这些值也几乎不适用于 ZeroER 所采用的大多数相似度度量。相比之下， D_5 和 D_7 包含用于描述电影的短属性值（例如演员姓名）。这些值非常适合 ZeroER 的特征，因此其有效性远高于 S-GTR-T5；后者并非针对罕见的、特定领域的（术语性）文本值而设计。

总体而言，尽管 S-GTR-T5 不需要参数调优，但在大多数数据集上，其表现显著优于或至少与 ZeroER 相当，同时速度快了几个数量级（参见第 6.3 节）。

5.3 有监督匹配

图 11(a)-(c) 展示了所有模型在表 3 所列数据集上执行该任务时的 F1 值。我们观察到，可以根据上下文感知能力将语言模型划分为两组。动态模型始终展现出最高的性能。就有效性而言，RoBERTa 实际上是最稳健的模型。它在大多数数据集上取得最高的 F1 值，平均排名第一。事实上，在所有数据集中，其与最高 F1 值的平均距离仅为 0.5%。紧随其后的是表现第二好的模型 S-MPNet，该模型在一个数据集（DSM_3）上排名第一，其与最高 F1 值的平均距离为 0.7%。其余模型按照每个数据集的最大 F_1 的平均距离排序如下：BERT、AlBERT、XLNet、S-DistilRoBERTa、S-MiniLM。

不过，即使是有效性最低的动态模型，平均而言也只比最佳模型差 5%。这是因为所有纳入考察的模型之间存在很强的相关性：某个模型在特定数据集上具有较高的有效性，意味着其余模型的有效性也同样较高。这一模式应部分归因于 DSM_3 和 DSM_4 数据集，在这些数据集中，F1 值的最大值与最小值之差小于 1.3%。尽管这两个数据集都以错置值的形式加入了人工噪声，但其中的值相对干净。这意味着，在我们所考察的模式无关设置中，重复实体共享多个相同的词元。因此，即使是使用字符串相似度度量作为特征的经典机器学习分类器，也能在这些数据集上取得非常高的 F1 值（>0.9）（参见 [25, 33] 中的 Magellan）。由于涉及的术语（例如产品名称）不同，其余数据集更具挑战性。因此，动态模型的 F1 值最大值与最小值之差介于 4.9%（DSM_1）和 17%（DSM_2）之间。

第二组模型包括静态且与上下文无关的模型，即使与 DeepMatcher 的最佳配置相结合，其表现仍相对较差；DeepMatcher 是针对该任务和模型类型的当前最先进算法。实际上，GloVe 和 FastText 都与一个两层全连接 ReLU HighwayNet 分类器相结合，随后接入一个 softmax 层，该分类器位于



(a) 静态 (b) BERT (c) SBERT (d) 最先进水平

图11：表 3 中所有数据集上各模型的有监督匹配 F度量。

分类模块中，并与属性相似度向量模块的混合模型结合使用。平均而言，GloVe 和 FastText 的表现分别比每个数据集上的最佳模型低 22% 和 37%。只有在 DSM_3 和 DSM_4 中，由于所有动态模型的表现都超过 0.95，这两个静态模型才展现出较高的性能，其 F1 值与最高值相差不超过 5%（这进一步表明这两个数据集的书目数据所提出的是较为直接的匹配任务）。需要注意的是，FastText 在所有数据集上的表现始终优于 GloVe，因为其字符级功能能够处理由各数据集的领域特定术语所产生的词汇表外词元，而 GloVe 则无法做到这一点。

总体而言，静态模型在大多数情况下的表现不如动态模型，正如文献 *[25]*, 所报道的那样；而基于*BERT*的模型与*SentenceBERT*模型的匹配效果相当，这与之前的实体解析任务不同，原因在于它们对最后一层进行了微调。*SentenceBERT*模型也受益于微调，但受益程度较小，这可能是因为用于表示每个实体的句子以临时方式构建，缺乏连贯性。

与最先进水平的比较。图11(d)展示了利用静态和动态模型的最先进的监督匹配算法 DeepMatcher+ [18] 和 DITTO [25]的性能。实际上，我们采用的是 [25]中报告的优化后 F1 值。

将 DITTO 与动态模型进行比较时，我们发现，在每个数据集中，其 F1 值都可以直接与表现最佳的语言模型相比较。在 DSM_2、DSM_3 和 DSM_4中，DITTO 的 F1 值仅低 ≤0.5%。不过，在 DSM_1 和 DSM_5中，DITTO 分别以 3% 和 1.5% 的优势超过了所有语言模型。这应归因于外部知识和数据增强；与作为 DITTO 核心的语言模型 RoBERTa 进行比较时，这两者的作用更加明显。平均而言，在所有数据集上，后者的表现比 DITTO 低 1.3%。

将静态模型与DeepMatcher+进行比较后，我们发现，在大多数数据集上，其性能与FastText几乎相同，因为它采用了相同的语言模型。只有在 DSM_2 和DSM_5上，DeepMatcher+ 的表现才显著更好，分别高出9%和23%。这应归因于其相较于原始DeepMatcher算法的优势，即DeepMatcher+ 将该算法与迁移学习和主动学习相结合。值得注意的是，DeepMatcher+ 始终优于GloVe，平均高出28%。

另请注意，DeepMatcher+ 在几乎所有情况下的表现都不如动态模-型。这一点在 DSM_1 和 DSM_5中尤其明显：表现最差的动态模型（DistilBERT）仍以 ~20%的优势超过了Deep-Matcher+ 。这证实了动态模型在有监督匹配方面优于静态模型，这是因为动态模型能够进行微调，根据当前数据优化其最后一层的权重。

	WC	FT	GE	BT	AT	RA	DT	XT	ST		S5	SA	SM
初始化	32.4	159.7	5.87	4.72	3.99	5.28			4.3	4.73	9.19	9.84	9.33	8.36
D1	0.0	0.2	1.9	2.6	2.4	2.3	1.3	4.0	1.1	1.1	0.7	0.5		
D2	0.1	1.6	0.2	3.1	2.4	2.3	2.2	3.3	3.4	3.4	1.8	0.9		
D3	0.9	9.6	0.4	10.1	6.7	6.3	8.6	8.3	10.3	12.4	5.8	2.3		
D4	0.2	2.5	0.3	5.9	5.2	5.3	4.2	7.8	5.1	5.4	2.8	1.4		
D5	0.4	3.8	0.4	13.6	13.0	12.9	8.7	20.3	10.7	12.1	6.0	3.2		
D6	0.6	5.5	0.5	15.4	14.3	13.8	10.4	21.3	14.9	17.2	8.2	3.9		
D7	0.4	3.6	0.4	11.9	11.2	11.4	8.0	17.7	9.7	10.4	5.3	2.8		
D8	1.0	10.0	0.8	28.8	25.0	24.2	19.5	38.9	28.5	27.3	14.9	6.7		
D9	2.4	27.7	1.9	73.4	66.0	65.5	49.9	99.9	58.0	61.5	31.4	16.0		
D10	1.1	10.6	1.2	51.1	49.1	47.9	31.6	78.9	28.9	30.2	16.5	10.3		

