

字符串2s tring：面向字符串到字符串算法的现代 Python 库

Mirac Suzgun斯坦福大学[1]

Stuart M.Shieber哈佛大学

Dan Jurafsky斯坦福大学

谨以此纪念 *Lynn*。

arXiv:2304.14395v1 [cs.CL] 27 Apr 2023

摘要

我们介绍**string2string**，这是一个开源库，提供了一套全面且高效的算法，用于解决各类字符串到字符串问题。它既包含传统算法方案，也包含近期先进的神经网络方法，可处理字符串对齐、距离度量、词法搜索、语义搜索和相似性分析等问题；此外还提供多种实用的可视化工具和评估指标，帮助解读和分析这些方法。该库包含多种知名算法，例如用于成对局部对齐的 Smith-Waterman算法、用于全局对齐的 Hirschberg 算法、用于计算编辑距离的 Wagner-Fisher算法、用于相似性分析的 BARTScore 和 BERTScore、用于词法搜索的 Knuth-Morris-Pratt算法，以及用于语义搜索的 Faiss。此外，在适当且合适的情况下，它还封装了某些框架和评估指标中现有的高效且广泛使用的实现，例如 sacreBLEU 和 ROUGE。总体而言，该库旨在与现有字符串库相比，提供更广泛的覆盖范围和更高的灵活性。它可用于自然语言处理、生物信息学和计算社会科学中的多种下游应用、任务和问题。该库使用 Python 实现，可通过 pip 轻松安装，并提供简洁的 API。源代码、文档和教程均可在我们的 GitHub 页面上获取：https://github.com/stanfordnlp/string2string。[1]

1 引言

字符串到字符串问题在众多领域和学科中具有广泛的应用，包括但不限于自然语言处理（例如信息抽取、拼写检查和语义搜索）、计算分子生物学（例如DNA序列比对）、编程语言与编译器（例如语法解析和编译）、以及计算社会科学和数字人文（例如对文学文本和语料库进行词汇与语义分析）。

当前字符串到字符串处理、对齐、距离、相似度和搜索算法的现状，表现为多种广泛使用的编程语言（如 C++、Java 和 Python）中存在大量实现。然而，其中许多实现彼此并未整合，也缺乏灵活性、模块化和全面的文档，从而阻碍了用户使用这些实现。因此，迫切需要一个统一平台，将这些功能整合到一个易于访问且全面的系统中。

在本文中，我们介绍了一个开源库，它提供了广泛的算法和技术集合，用于字符串到字符串映射的对齐、操作或评估。[2] 这些问题包括衡量两个字符串之间的词汇距离（例如，采用 Levenshtein 编辑距离指标），计算两个 DNA 序列之间的局部或全局对齐（例如，基于 BLOSUM 等替换矩阵），计算两段文本之间的语义相似度（例如，使用 BART 嵌入），以及执行高效的语义搜索（例如，通过 FAIR 的 Faiss 库（Johnson等人，2019））。

string2string库经过精心设计，旨在优先遵循模块化、完整性、效率、灵活性和清晰性等关键设计原则。作为一项开源计划，该库未来将持续发展和调整，以满足用户社区不断变化的需求；我们也致力于确保该库始终是一项灵活、易于访问且充满活力的资源，能够适应字符串到字符串问题和任务不断变化的格局。

[1]通信地址：msuzgun@cs.stanford.edu。

[2]我们将字符串 定义为按顺序排列的字符集合，例如字母、数字和符号，用于表示一个信息、文本或数据单元。字符串可以用来表示各种内容，从简单的句子到复杂的核酸序列或精密的计算机程序。

