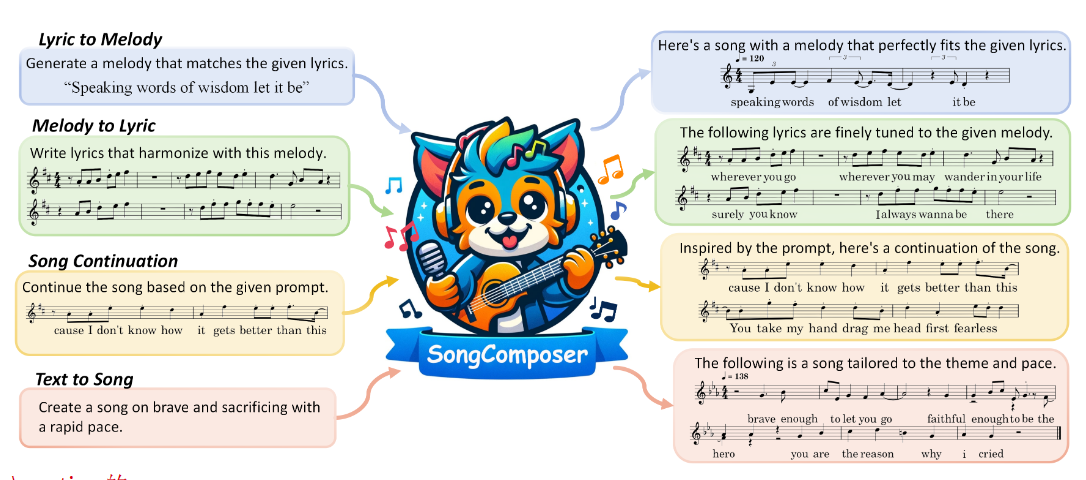
**SongComposer: A Large Language Model for Lyric and Melody Composition in Song Generation**

SongComposer：在歌曲生成中用于歌词和旋律创作的大语言模型

**太长不看版**

大语言模型的优越性能使得人们考虑使用其进行音乐创作，本文作者提出了一个名为SongComposer的大语言模型。研究团队音乐符号转化为独特的元组格式，并将其作为大语言模型的输入。 SongComposer在依照歌词生成旋律、依照旋律生成歌词、歌曲延续和依据文本创作歌曲的创作方面表现出卓越的性能。在研究过程中，研究团队也汇集并整理了一个全面的预训练数据集 SongCompose-PT，该数据集中旋律和歌词进行了单词级别的对齐。

**摘要**

本文讲解一个专为歌曲创作而设计的创新式大语言模型（LLM）SongComposer，它可以理解并生成旋律和歌词符号。现有的音乐相关大语言模型将音乐视为量化的音频信号，但这种隐式编码效率低下，灵活性差。 相比之下，研究人员采用了人类已有的，在音乐设计中成熟且高效的符号化表示方式，并使LLM能够像人类一样明确地创作歌曲。 在实践中，他们设计了一种新颖的元组设计，将旋律中的歌词和三个音符属性（音高、持续时间和休止时间）格式化，以此保证了LLM对音乐符号的正确理解，实现了歌词和旋律的精确对齐。为了向LLM传授基本的音乐理解，研究人员精心收集了SongCompose-PT，这是一个大规模的歌曲预训练数据集，包括中文或英文的歌词、旋律和成对的歌词-旋律。经过充分的预训练后，使用10K个精心制作的问答对（QA对）来赋予 LLM 指令遵循能力并可以解决各种任务。 通过广泛的实验，SongComposer 在依照歌词生成旋律、依照旋律生成歌词、歌曲延续和依据文本创作歌曲的创作方面表现出卓越的性能，优于 GPT-4 等先进LLM。 研究人员在项目页面上展示生成的示例。

**1.介绍**

最近大型语言模型（LLM）的繁荣极大地改变了人工智能领域，尤其是在自然语义理解和生成方面。这些模型为解析和生成人类语言建立了新的基准，在复杂语言环境中从翻译到推理等任务中展示了与人类等同的能力。一个自然而然的问题出现了：大语言模型能否在某些更情绪化、更抽象、更需要专业技能的领域表现得像人类一样？例如，LLM也可以像人类一样创作歌曲吗？