表4：每个模型和数据集的向量化时间（秒）。

6 效率比较

6.1 向量化

初始化。 每个模型的初始化时间显示在表4的第一行中。它指的是将必要数据结构加载到主内存中所需的时间（例如，静态模型所需的词典，以及动态模型所需的学习得到的神经网络），与所使用的数据集无关。静态模型由于需要将哈希表加载到主内存中，以将词元（或字符 n 元组）映射到嵌入向量，因此效率较低。对于动态模型，我们观察到，基于 BERT 的模型比 SentenceBERT 模型快得多，因为它们的平均运行时间分别为 4.7±0.8 秒和 8.9±0.6 秒。这是由于后者使用了规模更大、结构更复杂的神经模型。

转换。 表4的其余部分显示了每个模型将各数据集中的实体转换为稠密嵌入向量（初始化之后）所需的总计时间。Word2Vec 和 GloVe 是迄今为止速度最快的模型，在几乎所有情况下都表现出最低的处理运行时间。对于每个数据集而言，Word2Vec 的速度比效率排名第三的模型快一个数量级

，后者交替对应于 FastText 和 S-MiniLM。除 D_1 外，GloVe 的表现至少比这两个模型快 6 倍。就绝对时间而言，Word2Vec 和 GloVe 处理最小的八个数据集（D_1-D_8）所需时间远低于 1 秒，而处理两个较大的数据集所需时间也不到 2.5 秒。

在基于BERT的模型中，正如预期，DistilBERT明显快于BERT，其处理时间平均低33%。不过需要注意的是，除 D_3外，它平均比FastText慢>50%。该类别中效率次高的模型是AlBERT和RoBERTa，它们的平均表现分别比BERT快11%和13%。XLNet是耗时最长的基于BERT的模型，在除 D_3之外的所有数据集中，其平均耗时比BERT高 ~30%。这可以解释为 D_3 包含更长的句子，如表2(a)所示。

在SentenceBERT模型中，S-MiniLM的耗时最低。如上文所述，它与FastText并列为第三快的模型。该类别中表现第二好的模型是S-DistilRoBERTa，其平均慢30%。平均而言，这两个模型分别比BERT快63%和53%。相比之下，较为复杂的S-GTR-T5学习模型在除最小数据集之外的所有数据集中，都产生了所有模型中最高的处理时间。S-MPNet则处于这两个极端之间。

总结。 静态模型在处理时间方面表现出色，但初始化时间非常长。更值得注意的是，FastText 是所有数据集中速度最慢的模型，但 D_9除外，在该数据集中，S-GTR-T5 展现出高得多的处理时间。这意味着，对于表2(a)中这类仅包含几千个实体的数据集而言，FastText 较高的初始化成本并未带来相应收益。平均而言，FastText 每个数据集需要 2 分钟。所有其他模型都能在不到 1 分钟内完成所有数据集的向量化，但有少数例外：BERT 超过 D_9，XLNet 超过 D_9-D_{10}，以及 S-GTR-T5 超过 D_8-D_{10}。

6.2 阻塞

对于所有模型，阻塞的执行时间都非常短，在大多数情况下不超过 0.5 秒。唯一的例外是两个最大的数据集 D_9 和 D_{10}，但即使是它们，在所有情况下所需时间也不到 2 秒。不同模型之间的差异并不显著。在大多数情况下，这些范围的较低端对应于维度最低的语言模型，即静态模型（300）以及 S-MiniLM（385），以及

高于采用768维向量的其余模型（见表1）。具体而言，FastText和S-GTR-T5始终分别是运行速度最快和最慢的模型。不过，与表4中的向量化时间相比，这一额外时间可以忽略不计。

与最先进水平的比较。 表5(a)报告了图3最右列对应的运行时间。我们观察到，在除 D_{10}之外的所有数据集中，DeepBlocker对于 k=1 和 k=5 始终比S-GTR-T5更快。S-GTR-T5运行缓慢的原因在于其较高的向量化成本，该成本占阻塞总时间的 ~99%。这解释了为什么在每个数据集中，其运行时间在 k的所有取值下几乎保持稳定。不过，这也是意料之中的，因为S-GTR-T5采用768维嵌入向量，而DeepBlocker采用300维FastText向量。然而，对于 k=10，DeepBlocker仅在 D_2 和 D_3 中比S-GTR-T5更快（快了

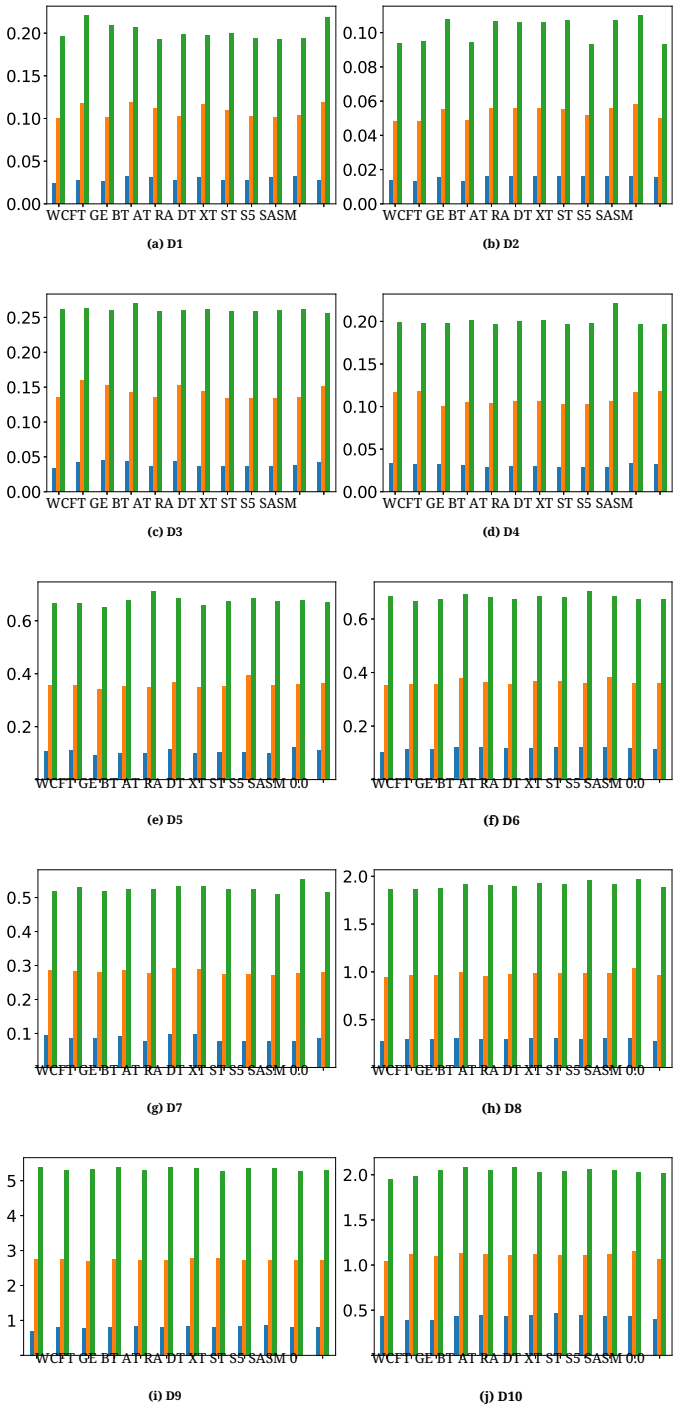
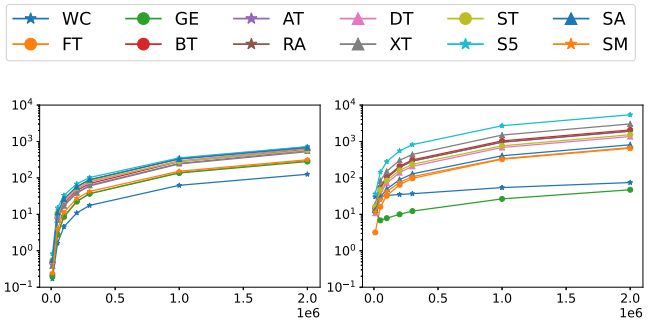