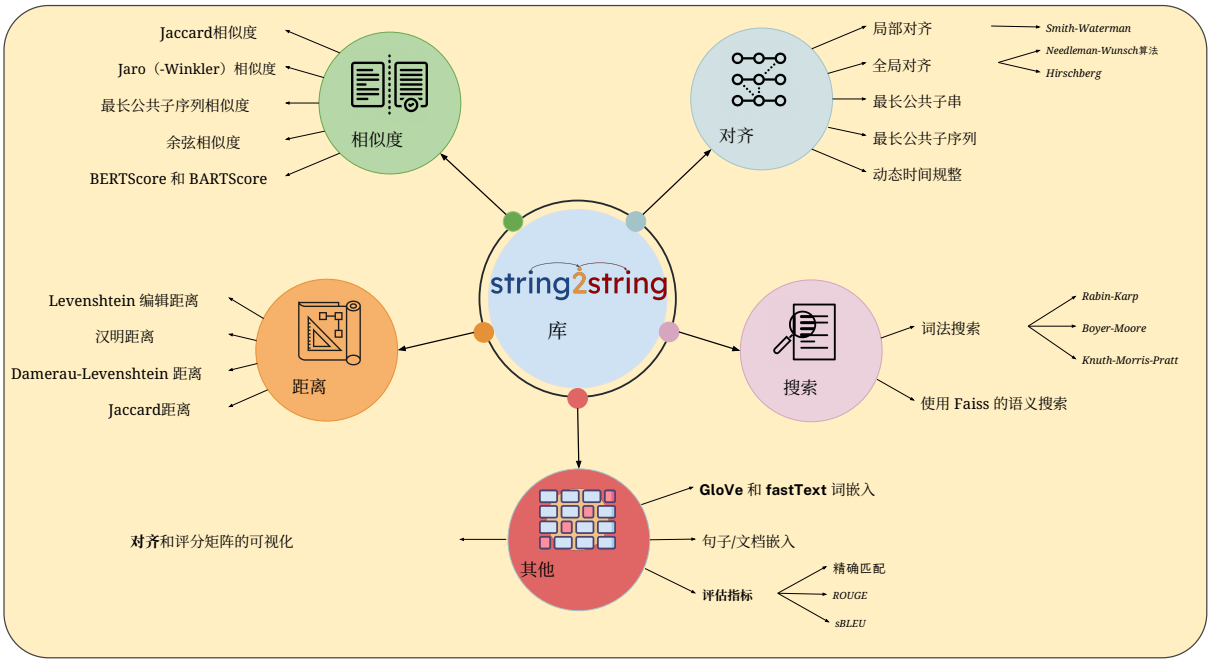


图 1： `string2string` 库提供了广泛的算法和技术，用于处理涉及字符串到字符串映射的成对对齐、比较、评估、操作和处理的各种问题与任务。其中包括广泛使用的算法，例如用于局部对齐的 Smith-Waterman（Smith 和 Waterman，1981年），用于完全字符串匹配（搜索）的 Knuth-Morris-Pratt（Knuth等人，1977），以及用于编辑距离的 Wagner-Fisher（Wagner和Fischer，1974年）；还包括较新的神经网络方法，例如用于语义相似度测量的 BERTScore（Zhang等人，2020年）和 BARTScore（Yuan等人，2021年），以及用于语义搜索的 Faiss（Johnson等人，2019）。该库不仅支持单个字符串，也支持字符串列表，因此用户可以在标记、单词或句子层面对字符串进行对齐和比较。此外，它还包含可视化功能，允许用户将字符串的对齐结果和评分矩阵进行可视化。

2 相关工作

自然语言处理和机器学习领域长期以来一直拥有鼓励开放源代码工具和库、并重视相关文化的优良传统。在设计我们自己的字符串到字符串库时，我们涵盖了针对各种问题的传统算法方法和神经网络方法，并从 Natural Language Toolkit（NLTK；Bird 和 Loper，2004年）、Gensim（ˇReh ̊uˇrek 和 Sojka，2010年）、OpenGrm Ngram（Roark 等，2012年）、斯坦福CoreNLP Toolkit（Manning 等，2014年）、OpenNMT（Klein 等，2017年）、Tensor2tensor（Vaswani 等，2018年）、AllenNLP（Gardner 等，2018年）、fairseq（Ott 等，2019年）、spaCy（Neumann 等，2019年）、Stanza（Qi 等，2020年）、Transformers（Wolf 等，2020年）和 Torch-Struct（Rush，2020年）等众多项目中汲取了灵感。

3 算法概述

string2string库提供了丰富的算法解决方案，用于处理各种字符串到字符串问题和任务。我们将这些算法分为四类：成对对齐、距离度量、相似性分析和搜索。[3] 每个类别都包含一套高效算法，专门用于解决各自领域中的特定问题。下面我们将简要介绍这些算法，以及它们旨在解决的相关问题或任务。

3.1 成对比对

成对字符串比对是指在两个字符串之间找出最优对齐方式的问题，例如 DNA 中的核苷酸序列或文本中的段落。这项任务需要对齐这些字符串，使匹配符号的数量最大化，同时在必要时允许出现空位或不匹配。成对字符串比对是一种应用广泛的技术，在 DNA序列比对、数据库搜索和系统发育分析等任务中发挥着重要作用。