歌曲是一种独特的语言表现形式，它兼具旋律和歌词两方面，而歌曲创作工作是一个成熟领域，有系统化表述表述方法，即将歌曲进行符号化表示。这种方法根据文本将旋律转换为简洁的音符，将音乐进行了有效的抽象表示。它是人类创造的一种专门捕捉音乐思想的语言，这使其天然适合作为大语言模型的输入或输出。

然而，目前的探索忽略了这样一个成熟的解决方案，而是使用LLM来间接（implicitly）生成音乐，他们将音乐视为连续的音频信号，并对其进行量化处理，以生成稀疏（sparse）的tokens供大型语言模型学习。这导致了token数量过多和非语义化表示的问题，限制了从音乐到歌曲（如声音表达和歌词生成）的进一步探索。

在本文中，研究人员介绍了SongComposer，它可以像人类一样创作歌曲，通过利用歌曲符号来生成兼顾旋律和歌词的作品。据研究人员所称，这是第一次尝试使用LLM对歌词和旋律之间的关系进行建模，他们解决了以下挑战。

第一个挑战是“大语言模型如何学习乐谱”。乐谱是高度抽象的，与常见的字母用法不同。例如，旋律中的“F4”指的是“第四个八度的F音符”，而不是纸张大小或毫无意义的字母组合。为此，研究人员将旋律分解为代表音符属性的三元组——音高、持续时间和休止时间，并将它们作为大语言模型词汇表中的tokens。这种方法保留了音乐符号的实际含义，而不是重复使用词汇表中的原始tokens或使用单独音频编码器去适应抽象符号。对于配对的歌词-旋律数据，研究人员在单词（word）级别将歌词与旋律对齐，从而产生一系列元组。每个元组都包含一个歌词及其相应的音符属性。

第二个挑战是“大语言模型需要学习什么音乐”。LLM需要从大规模的歌曲数据中掌握基本的音乐理解，包括旋律、歌词以及它们之间的精确对齐。为了实现这一目标，研究人员汇集并整理了一个全面的预训练数据集 SongCompose-PT。该数据集包括 280K 首纯歌词歌曲、20K 组纯旋律歌曲和 15K 对中英文歌词和旋律。这个大规模的数据集是训练LLM的关键资源，研究人员计划适当地发布这些数据集，以推动该领域的进一步研究。

最后一个挑战是“LLM如何根据指令创作歌曲”。经过充分的预训练后，大语言模型可以理解并续写一首歌，但是其难以根据灵活多变的指令采取行动。为了缓解这种情况，研究人员设计了一个包含 10,000 个 问答（QA）对话数据集。这种策略使大语言模型能够解决歌曲生成领域中的多数任务。

传统模型在面对不同任务需要专门的解码器，与之相对的是，研究人员的SongComposer将所有与歌曲有关的任务都统一在了一个语言模型当中。此外，通过采用符号表示旋律和直接将歌词与旋律进行对齐的方法，研究人员成功地使大语言模型可以生成带有歌词的旋律，并使其所需的tokens大大减少。

总而言之，研究人员的贡献如下：

• 研究人员介绍了 SongComposer，这是一个专为歌曲创作而设计的 LLM。它可以通过歌曲的符号化表示来创作旋律和歌词，具有更好的token效率、精确的表达、灵活的格式和人类易读的输出。

• 研究人员收集了 SongCompose-PT，这是一个全面的预训练数据集，包括中文或英文的歌词、旋律以及配对的歌词和旋律。

• 大量实验表明，SongComposer 在歌词到旋律生成、旋律到歌词生成、歌曲延续和文本到歌曲创作等任务中优于 GPT-4 等高级大语言模型。

**2.相关工作**

大型语言模型：大型语言模型 （LLM） 的最新进展显著增强了自然语言处理能力。早期的编码器-解码器模型（如 BERT），以及后来的纯解码器模型（例如 GPT），通过生成式预训练在多样化的语言任务中展现出了令人瞩目的结果。进一步催生了一系列具有划时代意义的模型，包括PaLM、GPT-4和LLaMA，这些模型拥有数十亿个参数，能够在复杂的语言环境中展现出高级推理能力。随后，ChatGPT、InstructGPT和Vicuna等模型通过有监督的集成指令调优，进一步提高了生成上下文相关响应的能力，而且接近人类互动的质量。受大型语言模型强大推理能力的启发，研究人员开发了一个统一的端到端大型语言模型框架，创新地将应用扩展到歌词和旋律创作，以实现歌曲生成。