图12：各方法的阻塞时间（秒）。有三个取值： $k \in \{1、5、10\}$。

分别为 14.4% 和 26%）。在 D_1 和 D_4-D_8中，情况则相反，S-GTR-T5 平均快 14.1%。这是因为随着每个查询实体的候选项数量增加，DeepBlocker 的扩展性不佳，



(a) 阻塞时间（秒，对数刻度） (b) 向量化时间（秒，对数刻度）

图13：表 2(b) 中合成数据集的可扩展性。横轴表示输入实体的数量。

	DeepBlocker			S-GTR-T5			ZeroER		S-GTR-T5	
	$k=1$	$k=5$	$k=10$	$k=1$	$k=5$	$k=10$	t_p	t_m	t_p	t_m
D1	8	8	17	11	11	11	2	1	13	1
D2	9	9	17	13	13	13	1,728	71	14	3
D3	19	19	36	22	22	23	-	-	23	4
D4	13	13	28	15	15	16	9,595	113	16	9
D5	24	25	63	22	22	23	1,291	10	23	12
D6	28	28	68	27	27	28	-	-	28	16
D7	22	22	49	20	22	21	1,599	338	21	13
D8	46	46	46	37	38	39	-	-	38	8
D9	111	111	270	72	74	77	-	-	72	10
D10	154	150	371	41	41	42	-	-	46	96

(a)阻塞 (b)无监督匹配 表5：S-GTR-T5与（a）用于阻塞的DeepBlocker以及（b）用于无监督匹配的ZeroER的比较。除最右列以毫秒为单位外，所有列均以秒为单位。 t_p (t_m)分别表示预处理（匹配）时间。

这是由于其核心深度神经网络的复杂度很高。最重要的是，随着输入数据规模增大，DeepBlocker 的扩展性较差：在 D_{10}中，对于 k∈{1, 5}，S-GTR-T5 的速度快 1.5 倍；对于 k=10，快 3.7 倍。

6.2.1 可扩展性 图13(a)显示，所有模型的阻塞时间均呈超线性但次二次增长：输入实体数量从 D_{10K} 增加到 D_{2M}，增长了200倍，运行时间最多增加了1,435倍。除效率最高的两种模型 Word2Vec（742倍）和 S-GTR-T5（873倍）外，其他模型的增长幅度介于1,053倍（SMPNet）和1,435倍（XLNet）之间。这主要是因为 FAISS(HNSW) 以较高的索引时间换取显著较低的查询时间，即前者占主导地位。索引过程中需要执行涉及长路径的复杂基于图的操作。输入规模越大，这些路径就越长，遍历和处理它们的成本也会超线性增长。

图13(b)报告了向量化时间的变化情况：对于所有模型，随着输入数据规模增大，向量化时间呈次线性增长。对于所有基于BERT的模型，增长幅度介于127倍（DistilBERT）和165倍（XLNet）之间，因此仍远低于输入规模的增幅（即从 D_{10K} 到 D_{2M} 增长200倍）。S-GTR-T5也表现出类似的行为，但其余SentenceBERT模型展现出更好的可扩展性，因为它们的增长分别降至59倍（S-MiniLM）、94倍（S-MPNet）和62倍（XLNet）。这应归因于各模型的初始化过程，该过程与输入规模无关，并且占据了每个模型总体向量化时间的很大一部分，如表4所示。初始化的相对成本越高，随着输入规模增大，向量化时间的增长幅度就越小。因此，Word2Vec和GloVe的运行时间分别仅增加2.5倍和4倍。前者实际上在达到 D_{300K} 之前基本保持稳定，因为在规模最小的五个数据集上，其向量化时间主要由较高的初始化时间所决定。处于另一极端的是FastText，它是唯一一个随着输入数据规模线性扩展的模型（增长205倍），这得益于其字符级功能。

从绝对时间来看，GloVe 在所有数据集中始终是最快的模型，其次是 D_{10K}，而 Word2Vec 排名第二，从 D_{200K} 开始。它们分别在 0.8 和 1.3 分钟内完成向量化 D_{2M}。FastText 是三个最小数据集中第二快的模型，但在较大的数据集中，其速度接近最快的动态模型 S-MiniLM。两者处理 D_{2M} 都需要 ~11 分钟。另一极端是 S-GTR-T5，它始终表现出最慢的向量化速度，而 XLNet 在所有数据集中都是第二慢的模型。对于 D_{2M}，它们分别需要 90.3 和 50.6 分钟。

6.3 无监督匹配

我们将模型的匹配时间定义为使用最优剪枝阈值对所有成对相似度应用UMC算法所需的时间。对于最有效的模型S-GTR-T5，其匹配时间通常远低于半秒，但最大的数据库集 D_{10} 除外，其匹配时间会增加到接近两秒。在候选对数量庞大的情况下，这种高效率应归因于其相当高的相似度阈值，该阈值在0.55至0.70之间波动，其中位数（均值）为0.625（0.615），因此剪除了绝大多数候选对。其余SentenceBERT模型也表现出类似的行为。在大多数数据集中，静态模型也能在几秒内应用UMC，具体取决于其相似度阈值。基于BERT的模型则较少能够做到这一点，因为它们的区分能力较低，从而产生更低的阈值，并需要处理数量显著更多的候选对。因此，在大多数数据集中，最慢的模型是基于BERT的模型之一，通常为AlBERT或XLNet。

与最先进水平的比较。表5(b)显示，ZeroER 的运行时间主要由其阻塞时间构成；在大多数数据集中，这一时间甚至超过了 S-GTR-T5 的总执行时间。由于功能简单，端到端的 S-GTR-T5 方法在所有数据集上都比 ZeroER 快两个数量级，但 D_1。其运行时间主要花在输入实体的向量化和索引上；由于采用了阻塞，即使在最大的数据集上，匹配时间也只有几毫秒。