如表1所示，该库目前能够为局部对齐、全局对齐、最长公共子串

[3]根据对偶性，距离度量方法可以自然地用于字符串相似性分析，反之亦然。

成对比对
局部对齐：基于替换 矩阵和空位罚分函数，在输入序列的指定区域内允许出现空位和不匹配，从而求得两个字符串之间最佳的匹配子串或子序列对齐。 ⋆（a）动态规划解法（Smith 和 Waterman，1981年）：$\mathcal{O}(nm)$按空间和时间复杂度计算。
全局对齐：两个字符串在整个长度范围内的最佳对齐。 ⋆（a）动态规划解法（Needleman 和 Wunsch，1970）：$\mathcal{O}(nm)$按空间和时间复杂度计算。 ⋆（b）分治 + 动态规划解法（Hirschberg，1975）：$\mathcal{O}(m)$在空间和 $\mathcal{O}(nm)$时间方面。
最长公共子串：同时出现在两个字符串中的最长连续子串。 ⋆（a）动态规划解法：$\mathcal{O}(nm)$在空间和时间方面。
最长公共子序列：以相同顺序出现在两个字符串中的最长可能符号序列。 ⋆（a）动态规划解法：$\mathcal{O}(nm)$在空间和时间方面。
动态时间规整（DTW）：使不同长度序列之间的距离最小化的最优规整路径。 ⋆（a）动态规划解法（Sakoe 和 Chiba，1978年）：$\mathcal{O}(nm)$空间和时间方面。 ⋆（b）通过 Hirschberg （1975）算法对（a）进行时空改进：将空间复杂度降低至 $\mathcal{O}(m)$。
距离
Levenshtein 编辑距离：插入所需的最少次数 , 删除 , 和替换，以将 S 转换为 T。 ⋆（a）动态规划解法（Wagner和Fischer，1974年）：$\mathcal{O}(nm)$在空间和时间方面。 ⋆（b）（a）的空间优化版本：通过仅存储两行，将空间复杂度降低至 $\mathcal{O}(m)$。
汉明距离：等长字符串 S 和 T中不同的索引总数。 ⋆（a）朴素解法：在空间和时间方面为 $\mathcal{O}(n)$。
Damerau–Levenshtein距离：将字符串转换为所需的插入、删除、替换和相邻转置的最少次数以将 S 转换为 T。 ⋆ (a) 动态规划解法（Wagner-Fisher算法的简单扩展）：空间复杂度和时间复杂度均为 $\mathcal{O}(nm)$。 ⋆ (b) (a) 的空间优化版本：通过仅存储两行，将空间复杂度降至 $\mathcal{O}(m)$。
Jaccard距离：Jaccard相似度的逆（即 1.0 - Jaccard相似系数）。 ⋆ (a) 朴素解法：空间复杂度和时间复杂度均为 $\mathcal{O}(n)$。

表 1： 该库所处理的成对字符串比对和距离问题概览，以及用于解决这些问题的算法方法。在所有情况下，我们假设给定两个字符串 S 和T，它们取自有限字母表 Σ，其中 $n = |S|$、 $m = |T|$ 和 $k = |\Sigma|$，且满足 $m \leq n$。此外，在可能的情况下，我们还会提供这些问题的暴力法和记忆化解法（例如编辑距离问题）。

（LCSubstring）、最长公共子序列（LC-Subsequence）和动态时间规整（DTW）问题提供高效解决方案。值得注意的是，该套件涵盖的所有问题和任务均可使用基于标准动态规划的解决方案来处理。对于长序列或字符串比对问题，也存在其他方法，例如 FASTA（Lipman 和 Pearson，1985）和 BLAST（Altschul等人，1990）；这些方法通过采用概率方法或启发式方法提高效率，但并不总能保证获得最优解，并且可能以牺牲准确性为代价换取速度。由于能够快速处理大型数据集并提供较为准确的结果，BLAST 和 FASTA 目前仍广泛应用于生物信息学领域，因此我们计划未来将它们纳入 `string2string`库。

3.2 距离

字符串距离是指根据某个距离函数，量化给定的两个字符串之间差异程度的问题。例如，Levenshtein 编辑距离度量对应于将一个字符串转换为另一个字符串所需的最少插入、删除或替换操作次数。该问题有一个著名的动态规划解法，通常称为 Wagner-Fisher 算法（Wagner和Fischer，1974年）。在这个库中，我们实现了 Wagner-Fisher 算法——其时间和空间复杂度均为二次方——并提供了它的改进版本，将总体空间复杂度降低至线性。[4] 我们还介绍并提供了高效的解决方案，用于计算汉明距离