**音乐生成：**近年来，音乐生成取得了重大进展。其中有许多成果值得注意，例如Noise2music 的方法和Mou^sai利用的流行的扩散模型（Diffusion Model）根据文本描述生成音乐。另一条研究线的重点是将音乐信息转换为序列样式（sequence-style）的token，然后用Transformers处理这些序列。随着大型语言模型的进步，M2UGen扩展了已有功能，包括音乐理解和使用文本、图像甚至视频来指导创作过程。然而，这些现有作品主要产生没有歌词的纯旋律。与之相反，SongComposer不仅可以直接生成符号化的旋律，还可以在单词级别上对齐歌词以生成歌曲。

**配对歌词-旋律演唱数据集：**对于歌曲生成来说，具有成对的旋律和歌词标注的歌曲数据是极为重要的。具体来说，JVS-MuSiC、PopCS和OpenSinger虽然提供广泛的歌曲数据，但缺乏关键的歌词-旋律在时间上的对齐。NUS-48E、NHSS，Tohoku Kiritan和 Opencpop提供英语、日语和中文的歌唱语料库，并手动对齐歌词和旋律。然而，它们的规模有限，只有单声道歌手和风格。最近，M4Singer编译了大约 700 首带有歌词和旋律对的中文歌曲，但这个数量仍然不足以训练用于符号化音乐生成的大语言模型。在这项工作中，研究人员从头开始收集大约 4K 的中文歌曲，并制作了几个可公开访问的配对数据集，以构建大规模歌词-旋律配对数据集。

**3.SongCompose-PT数据集**

本节概述了研究人员的 SongCompose-PT 数据集的汇集、创建和结构，其中包括歌词、旋律和歌词-旋律对的单独集合，这些集合在单词级别将歌词与旋律同步。研究人员的目标是公开发布这个三重数据集，为后续研究提供基础资源。有关数据集的统计细分，请参阅研究人员的附录 A。

**纯歌词数据集：**研究人员从两个在线来源收集纯歌词数据集： （1） Kaggle数据集，包括 150K 首歌曲的被标注有Spotify Valence歌词，这是一个衡量歌曲积极性的指标。 （2） Music Lyric Chatbot数据集，包含140K 首华语歌曲的歌词。 经过一系列的歌词清理过程，研究人员从 283K 首歌曲中收集了高质量的歌词，其中包括 150K 的英文和 133K 的中文。

**纯旋律数据集：**为了将旋律数据集组织成基于文本的结构，研究人员自然地选择收集 MIDI 文件。 使用MIDI文件作为研究人员的纯旋律数据集简化了研究人员的编写步骤，MIDI文件固有的结构简单性使得研究人员能够高效地提取和操作旋律，无需复杂的音频处理。在研究人员的收集中，有45K来自LMD-matched MIDI数据集，而大约80K则是通过网络爬虫获取的。

为了解析MIDI文件，研究人员采用了*pretty\_midi*，它是一个可用于创建、操作和分析 MIDI 文件的 Python 模块。 研究人员只从这些MIDI文件中提取“旋律”或“人声”轨道。 由于MIDI中的旋律表示为随时间变化的音符序列，并且每个音符都有特定的音高，开始和结束时间戳，因此研究人员获得了由{音符音高，持续时间，休止时间}（{note pitch, note duration, rest duration}）组成的旋律三元组列表。

• **音符音高：**音符的音高由其对应的 MIDI 音符编号表示，范围从 0 到 127，数字 60被预定义为中音C。

• **音符持续时间：**音符的持续时间被定义为演奏音符的时间长度（以秒为单位）。这是根据嵌入在 MIDI 文件中的每个音符的开始和结束时间计算得出的，第k个音符持续时间=第k个音符结束时间-第k个音符开始时间。

• **休止持续时间：**休息持续时间表示演奏音符后的静音期。休止持续时间=第k+1个音符开始时间-第k个音符结束时间，研究人员执行必要的数据过滤以删除重复和质量差的样本，留下大约 20K 个 MIDI 样本。