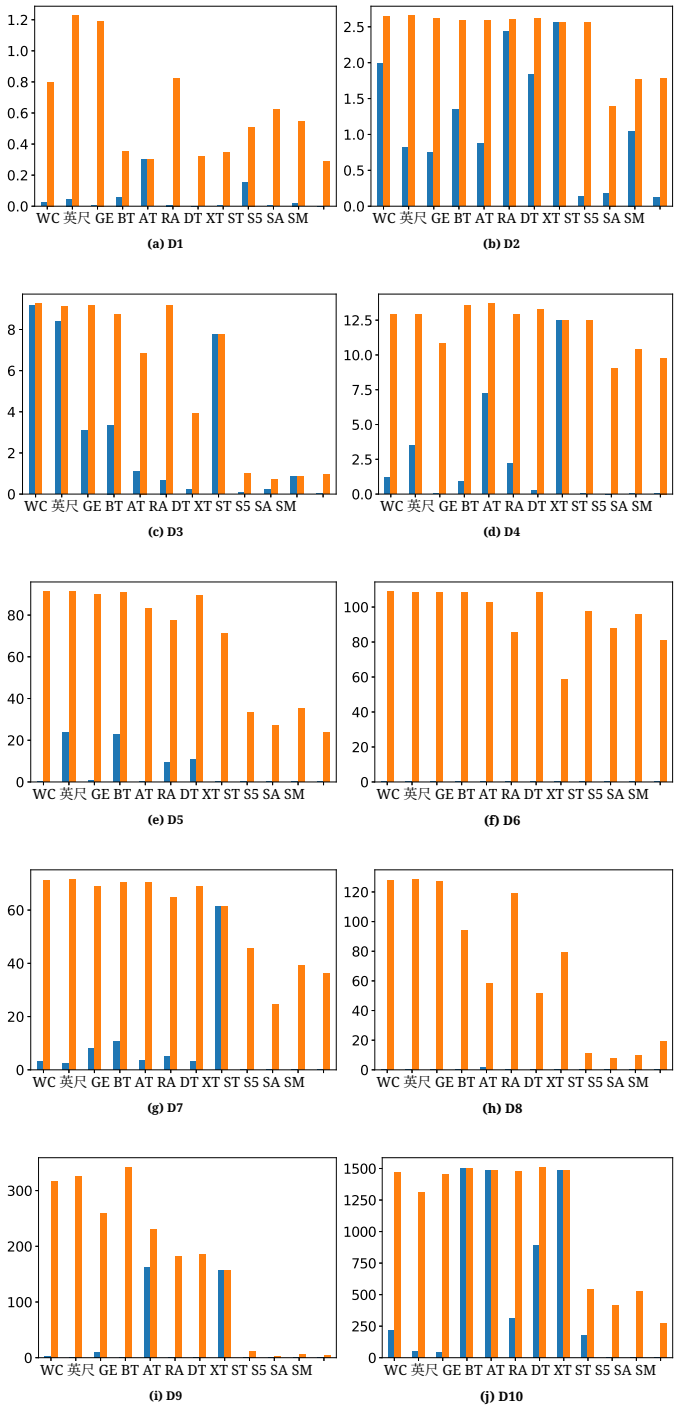


图14：各方法的无监督匹配时间（秒）。蓝色表示达到最高 F1 值之前的时间；橙色表示总计时间。

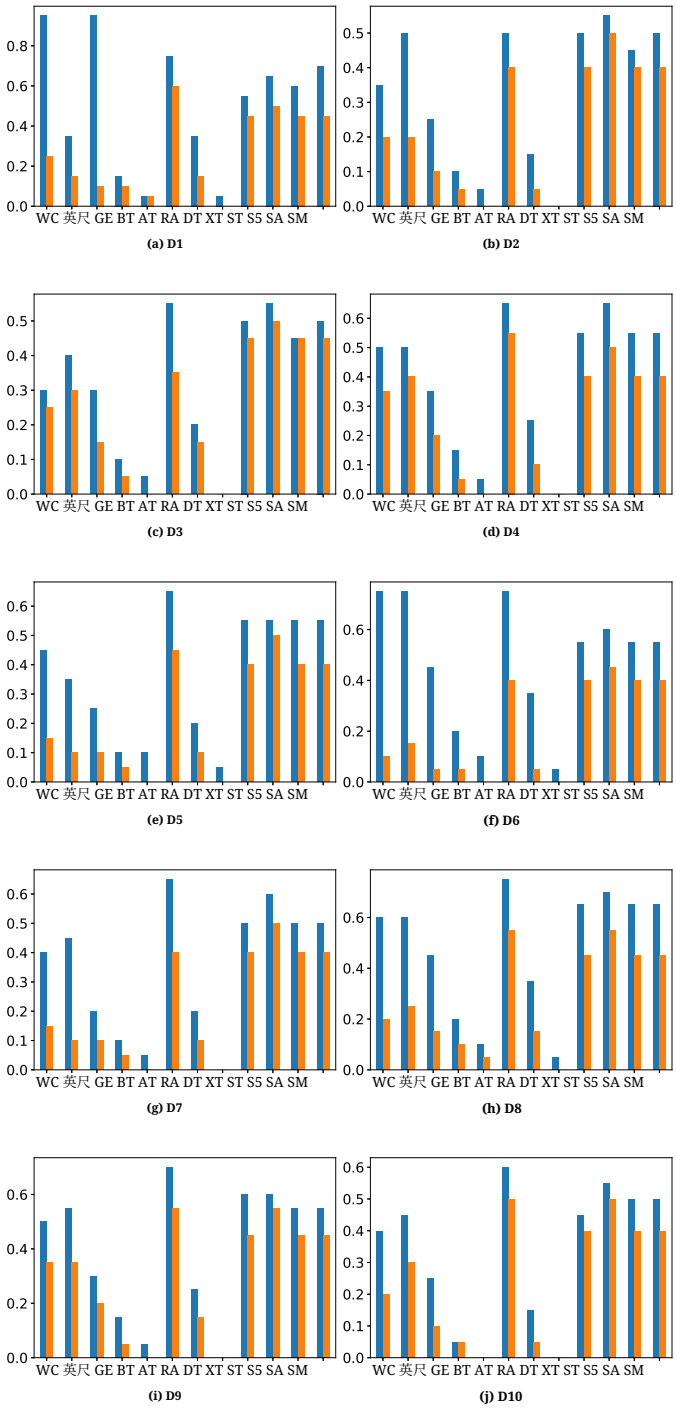


图15：每种方法的无监督匹配 δ。蓝色表示找到最佳 F1 值的 δ，橙色表示算法终止的 δ。

6.4 有监督匹配

表6显示，对于愿意牺牲少许

有效性较低：其训练时间和预测时间始终低于表现最佳的模型 RoBERTa，平均分别低 9% 和 7%，

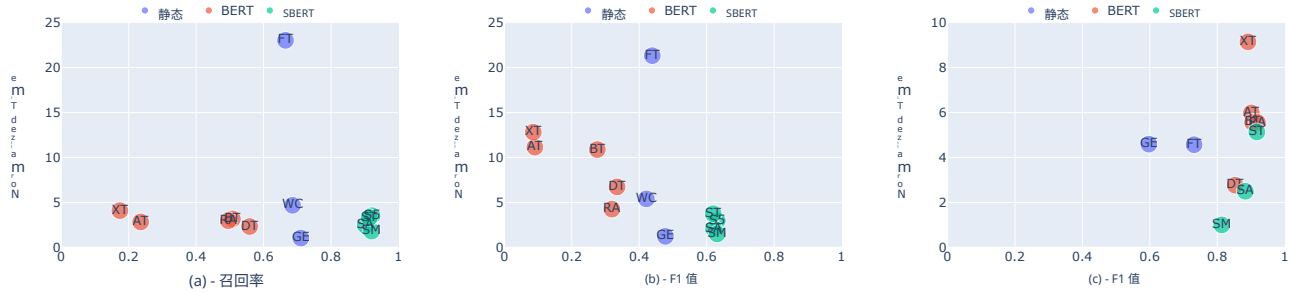


图16：E **在表2(a)的所有数据集上，(a) 阻塞的有效性与时间效率的平均权衡 wi** **th**
k=10和 (b) 无监督 M **匹配（相对于可达到的最佳 F1 值）以及表 3 中有监督匹配的所有数据集。**