[4]顺带一提，我们强调Backurs和Indyk（2015年的一项重要发现：除非强指数时间假设不成立，否则无法在强次二次时间内计算两个字符串之间的编辑距离。

相似度
Jaccard相似度：两个字符串中都出现的唯一符号集合（即交集）的大小，除以两个字符串中符号并集的大小。⋆ (a) 朴素解法：空间和时间复杂度均为 $\mathcal{O}(n)$。Jaro（-Winkler）相似度：一种基于两个字符串中的匹配符号和换位来衡量相似度的方法。⋆ (a) 朴素解法：时间复杂度为 $\mathcal{O}(nm)$， $\mathcal{O}$空间复杂度为 $n(n)$。最长公共子序列相似度：根据两个字符串的最长公共子序列的长度衡量它们的相似程度。⋆(a) 基于最长公共子序列问题的高效解法。♣余弦相似度：根据两个字符串中对应向量表示之间的夹角衡量它们的相似度。⋆ (a) 使用 numpy 和 torch 函数：时间复杂度为 $\mathcal{O}(E)$， $\mathcal{O}$空间复杂度为 $E(E)$。♢BERTScore（Zhang等人，2020年）：一种语义相似度度量，利用从预训练BERT模型（Devlin等人，2019）的上下文嵌入，估计两段文本之间的语义接近度。⋆ (a) 对原始 BERTScore 实现的改编：时间复杂度为 $\mathcal{O}(nm)$，空间复杂度为 $\mathcal{O}(nm \cdot E)$。BARTScore（Yuan等人，2021年）：一种利用预训练BART模型（Lewis等人，2019）测量语义相似度的方法，与人工判断具有较高的相关性。⋆ (a) 对原始 BARTScore 实现的改编：时间复杂度为 $\mathcal{O}(nm)$，空间复杂度为 $\mathcal{O}(nm \cdot E)$。
搜索
词法搜索： ⋆ (a) 朴素（暴力法）搜索：匹配时间方面为 $\mathcal{O}(mn)$，空间方面为 $\mathcal{O}(1)$。⋆ (b) Rabin-Karp算法（Karp和Rabin，1987）：匹配时间方面为 $\mathcal{O}(mn)$，空间方面为 $\mathcal{O}(1)$。⋆ (c) Boyer-Moore算法（Boyer和Moore，1977）：匹配时间方面为 $\mathcal{O}(mn)$，空间方面为 $\mathcal{O}(\lvert\Sigma\rvert)$。。⋆ (d) Knuth-Morris-Pratt算法（Knuth等人，1977）：匹配时间方面为 $\mathcal{O}(n)$，空间方面为 $\mathcal{O}(m)$。语义搜索：⋆ (a) FAISS（Johnson等人，2019）：匹配时间方面为 $\mathcal{O}(\log^2 n)$，空间方面为 $\mathcal{O}(n \cdot E)$。♠

表2：库中使用的字符串相似度和搜索方案概览。与表1一样，我们假设给定两个字符串， S和 T，它们来自有限字母表 Σ，其中 $n = |S|$、 $m = |T|$和 $k = |\Sigma|$，且 $m \leq n$。此外，在适用的情况下，我们使用 E 表示嵌入空间（或标记）的大小。Rabin-Karp和Knuth-Morris-Pratt算法的预处理都需要 $\Theta(m)$ 时间，搜索需要 $\Theta(n)$ 时间，而Boyer-Moore算法的预处理时间复杂度为 $\Theta(m + k)$ 。在空间方面，Rabin-Karp、Boyer-Moore和Knuth-Morris-Pratt算法分别需要 $\Theta(1)$、 $\Theta(k)$和 $\Theta(m)$。脚注 ♣：关于最长公共子序列相似度的数学表述，请参阅（Suzgun等，2022a）中的公式（11）。请注意，作者将这一相似度度量称为“Sim-LCS”。脚注 ♢：我们假设两个向量的维度（大小）均为 E。脚注 ♠：我们建议读者阅读Feinberg（2019）撰写的博客文章，以详细了解Facebook人工智能研究院的Faiss算法。

tance、Damerau-Levenshtein 距离和Jaccard距离问题，如表 1所示。

该库的一个显著特性是，用户可以根据所选的距离函数指定字符串操作（即插入、删除、替换和换位）的权重。此外，它不仅可以计算字符串对之间的距离，还可以计算字符串列表对之间的距离，因此不会将用户限制在仅进行字符或符号级别的比较。