**成对歌词-旋律数据集：**经过时间对齐处理的大规模成对歌词-旋律数据集对于训练用于歌曲创作的大语言模型至关重要，但由于歌词和旋律之间需要精确同步，因此难以编写，如图 1 所示，此任务需要详细的注释和特定的专业知识。研究人员利用了先前研究的数据集，包括来自 LMD-full数据集的 7,998 首歌曲和来自 Reddit 数据集的 4,199 首歌曲。此外，研究人员还整合了包含 100 首普通话流行歌曲的OpenCpop和收录了 700 首高品质的普通话歌曲的M4Singer。

为了进一步丰富配对的歌词-旋律数据，研究人员高效地处理了网络抓取的大规模信息，创建了一个4K经典国语歌曲数据集。如图 2 所示，用于收集歌词旋律数据的流程（pipeline）如下所示：

（1） 源数据抓取：研究人员抓取了网络上包含mp3文件及其对应的歌词文件的大型数据集，包括句子级（sentence-level）时间戳。

（2） 歌词清理：研究人员使用GPT-4清理歌词文本中不相关的细节，如歌名、艺人姓名、制作信息等。

（3） 片段切片：为了减轻长时间对齐的难度和错误积累，研究人员根据歌词文件中提供的时间戳将音频和歌词切成大约 10 秒（每个大约三句话）的成对片段。

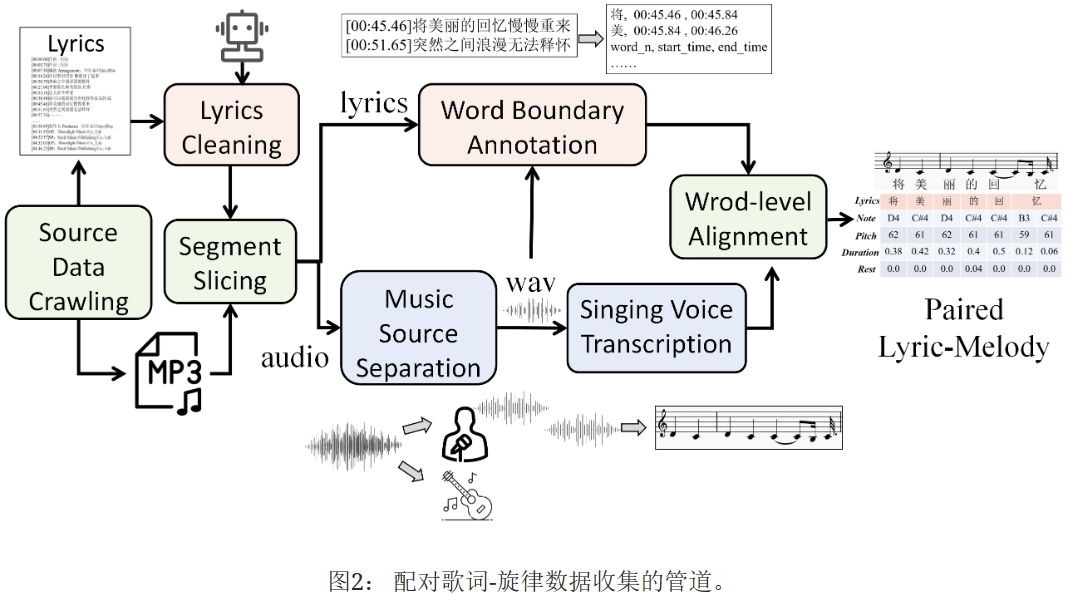
（4） 音乐资源分割：研究人员利用了UVR4，它是一个开源的音乐分割工具，用于将原始音频中的人声与伴奏部分分开。

（5） 演唱语音转录： 将歌声的wav文件输入FL Studio5，它是一个数字音频工作站软件，可以自动生成初步乐谱，捕捉每个音符的音符音高和起止时间。

（6） 单词边界标注：研究人员通过Pypinyin将歌词转换为音素序列，然后使用音频对齐工具Montreal Forced Aligner7获取歌词中每个单词的起止边界。