平均而言。其余 SentenceBERT模型 S-DistilRoBERTa 和 S-MiniLM 带来了更显著的效率提升：在所有情况下，它们都将 RoBERTa 的训练时间和测试时间缩短了 50% 以上，而其 F1 值平均仅低 5%。类似地，DistilBERT 将 RoBERTa 的运行时间缩短至一半，但 F1 值降低了 7%。XLNet 在各个方面都逊于 RoBERTa。在所有数据集中，XLNet 始终是明显最慢的模型，因此在各个方面都逊于 RoBERTa。BERT 也是如此，但程度较轻，即在所有指标上相差 <2%。ALBERT 的训练时间略低于 RoBERTa（低 7%），但代价是预测时间更长（高 8%）；与此同时，其 F1 值平均仅低 2%。至于静态模型，尽管其 F1 值非常低，但平均而言，它们在训练时间和测试时间方面分别仅比 RoBERTa 快 10% 和 17%。总体而言，我们可以得出结论：*SentenceBERT*模型明显快于 *BERT-base* 模型，因此在有效性与时间效率之间实现了更好的权衡。

	DSM1		DSM2		DSM3		DSM4		DSM5	
	t_t	t_e	t_t	t_e	t_t	t_e	t_t	t_e	t_t	t_e
FT	851.9	4.8	100.9	0.6	1,155.0	6.9	2,527.3	14.8	882.1	4.7
GE	847.0	4.8	101.4	0.6	1,157.2	6.9	2,534.3	14.7	876.9	4.7
BT	1,811.2	11.2	66.3	0.4	1,525.9	9.6	2,615.9	15.8	1,093.8	6.8
AT	1,700.5	12.0	62.8	0.5	1,448.4	10.4	2,422.1	17.0	1,026.6	7.4
RA	1,810.8	11.2	66.2	0.4	1,548.7	9.6	2,666.9	15.8	1,111.3	6.9
DT	915.2	5.5	34.0	0.2	779.2	4.8	1,341.9	7.9	559.1	3.4
XT	2,920.7	24.2	92.2	0.7	2,120.5	15.9	3,196.5	21.0	1,423.2	10.1
ST	1,667.5	10.5	63.4	0.4	1,475.8	8.8	2,549.9	14.4	1,076.8	6.3
SA	828.2	5.0	31.4	0.2	716.9	4.3	1,253.7	7.1	518.1	3.1
SM	406.6	2.1	13.4	0.1	299.2	1.7	521.6	2.7	216.6	1.2

表6：表 3 中各数据集上有监督匹配所有模型的训练（t_t）和测试（t_e）时间（以秒为单位）。

7 讨论与结论

图16总结了我们在三个实体解析任务上的实验结果。每个图中的横轴对应相应任务的有效性度量，纵轴对应归一化运行时间，该时间通过将某个模型的总体运行时间除以运行速度最快的模型的运行时间计算得出（即将最快的模型设为 1）。由这些坐标轴形成的空间展示了有效性与时间效率之间的权衡，其中表现最佳的模型更接近（1,1），即右下角。请注意，对于每个模型，我们都计算了其在表2(a)所列全部数据集上的平均有效性和归一化时间。

我们观察到，根据三种模型类型的相对性能，可以将实体解析任务分为两组。第一组包括无监督任务，即阻塞和无监督匹配；在这些任务中，SentenceBERT模型的表现始终显著优于另外两种类型。原因在于，无需微调其顶层注意力层，它们就能够区分匹配实体和不匹配实体。它们之间的差异平均而言并不明显。不过，S-GTR-T5通常在有效性方面表现出色，但速度较慢；而S-MiniLM则更为均衡，以略低的有效性换取略高的时间效率。第二优秀的类型是静态模型，其中GloVe明显优于FastText和Word2Vec；后两者的初始化成本显著更高，且有效性略低。实际上，在所有数据集和两种实体解析任务中，GloVe都是速度最快的模型，因此非常适合那些强调时间效率、可以接受较低有效性的实体解析应用。最后，所有基于BERT的模型在各方面的表现始终逊于另外两种类型。它们有效性不佳，源于缺少微调，这导致它们对匹配实体对和不匹配实体对都给出较低的相似度分数。

在有监督匹配中可以观察到不同的模式，其中所有基于BERT的模型在有效性方面都表现出色。不出所料，Distil-BERT牺牲了有效性，以换取显著更高的运行时间。请注意，在这种情况下，我们只考虑每个模型的测试/预测时间，因为训练时间属于一次性成本。S-MPNet同样展现出非常高的有效性，而S-DistilRoBERTa和S-MiniLM则侧重于时间效率。前者优于DistilBERT，后者虽然是最快的动态模型，却也是有效性最低的动态模型。静态模型的准确性低于所有动态模型，同时在运行时间方面也没有任何优势。因此，在这项任务中应避免使用静态模型。相反，可以根据具体应用的需求选择任意一种动态模型。唯一的例外是XLNet：它的速度慢得多，但有效性并不高于大多数动态模型。