3.3 相似度

字符串相似度是指这样一个问题：根据相似度函数来衡量给定的两个字符串彼此之间的相似程度。相似度函数可以基于多种标准定义，例如字符匹配、最长公共子串或子序列比较，或结构对齐。字符串相似度度量与字符串距离度量之间存在一种自然的对偶关系，这意味着可以轻松地将二者相互转换；因此，人们常常会交替使用字符串相似度和距离度量。

Jaccard相似度、Jaro相似度、Jaro-Winkler相似度、最长公共子序列相似度、余弦相似度、BERTScore 和 BARTScore 都是该库涵盖的相似度度量。其中前四种可以视为词汇相似度度量，因为它们评估的是表面或结构上的接近程度；而后面三种则可以视为语义相似度度量，因为它们将给定字符串中各组成部分所隐含的意义纳入考量。

当前库使用户能够计算余弦相似度，不仅可以比较单个单词——通过预训练 GloVe（Pennington 等人，2014）或 fastText（Joulin 等，2016）词嵌入——还可以比较更长的文本片段，例如句子、段落，甚至

文档——通过神经语言模型（如 BERT（Devlin等人，2019）生成的平均嵌入或末标记嵌入。正如我们在第 4节中提到的，这项功能使用户只需编写几行代码，就能比较较长文本片段的语义相似度，从而在文本分析任务中获得更大的灵活性。

3.4 搜索

字符串搜索，也称为字符串匹配，是指确定给定模式字符串是否存在于更长字符串中的问题。字符串搜索的暴力方法是检查较长字符串中的每个位置，以确定其是否与模式字符串匹配；然而，这种方法可能效率低下，尤其是在处理较长字符串时。因此，该库还包含用于完全字符串匹配的 Rabin-Karp（Karp和Rabin，1987）、Boyer-Moore（Boyer和Moore，1977）和 Knuth-Morris-Pratt（Knuth等人，1977）算法。

该库还通过 Facebook人工智能研究院的 Faiss 库（Johnson等人，2019）支持语义搜索；从本质上说，这使得对稠密向量进行高效的相似性搜索和聚类成为可能。与之前的完全字符串匹配设置不同，Faiss 首先要求用户提供一个字符串（文本）列表作为语料库，并使用神经语言模型为每个字符串创建固定向量表示。[5] 语料库初始化完成后，便可以对语料库执行“查询”。给定一个新的查询，可以自动获取该查询的嵌入，将其映射到语料库的嵌入空间中，并返回该查询在嵌入空间中的最近邻，从而找到与查询在语义上最接近的文本。

正如人们所想象的那样，字符串搜索算法是极具实用性的工具，在不同领域拥有广泛的应用。这些算法使用户能够在长文本或大型语料库中定位并检索特定模式。例如，string2string 库中介绍的字符串搜索算法（如表2所示）可用于模式识别、DNA匹配、抄袭检测和数据挖掘，以及众多其他下游应用和任务。

[5]用户可以灵活地决定如何为每段文本获取固定嵌入。具体而言，用户可以选择不同的嵌入方法，例如对词元嵌入取平均，或选取序列中最后一个词元的嵌入。

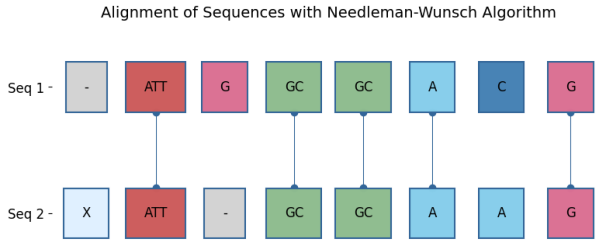


图 2：使用 Needleman-Wunsch算法得到的两个序列的对齐：`[ATT G GC GCA C G]` 和 `[X ATT GC GC A A G]`，（Needleman 和 Wun-s ch，1970年）。我们的库允许用户可视化字符串（或字符串列表）之间的成对比对。

4 其他功能

该库的一个显著特点是能够简化 GloVe（Pennington 等，2014）或 fastText（Joulin 等，2016）词嵌入的使用，让用户只需一行代码即可下载并使用这些词嵌入。这一简化流程不仅节省了用户的时间和精力，还免去了额外安装或复杂配置的需要。通过提供这一功能，我们旨在让该库更易于用户使用，并鼓励人们在各种字符串到字符串任务和应用中使用预训练词嵌入，例如计算两个单词之间的余弦相似度。[6]

同样，用户只需几行代码，就能从托管在 Hugging Face Models（Wolf 等人，2020年）或本地路径上的预训练语言模型中，轻松获取一段文本的词元嵌入平均值或最后一个词元的嵌入；我们还为这些计算提供 CPU 和 GPU 支持。