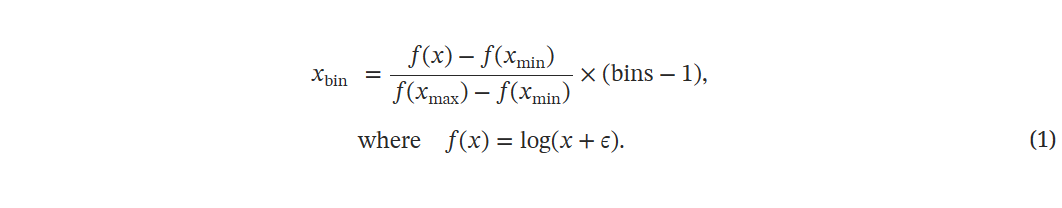
（7） 单词级对齐：动态时间扭曲 （DTW）算法被用于根据起始时间在单词等级对齐歌词和旋律。

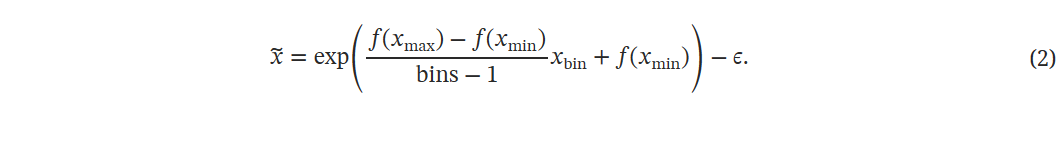
最后，研究人员开发了一个包含 15K 对歌词旋律的数据集，其中大约 5K 是中文，10K 是英文。

**4.SongComposer**

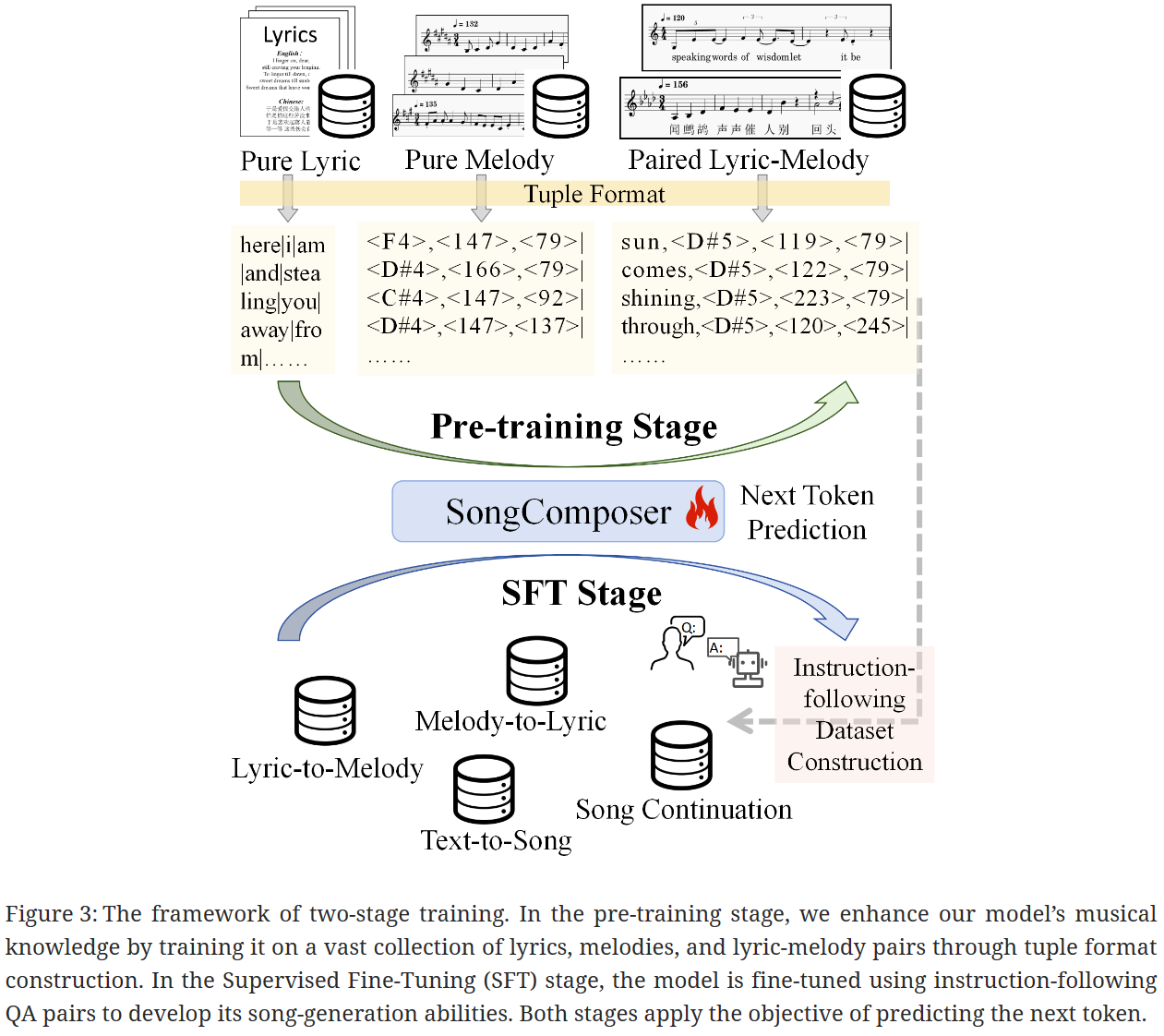
在本节中，研究人员从两个方面阐述了研究人员的 SongComposer。 首先，为了利用大型语言模型 （LLM） 功能的强大功能，研究人员在第 4.1 节中提出了一个将符号表示歌曲转换为对大语言模型友好的格式的原则性策略。然后，按照这个基本原则，第 4.2 节深入研究了一个两阶段的训练范式，该范式使用了研究人员收集的数据集并使模型能够创作歌曲。