未来，我们将扩展对包含数值的实体解析数据集的分析。我们还计划改进我们的端到端、参数化

且无需学习的实体解析方法，该方法采用 SentenceBERT模型，其在图 8(d) 中的性能十分出色。我们将探索用数据驱动的阈值替代其默认阈值的方法。

参考文献

[1] Dzmitry Bahdanau、Kyunghyun Cho 和 Yoshua Bengio。2014。通过联合学习对齐与翻译实现神经机器翻译。arXiv 预印本 arXiv:1409.0473（2014）。[2] Piotr Bojanowski、Edouard Grave、Armand Joulin 和 Tomas Mikolov。2017。利用子词信息丰富词向量。《计算语言学协会会刊》5（2017），135–146。[3] Ursin Brunner 和 Kurt Stockinger。2020。采用 Transformer 架构的实体匹配：数据集成的进步。载于 EDBT。463–473。[4] Daniel Cer、Mona Diab、Eneko Agirre、Inigo Lopez-Gazpio 和 Lucia Specia。2017。SemEval-2017 任务 1：语义文本相似度——多语言和跨语言重点评测。arXiv 预印本 arXiv:1708.00055（2017）。[5] Runjin Chen、Yanyan Shen 和 Dongxiang Zhang。2020。GNEM：通用的一对多神经实体匹配框架。载于 WWW。1686–1694。[6] Peter Christen。 [n. d.]Febrl：一个带有图形用户界面的开源数据清洗、去重和记录链接系统。载于 SIGKDD，Ying Li、Bing Liu 和 Sunita Sarawagi（编）。1065–1068。[7] Peter Christen。2012。数据匹配——记录链接、实体解析和重复检测的概念与技术。Springer。[8] Peter Christen。2012。可扩展记录链接和去重的索引技术综述。《IEEE 知识与数据工程汇刊》24，9（2012），1537–1555。[9] Vassilis Christophides、Vasilis Efthymiou、Themis Palpanas、George Papadakis 和 Kostas Stefanidis。2021。大数据端到端实体解析概述。《ACM 计算调查》53，6（2021），127:1–127:42。[10] Vassilis Christophides、Vasilis Efthymiou 和 Kostas Stefanidis。2015。数据网络中的实体解析。Morgan & Claypool Publishers。[11] Jacob Devlin、Ming-Wei Chang、Kenton Lee 和 Kristina Toutanova。2018。BERT：用于语言理解的深度双向 Transformer 预训练。arXiv 预印本 arXiv:1810.04805（2018）。[12] Xin Luna Dong 和 Divesh Srivastava。2013。大数据集成。《Proc. VLDB Endow.》6，11（2013），1188–1189。[13] Muhammad Ebraheem、Saravanan Thirumuruganathan、Shafiq R. Joty、Mourad Ouzzani 和 Nan Tang。2018。用于实体解析的元组分布式表示。《Proc. VLDB Endow.》11，11（2018），1454–1467。[14] Cheng Fu、Xianpei Han、Jiaming He 和 Le Sun。2020。异构实体解析的层次匹配网络。载于 IJCAI。3665–3671。[15] Lise Getoor 和 Ashwin Machanavajjhala。2012。实体解析：理论、实践与开放性挑战。《Proc. VLDB Endow.》5，12（2012），2018–2019。[16] Geoffrey Hinton、Oriol Vinyals、Jeff Dean 等。2015。提炼神经网络中的知识。arXiv 预印本 arXiv:1503.02531 2，7（2015）。[17] Xiaoqi Jiao、Yichun Yin、Lifeng Shang、Xin Jiang、Xiao Chen、Linlin Li、Fang Wang 和 Qun Liu。2019。TinyBERT：为自然语言理解提炼 BERT。arXiv 预印本 arXiv:1909.10351（2019）。[18] Jungo Kasai、Kun Qian、Sairam Gurajada、Yunyao Li 和 Lucian Popa。2019。借助迁移学习和主动学习实现低资源深度实体解析。载于 ACL。5851–5861。[19] Batya Kenig 和 Avigdor Gal。2013。MFIBlocks：一种有效的实体解析阻塞算法。《信息系统》38，6（2013），908–926。[20] Hanna Köpcke、Andreas Thor 和 Erhard Rahm。2010。真实世界匹配问题中的实体解析方法评估。《Proc. VLDB Endow.》3，1（2010），484–493。[21] Simon Lacoste-Julien、Konstantina Palla、Alex Davies、Gjergji Kasneci、Thore Graepel 和 Zoubin Ghahramani。2013。SIGMa：用于对齐大型知识库的简单贪心匹配。载于 KDD。572–580。[22] Zhenzhong Lan、Mingda Chen、Sebastian Goodman、Kevin Gimpel、Piyush Sharma 和 Radu Soricut。2019。AlBERT：用于语言表示自监督学习的轻量级 BERT。arXiv 预印本 arXiv:1909.11942（2019）。[23] Bing Li、Wei Wang、Yifang Sun、Linhan Zhang、Muhammad Asif Ali 和 Yi Wang。2020。GraphER：基于图卷积神经网络的以词元为中心的实体解析。载于 IAAI。8172–8179。[24] Wen Li、Ying Zhang、Yifang Sun、Wei Wang、Mingjie Li、Wenjie Zhang 和 Xuemin Lin。2019。高维数据上的近似最近邻搜索——实验、分析与改进。《IEEE 知识与数据工程汇刊》32，8（2019），1475–1488。[25] Yuliang Li、Jinfeng Li、Yoshihiko Suhara、AnHai Doan 和 Wang-Chiew Tan。2020。基于预训练语言模型的深度实体匹配。《Proc. VLDB Endow.》14，1（2020），50–60。[26] Yuliang Li、Jinfeng Li、Yoshihiko Suhara、AnHai Doan 和 Wang-Chiew Tan。2020。基于预训练语言模型的深度实体匹配。CoRR abs/2004.00584（2020）。https://arxiv.org/abs/2004.00584

[27] Qi Liu、Matt J. Kusner 和 Phil Blunsom。2020。《上下文嵌入综述》。CoRR abs/2003.07278（2020）。https://arxiv.org/abs/2003.07278[28] Yinhan Liu、Myle Ott、Naman Goyal、Jingfei Du、Mandar Joshi、Danqi Chen、Omer Levy、Mike Lewis、Luke Zettlemoyer 和 Veselin Stoyanov。2019。《Roberta：一种经过稳健优化的BERT预训练方法》。arXiv预印本 arXiv:1907.11692（2019）。[29] Yury A. Malkov 和 Dmitry A. Yashunin。2020。《使用分层可导航小世界图进行高效且稳健的近似最近邻搜索》。IEEE模式分析与机器智能汇刊 42，4（2020），824–836。[30] Christopher D. Manning、Prabhakar Raghavan 和 Hinrich Schütze。2008。《信息检索导论》。剑桥大学出版社。[31] Tomas Mikolov、Kai Chen、Greg Corrado 和 Jeffrey Dean。2013。《向量空间中词表示的高效估计》。arXiv预印本 arXiv:1301.3781（2013）。[32] Tomas Mikolov、Ilya Sutskever、Kai Chen、Greg S Corrado 和 Jeff Dean。2013。《词与短语的分布式表示及其组合性》。神经信息处理系统进展 26（2013）。[33] Sidharth Mudgal、Han Li、Theodoros Rekatsinas、AnHai Doan、Youngchoon Park、Ganesh Krishnan、Rohit Deep、Esteban Arcaute 和 Vijay Raghavendra。2018。《实体匹配的深度学习：设计空间探索》。载于 SIGMOD。19–34。[34] Jianmo Ni、Chen Qu、Jing Lu、Zhuyun Dai、Gustavo Hernández Ábrego、Ji Ma、Vincent Y Zhao、Yi Luan、Keith B Hall、Ming-Wei Chang 等。2021。《大型双编码器是可泛化的检索器》。arXiv预印本 arXiv:2112.07899（2021）。[35] Hao Nie、Xianpei Han、Ben He、Le Sun、Bo Chen、Wei Zhang、Suhui Wu 和 Hao Kong。2019。《面向异构实体解析的深度序列到序列实体匹配》。载于信息与知识管理国际会议。629–638。[36] Daniel Obraczka、Jonathan Schuchart 和 Erhard Rahm。2021。《EAGER：嵌入辅助的知识图谱实体解析》。CoRR abs/2101.06126（2021）。[37] Matteo Paganelli、Francesco Del Buono、Andrea Baraldi 和 Francesco Guerra。2022。《分析BERT如何执行实体匹配》。Proc. VLDB Endow. 15，8（2022），1726–1738。[38] Matteo Paganelli、Francesco Del Buono、Pevarello Marco、Francesco Guerra 和 Maurizio Vincini。2021。《用于实体匹配任务的自动化机器学习》。载于 EDBT。[39] George Papadakis、George Alexiou、George Papastefanatos 和 Georgia Koutrika。2015。《同构数据上阻塞方法的模式无关与基于模式的配置》。Proc. VLDB Endow. 9，4（2015），312–323。[40] George Papadakis、Vasilis Efthymiou、Emmanouil Thanos 和 Oktie Hassan-zadeh。2022。《用于清洁-清洁实体解析的二分图匹配算法：实证评估》。载于 EDBT。2:462–2:474。[41] George Papadakis、Ekaterini Ioannou、Claudia Niederée 和 Peter Fankhauser。2011。《面向大型异构信息空间的高效实体解析》。载于 WSDM。535–544。[42] George Papadakis、Ekaterini Ioannou、Emanouil Thanos 和 Themis Palpanas。2021。《实体解析的四代》。Morgan & Claypool出版社。[43] George Papadakis、Dimitrios Skoutas、Emmanouil Thanos 和 Themis Palpanas。2021。《实体解析的阻塞与过滤技术：综述》。ACM Comput. Surv. 53，2（2021），31:1–31:42。[44] George Papadakis、Jonathan Svirsky、Avigdor Gal 和 Themis Palpanas。2016。《实体解析近似阻塞技术的比较分析》。Proc. VLDB Endow. 9，9（2016），684–695。[45] Ralph Peeters 和 Christian Bizer。2021。《针对实体匹配的BERT双目标微调》。Proc. VLDB Endow. 14，10（2021），1913–1921。[46] Jeffrey Pennington、Richard Socher 和 Christopher D Manning。2014。《GloVe：用于词表示的全局向量》。载于2014年自然语言处理经验方法会议（EMNLP）论文集。1532–1543。[47] Mohammad Taher Pilehvar 和 José Camacho-Collados。2020。《自然语言处理中的嵌入：意义的向量表示理论与进展》。Morgan & Claypool出版社。[48] Colin Raffel、Noam Shazeer、Adam Roberts、Katherine Lee、Sharan Narang、Michael Matena、Yanqi Zhou、Wei Li 和 Peter J. Liu。2020。《探索统一文本到文本转换器的迁移学习极限》。机器学习研究杂志 21，140（2020），1–67。[49] Nils Reimers 和 Iryna Gurevych。2019。《Sentence-BERT：使用孪生BERT网络的句子嵌入》。arXiv预印本 arXiv:1908.10084（2019）。[50] Adriana Romero、Nicolas Ballas、Samira Ebrahimi Kahou、Antoine Chassang、Carlo Gatta 和 Yoshua Bengio。2014。《FitNets：轻量深度网络的提示》。arXiv预印本 arXiv:1412.6550（2014）。[51] Victor Sanh、Lysandre Debut、Julien Chaumond 和 Thomas Wolf。2019。《DistilBERT，BERT的蒸馏版本：更小、更快、更便宜且更轻量》。arXiv预印本 arXiv:1910.01108（2019）。[52] Kaitao Song、Xu Tan、Tao Qin、Jianfeng Lu 和 Tie-Yan Liu。2020。《MPNet：用于语言理解的掩码与置换预训练》。神经信息处理系统进展 33（2020），16857–16867。