最后，我们要指出，该库提供了多种可视化功能，使用户能够直观地检查两个字符串之间的对齐情况，以及它们之间距离或相似度的评分矩阵。这一功能有助于理解和解读各种算法的输出结果，也能帮助用户为特定任务选择最合适的算法。将此功能纳入该库，旨在提升用户体验，并提供直观的方式来解读输出结果。[7] 图 2 展示了由我们的库生成的两个字符串列表之间的简单对齐。

[6]值得一提的是，fastText 为 157 种语言提供了预训练词嵌入，这些嵌入基于 Common Crawl 和维基百科训练而成，用户可以轻松下载并通过我们的库使用。[7]例如，我们的库提供了一份实用的实践教程，重点介绍 HUPD（Suzgun 等人，2022年b），这是一个大型专利语料库。该教程展示了如何高效地使用我们库的功能，进行语义搜索并可视化专利文档的文本内容。

5 库设计原则

我们致力于构建一个全面且易于使用的平台，涵盖众多用于字符串到字符串处理、比较、操作和搜索的算法。我们有意对该库进行结构设计与组织，使其易于定制化、功能扩展和模块化集成。

完整性。 该库提供了一套全面的经典算法以及神经方法，可应对各种字符串到字符串问题。我们有意纳入了高效且较为简单的解决方案，例如暴力法和基于记忆化的方法，以满足不同场景的需求。针对同一问题提供多种解决方案，使用户能够比较不同算法方法的性能。这种做法有助于用户深入理解不同算法解决方案之间的权衡，并认识到每种方法的优势与局限。

模块化。 为了提高代码库的效率和可维护性，我们采用了模块化设计，将代码拆分为更小的自包含模块。这种做法简化了开发人员添加新功能、修改现有功能的流程，同时也有助于高效地测试、调试和维护整个库。在开发过程中，模块化设计使我们能够快速定位并修复错误，而不会影响整个代码库。此外，这种模块化方法确保该库能够扩展并适应未来的更新与变更，从而帮助我们轻松改进库的功能，并在未来将其应用于各种任务和场景。

效率。 我们非常注重确保算法实现在计算和内存方面都高效，从而能够轻松处理大型数据集和复杂任务。我们通过 Python 内置的 `multiprocessing` 包以及 `joblib`，为基于进程的并行提供基本支持。此外，在适用情况下，我们也为基于神经网络的方法提供 GPU 支持。我们致力于在效率与清晰性之间取得平衡，同时也承认二者有时需要权衡。在这种情况下，我们更注重清晰性，确保算法透明且易于理解，即使这会牺牲一定的效率。尽管如此，我们仍认为，该库整体上的效率，加上它的透明性和可理解性，使其成为对社区有价值的资源。

支持字符串列表。 该库的设计不仅支持单个字符串，也支持字符串列表；在可行的情况下，用户可以在子词或词元级别对齐或比较字符串。这一特性让该库的应用场景更加灵活，使用户能够分析和比较更复杂的数据结构，而不局限于单个字符串。通过支持字符串列表，该库能够处理更广泛的文本输入类型和结构。

强类型。 强类型要求是该库的一个重要方面，可确保输入始终一致且准确，这对于生成可靠的结果至关重要。通过仔细标注库中所用算法的所有参数，我们力求提高代码库的稳健性和可靠性。这种方法有助于避免在执行过程中出现与输入相关的错误，例如数据类型或格式不正确。

可访问性。 该库使用 Python 实现。Python 是一种编程语言，也是学术界和工业界许多自然语言处理工具与应用的基础。string2string库可通过“pip”安装，并可集成到 PyTorch、TensorFlow 和 scikit-learn 等常用机器学习及自然语言处理框架中。

开源工作。 该库现在以及未来都将对所有用户免费且开放。我们希望这种方式能够促进社区驱动开发，并鼓励研究人员和开发者之间的协作，使他们能够为该库作出贡献并加以改进。

6 结论

我们推出了 string2string，这是一个开源库，提供大量算法，可用于解决各种字符串到字符串问题。该库使用 Python 实现，托管于 GitHub，并可通过 pip 安装。它包含详尽的文档和多个实践教程，帮助用户探索并有效使用该库。我们希望在开源社区的帮助下，不断扩展和改进这个库。我们鼓励用户随时向我们提供反馈、报告问题，并提出新功能建议，以拓展该库的功能和适用范围。