**元组数据格式：**当研究人员考虑如何将歌词和旋律输入到大型语言模型中时，一个显而易见的策略是将它们按顺序输入，首先是歌词序列，然后是音符值和时间长度的序列。然而，在这种处理方式之下，模型在歌词和旋律对齐方面存在困难，会导致输出出现错位。为解决此问题，研究人员提出一种基于元组的统一输入组织方式，其中每个元组代表一个离散的音乐单元，可以是歌词、旋律或歌词-旋律对。具体来说，在纯歌词数据的情况下，研究人员将单个单词插入到每个元组中。对于纯旋律，研究人员在每个元组中包括音符音高、音符时长和休止时长。对于歌词-旋律对，研究人员将歌词和相应的音符元素合并到同一个元组中。由于一个单词可能对应多个音符，因此一个元组可能包含一个单词以及多个音符和它们各自的时长。如图3所示，研究人员使用竖线（|）来分隔不同的元组，并将元组序列作为输入传递给大语言语言模型。这种元组格式明确地为模型提供了歌词和旋律之间的相互映射关系，有助于模型把握元素之间的对应关系。

**离散化时长：**为了有效地处理旋律中的时长信息，研究人员采用对数编码将时长分类为到预定义数量的区间，从而将连续的时长范围转换为一组离散值。对于大语言模型而言，这种组织良好且简洁的输入可以增强模型精确捕捉音乐中时间变化的能力。具体来说，根据时长的统计分布，研究人员将它们裁剪到 [xmin,xmax] 的范围内。对于任何时长 x，编码过程定义如下：

这里研究人员设置[xmin,xmax]=[-0.3,6]，bins=512，ε=1，则可以将长达 6.3 秒的连续时间编码为 512 个离散化整数。 相反，为了将离散化值解码回其原始持续时间，研究人员使用

通过使用对数方法，研究人员可以为较短的时长分配更细粒度的区间，而为较长的时长使用较粗粒度的区间。这种自适应的区间划分策略能够更高效地表示时间，也更符合人类的感知特点，即人类往往对区分较短的时长更为敏感。

**词汇扩展：**为了确保清晰性，研究人员特意引入用 ⟨⋅⟩ 符号作为词汇中离散时间单位和音符值的表示。对于时间的离散化，研究人员扩展了词汇表，增加了512个独特的tokens，表示为 ⟨xbin⟩，其中每个token代表一个特定的离散化的时间区间（temporal bin）。对于音符的表示，研究人员将10个八度音阶中的12个音高类别进行标记化，从而添加了120个不同的音乐标记。这些新引入的词汇表和最终的语言建模头在整个训练过程中都得到了优化，而无需任何复杂的初始化。