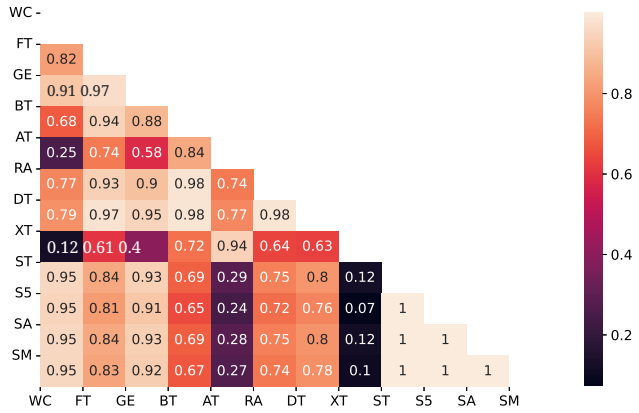


图17：模型与阻塞召回率（基于模式）的皮尔逊相关系数。



图17：方法相对于阻塞召回率的排名（越低越好）（基于模式）。

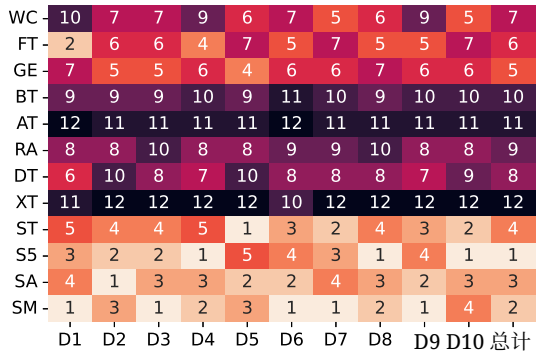


图19：在无监督设置下，每个方法在每个数据集上的匹配排名位置。数值越低越好（基于模式）。

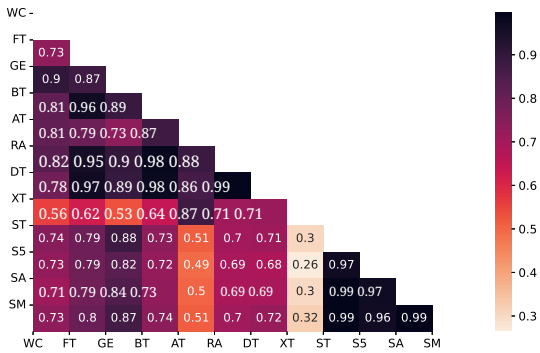
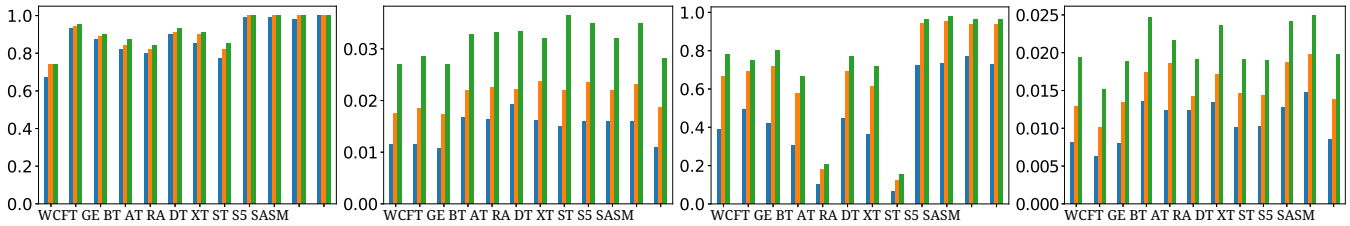