致谢

我们谨向 Corinna Coupette 致以最诚挚的谢意，感谢她为我们创建这个开源库提供灵感。此外，我们还要感谢 Sarah DeMott、Sebastian Gehrmann、Elena Glassman、Tayfun Gür、Daniel E. Ho、Sule Kahraman、Deniz Keles、Scott D. Kominers、Christopher Manning、Luke Melas-Kyriazi、Tolúlo. pé. Ògúnrè.mí、Alexander “Sasha” Rush、George Saussy、Kutay Serova、Holger Spamann、Kyle Swanson 和 Garrett Tanzer，感谢他们提出宝贵意见和实用建议，并给予支持。我们尤其要感谢 Federico Bianchi，他多次检查我们的代码、文档和教程，并为我们提供了极具帮助且富有建设性的反馈。

参考文献

SF Altschul、W Gish、W Miller、EW Myers 和 DJ Lipman。1990 年。基本局部比对搜索工具。《*Journal of Molecular Biology*》，215(3)：403–410。

Arturs Backurs 和 Piotr Indyk。2015 年。编辑距离无法在强次二次时间内计算（除非 SETH 为假）。载于 第四十七届 *ACM* 计算理论年度研讨会论文集，STOC ’15，第 51–58 页，美国纽约州纽约市。计算机协会。

Steven Bird 和 Edward Loper。2004 年。NLTK：自然语言工具包。载于 *ACL 互动海报与演示会议论文集*，第 214–217 页，西班牙巴塞罗那。计算语言学协会。

Robert S. Boyer 和 J. Strother Moore。1977 年。一种快速字符串搜索算法。*Commun. ACM*，20(10)：762–772。

Jacob Devlin、Ming-Wei Chang、Kenton Lee 和 Kristina Toutanova。2019年。BERT：用于语言理解的深度双向变换器预训练。收录于《*2019*年北美计算语言学协会分会年会论文集：人类语言技术》，第*1*卷（长篇与短篇论文），第4171–4186页，明尼阿波利斯，明尼苏达州。计算语言学协会。

Vlad Feinberg。2019年。Facebook 人工智能相似性搜索（FAISS），第二部分。访问日期：2023年3月13日。

Matt Gardner、Joel Grus、Mark Neumann、Oyvind Tafjord、Pradeep Dasigi、NelsonF. Liu、Matthew Peters、Michael Schmitz 和 Luke Zettlemoyer。2018 年。AllenNLP：深度语义自然语言处理平台。收录于自然语言处理开源软件研讨会论文集（*NLP-OSS*），第1–6页，澳大利亚墨尔本。计算语言学协会。

Daniel S. Hirschberg。1975年。一种用于计算最长公共子序列的线性空间算法。《*ACM*通信》，18(6):341–343。Jeff Johnson、Matthijs Douze 和 Hervé Jégou。2019年。使用图形处理器进行十亿规模相似性搜索。*IEEE*大数据汇刊，7(3):535–547。Armand Joulin、Edouard Grave、Piotr Bojanowski、Matthijs Douze、Hérve Jégou 和 Tomas Mikolov。2016年。FastText.zip：压缩文本分类模型。*arXiv*预印本 *arXiv:1612.03651*。Richard M Karp 和 Michael O Rabin。1987年。高效的随机模式匹配算法。《*IBM*研究与开发杂志》，31(2):249–260。Guillaume Klein、Yoon Kim、Yuntian Deng、Jean Senellart 和 Alexander Rush。2017年。OpenNMT：用于神经机器翻译的开源工具包。载于*ACL 2017*论文集：系统演示，第67–72页，加拿大温哥华。计算语言学协会。Donald E. Knuth、James H. Morris, Jr. 和 Vaughan R. Pratt。1977年。字符串中的快速模式匹配。《*SIAM*计算杂志》，6(2):323–350。Mike Lewis、Yinhan Liu、Naman Goyal、Marjan Ghazvininejad、Abdelrahman Mohamed、Omer Levy、Veselin Stoyanov 和 Luke Zettlemoyer。2020年。BART：用于自然语言生成、翻译和理解的序列到序列去噪预训练。载于计算语言学协会第*58*届年会论文集，第7871–7880页。David J Lipman 和 William R Pearson。1985年。快速而灵敏的蛋白质相似性搜索。《科学》，227(4693):1435–1441。Christopher Manning、Mihai Surdeanu、John Bauer、Jenny Finkel、Steven Bethard 和 David McClosky。2014 年。斯坦福 CoreNLP 自然语言处理工具包。载于计算语言学协会第*52*届年会论文集：系统演示，第55–60页，美国马里兰州巴尔的摩。计算语言学协会。Saul B Needleman 和 Christian D Wunsch。1970年。一种适用于搜索两种蛋白质氨基酸序列相似性的通用方法。《分子生物学杂志》，48(3):443–453。Mark Neumann、Daniel King、Iz Beltagy 和 Waleed Ammar。2019年。ScispaCy：用于生物医学自然语言处理的快速稳健模型。载于第*18*届*BioNLP*研讨会和共享任务论文集，第319–327页，意大利佛罗伦萨。计算语言学协会。Myle Ott、Sergey Edunov、Alexei Baevski、Angela Fan、Sam Gross、Nathan Ng、David Grangier 和 Michael Auli。2019年。fairseq：一种快速、可扩展的