图20：无监督匹配中，语言模型每对样本的 F1 值与其皮尔逊相关系数（基于模式）。

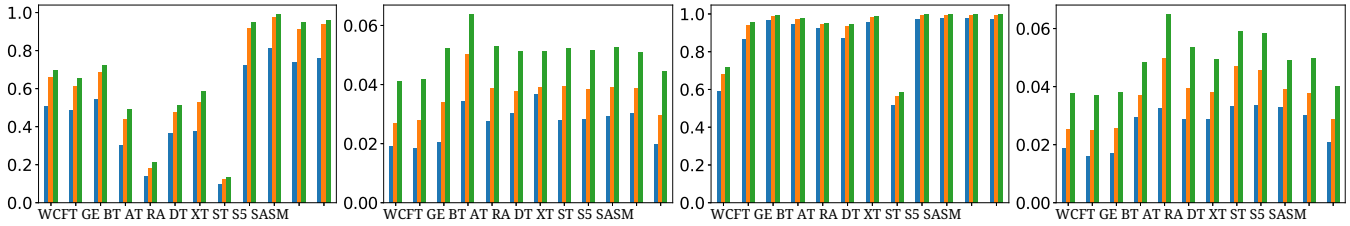
[53] Zhiqing Sun、Hongkun Yu、Xiaodan Song、Renjie Liu、Yiming Yang 和 Denny Zhou。2019。《MobileBERT：通过渐进式知识迁移实现BERT的任务无关压缩》。（2019）。[54] Ilya Sutskever、Oriol Vinyals 和 Quoc V Le。2014。《使用神经网络进行序列到序列学习》。神经信息处理系统进展 27（2014）。[55] Saravanan Thirumuruganathan、Han Li、Nan Tang、Mourad Ouzzani、Yash Govind、Derek Paulsen、Glenn Fung 和 AnHai Doan。2021。《实体匹配中的阻塞深度学习：设计空间探索》。*Proc. VLDB Endow.* 14，11（2021），2459–2472。[56] Immanuel Trummer。2022。《从BERT到GPT-3 Codex：释放大语言模型在数据管理中的潜力》。*Proc. VLDB Endow.* 15，12（2022），3770–3773。[57] Ashish Vaswani、Noam Shazeer、Niki Parmar、Jakob Uszkoreit、Llion Jones、Aidan N Gomez、Łukasz Kaiser 和 Illia Polosukhin。2017。《注意力机制就是你所需要的一切》。神经信息处理系统进展 30（2017）。[58] Alex Wang、Amanpreet Singh、Julian Michael、Felix Hill、Omer Levy 和 Samuel R Bowman。2018。《GLUE：面向自然语言理解的多任务基准与分析平台》。*arXiv*预印本 *arXiv:1804.07461*（2018）。[59] Wenhui Wang、Furu Wei、Li Dong、Hangbo Bao、Nan Yang 和 Ming Zhou。2020。《MiniLM：面向任务无关压缩的深度自注意力蒸馏预训练Transformer》。神经信息处理系统进展 33（2020），5776–5788。[60] Zhengyang Wang、Bunyamin Sisman、Hao Wei、Xin Luna Dong 和 Shuiwang Ji。2020。《CorDEL：一种用于实体链接的对比式深度学习方法》。载于 *ICDM*。1322–1327。[61] Renzhi Wu、Sanya Chaba、Saurabh Sawlani、Xu Chu 和 Saravanan Thirumuruganathan。2020。《ZeroER：使用零标注样例进行实体解析》。载于*ACM SIGMOD*数据管理国际会议*2020*论文集。1149–1164。[62] Zhilin Yang、Zihang Dai、Yiming Yang、Jaime Carbonell、Russ R Salakhutdinov 和 Quoc V Le。2019。《XLNet：面向语言理解的广义自回归预训练》。神经信息处理系统进展 32（2019）。[63] Zijun Yao、Chengjiang Li、Tiansi Dong、Xin Lv、Jifan Yu、Lei Hou、Juanzi Li、Yichi Zhang 和 Zelin Dai。2021。《通过将特征学习与决策解耦实现可解释的低资源实体匹配》。载于 *ACL/IJCNLP*。2770–2781。[64] Dongxiang Zhang、Yuyang Nie、Sai Wu、Yanyan Shen 和 Kian-Lee Tan。2020。《面向实体匹配的多上下文注意力》。载于 *WWW*。2634–2640。[65] Wei Zhang、Hao Wei、Bunyamin Sisman、Xin Luna Dong、Christos Faloutsos 和 David Page。2020。《AutoBlock：面向实体匹配的自动化阻塞框架》。载于 *WSDM*。744–752。

8 附录I：基于模式的实验

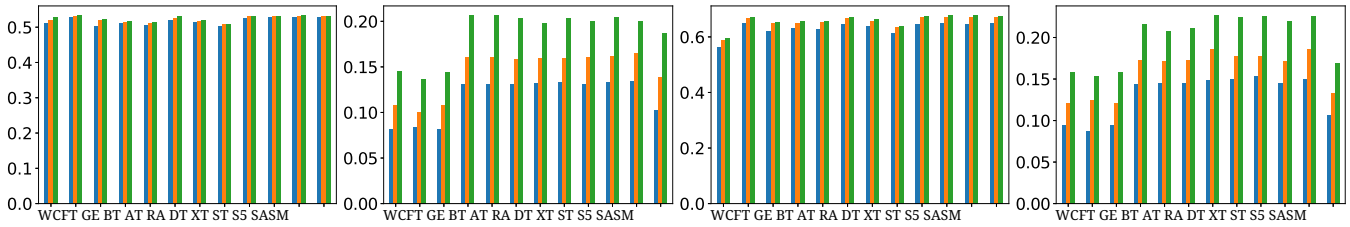
图 17、18 和 21 展示了基于模式的设置下阻塞的结果。其趋势与模式无关的情况相似：静态模型速度更快，但效率不高，而 SBERT 模型占据主导地位。图 22、20 和 19 中的无监督匹配也呈现出类似趋势。



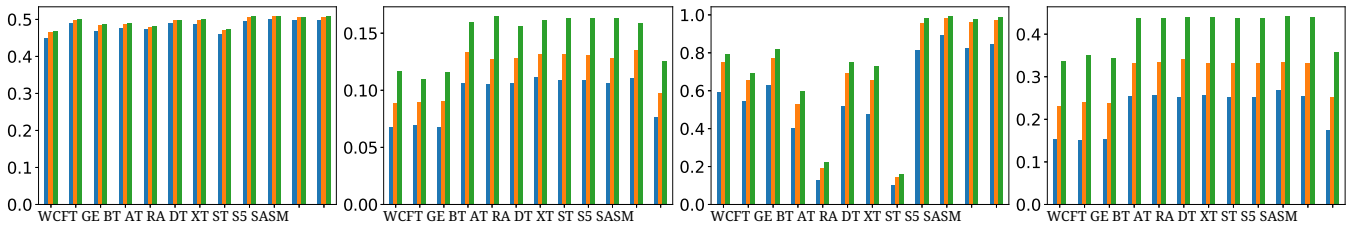
(a) D1 / 召回率 (b) D1 / 时间 (c) D2 / 召回率 (d) D2 / 时间



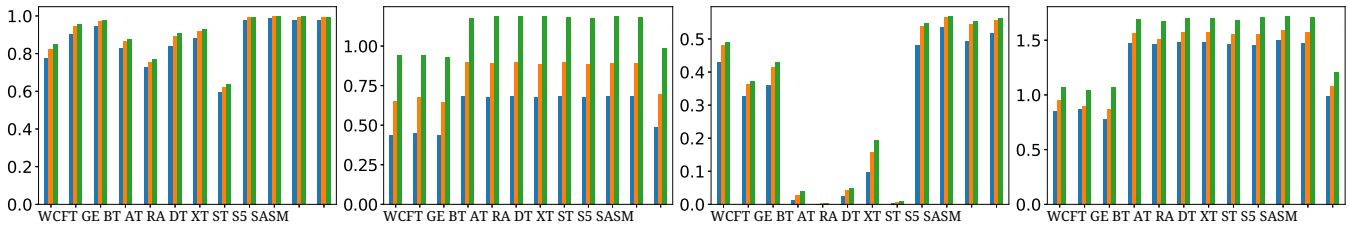
(e) D3 / 召回率 (f) D3 / 时间 (g) D4 / 召回率 (h) D4 / 时间



(i) D5 / 召回率 (j) D5 / 时间 (k) D6 / 召回率 (l) D6 / 时间



(m) D7 / 召回率 (n) D7 / 时间 (o) D8 / 召回率 (p) D8 / 时间



(q) D9 / 召回率 (r) D9 / 时间 (s) D10 / 召回率 (t) D10 / 时间

图 21：真实数据中各案例采用不同方法时的召回率和阻塞时间（秒）。对于 $k \in \{1、5、10\}$（基于模式）。



（a）D1 / 分数　（b）D1 / 时间　（c）D2 / 分数　（d）D2 / 时间

(e) D3 / 分数　(f) D3 / 时间　(g) D4 / 分数　(h) D4 / 时间

(i) D5 / 分数　(j) D5 / 时间　(k) D6 / 分数　(l) D6 / 时间

(m) D7 / 分数　(n) D7 / 时间　(o) D8 / 分数　(p) D8 / 时间

(q) D9 / 分数　(r) D9 / 时间　(s) D10 / 分数　(t) D10 / 时间

图 22：精确率、召回率、 F1 值 和 匹配时间：表 2(a)（基于模式）中各数据集的无监督匹配结果。