序列建模工具包。载于*2019*年北美计算语言学协会分会会议（演示）论文集，第48–53页，美国明尼苏达州明尼阿波利斯。计算语言学协会。

Jeffrey Pennington、Richard Socher 和 Christopher Manning。2014年。GloVe：词语表示全局向量。载于《*2014*年自然语言处理经验方法会议（*EMNLP*）论文集》，第 1532–1543 页，多哈，卡塔尔。计算语言学协会。

Peng Qi、Yuhao Zhang、Yuhui Zhang、Jason Bolton 和 Christopher D. Manning。2020年。Stanza：适用于多种人类语言的 Python 自然语言处理工具包。载于《计算语言学协会第 *58* 届年会论文集：系统演示》，第 101–108 页，线上。计算语言学协会。

Radim ˇReh ̊uˇrek 和 Petr Sojka。2010年。使用大型语料库进行主题建模的软件框架。载于《*LREC 2010 NLP* 框架新挑战研讨会论文集》，第 45–50 页，瓦莱塔，马耳他。ELRA。
http://is.muni.cz/publication/884893/en。

Brian Roark、Richard Sproat、Cyril Allauzen、Michael Riley、Jeffrey Sorensen 和 Terry Tai。2012年。OpenGrm 开源有限状态语法软件库。载于 *ACL 2012* 年系统演示论文集，第 61–66 页，韩国济州岛。计算语言学协会。

Alexander Rush。2020年。Torch-Struct：深度结构化预测库。载于第 *58* 届计算语言学协会年会：系统演示论文集，第 335–342 页，线上。计算语言学协会。

Hiroaki Sakoe 和 Seibi Chiba。1978年。用于口语词识别的动态规划算法优化。《*IEEE* 声学、语音与信号处理汇刊》，26(1)：43–49。

T. F. Smith 和 Michael S. Waterman。1981年。常见分子子序列的识别。《分子生物学杂志》，147 1：195–7。

Mirac Suzgun、Luke Melas-Kyriazi 和 Dan Jurafsky。2022年a。遵循群体智慧：通过最小贝叶斯风险解码实现有效的文本生成。*arXiv*预印本 *arXiv:2211.07634*。

Mirac Suzgun、Luke Melas-Kyriazi、Suproteem K Sarkar、Scott Duke Kominers 和 Stuart MShieber。2022年b。哈佛大学—美国专利商标局专利数据集：一个大规模、结构良好且用途广泛的专利申请语料库。*arXiv* 预印本 *arXiv:2207.04043*。

Ashish Vaswani、Samy Bengio、Eugene Brevdo、Fran- cois Chollet、Aidan N. Gomez、Stephan Gouws、Llion Jones、Łukasz Kaiser、Nal Kalchbrenner、Niki Parmar、Ryan Sepassi、Noam Shazeer 和 Jakob Uszkoreit。2018年。用于神经机器翻译的 Tensor2tensor。*CoRR*，abs/1803.07416。

Robert A Wagner 和 Michael J Fischer。1974年。字符串到字符串校正问题。*ACM*学报（*JACM*），21(1)：168–173。

Thomas Wolf、Lysandre Debut、Victor Sanh、Julien Chaumond、Clement Delangue、Anthony Moi、Pier- ric Cistac、Tim Rault、Remi Louf、Morgan Funtow- icz、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 Rush。2020年。Transformers：最先进的自然语言处理。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。计算语言学协会。

Weizhe Yuan、Graham Neubig 和 Pengfei Liu。2021年。BARTScore：将生成文本作为文本生成进行评估。载于 神经信息处理系统进展，第 34 卷，第 27263–27277 页。Curran Associates, Inc.。

Tianyi Zhang、Varsha Kishore、Felix Wu、Kilian Q. Weinberger 和 Yoav Artzi。2020年。BERTScore：使用 BERT 评估文本生成。载于国际学习表征会议。