**4.2两阶段训练**

**预训练阶段：**为了丰富 LLM 的基本音乐知识，研究人员首先在纯歌词和纯旋律的大量语料库上预训练模型，并进行下一个token预测。值得注意的是，对于旋律，研究人员采用音高转换来增强研究人员的数据集。具体来说，研究人员在两个方向上将音高最多了移动四个半音，从而将研究人员的数据集大小扩大了九倍。 通常，研究人员将整个歌曲数据分解为基于行的样本，每个样本包含五到十行。这种表述为模型提供了一个更有针对性和相关性的学习环境，从而促进了更连贯的歌曲生成。 通过这种纯粹的歌词和旋律训练过程，研究人员的模型获得了坚实的音乐知识基础，为专注于歌词和旋律之间对应关系的更高级学习奠定了坚实的基础。 在此之后，研究人员使用成对数据来指导模型生成对齐的歌词-旋律对。根据经验，为了保持模型分别处理旋律和歌词的能力，研究人员按照 1：1：1 的采样率将等量的纯旋律、纯歌词和成对数据输入模型。

共计有 0.23B 的歌词tokens、0.28B 的旋律tokens和 0.16B 的成对数据tokens。所有数据均使用元组数据格式输入。

**监督微调阶段：**在将大量乐谱和歌词输入到模型之后，研究人员为【歌曲生成】（song-generation）任务整理【指令遵循】（instruction-following）方面的数据，这些任务包括为给定的歌词创作旋律、为旋律编写歌词、扩展歌曲片段以及根据文本描述生成歌曲。为了促进这些目标的实现，研究人员手动为三个任务分别准备了3K个问答对。此外，对于最后一个任务，研究人员使用GPT-4生成了1K首歌曲摘要，这些摘要进而形成了一个由文本到歌曲的数据集，指导歌曲创作过程。

**训练细节：**研究人员采用InternLM-7B作为研究人员的基础模型，并训练模型根据先前的文本预测下一个token，以最大化给定样例中token的对数似然性（log-likelihood of tokens）。在优化方面，研究人员使用AdamW优化器，学习率为10−5，β1=0.9，β2=0.95，权重衰减为0.1。整个数据集迭代一次的batch大小为8。此外，在训练的最初的1%中，应用学习率的线性预热（a linear warm-up of the learning rate），从10−6增加到10−5。之后，应用余弦退火策略（cosine schedule）调整学习率，将学习率降低到最小值0。这一设置在预训练和监督微调阶段均保持一致。SongComposer的训练在16个Nvidia A100（80G）GPU上进行了大约2天。

**5实验**

**5.1客观评价指标**

研究人员构造了一个包含 1188 首歌曲的验证集。其中，415首是中文歌曲，773首是英文歌曲，这些歌曲在训练阶段都没有接触过研究人员的模型。在随后的测试中，研究人员将几首歌曲随机分割成 5 到 10 个句子。 客观评估的重点是评估生成的输出与真实情况之间的相似性。

**旋律生成：**对于旋律评估，研究人员采用SongMASS提出的指标。这包括音高分布相似度 （PD）、持续时间分布相似度 （DD） 和旋律距离 （MD）。 由于与SongMASS的校准策略存在差异，研究人员对音符的音高和时长进行了后期处理，以实现与标准的相统一。值得注意的是，研究人员通过加法运算调整真实值和生成音符的平均音高，使它们对齐。对于时长，研究人员首先计算生成音乐与真实值之间的平均时间比，然后使用该比例通过乘法缩放时长。在将节拍速度（bpm）设置为60后，研究人员将1/16秒转换为1/16音符，从而能够进行更一致的比较。

**歌词生成：**在歌词生成方面，研究人员使用CoSENT（Cosine Sentence）模型的基础多语言版本来评估句子级别的对齐情况。该模型允许研究人员输入歌词，并计算生成歌词与原始歌词之间的余弦相似度。此外，研究人员还采用了ROUGE-2得分和BERT得分作为评价歌词生成的另外两个指标。ROUGE-2得分主要关注生成文本与原始文本之间二元组的重叠情况。至于BERT得分，研究人员利用BERT基础多语言，不区分大小写的模型，基于上下文嵌入来测量文本相似度。

**5.2主观评价指标**

对于主观评估，研究人员进行了一项用户研究，有 30 名参与者，每项任务 10 个样例。研究人员为每项任务制定两个指标，并要求参与者对它们进行评分。评分范围是1到5分，其中分数越高表示质量越好。通过这种方式，研究人员从用户的角度收集有关生成内容质量的反馈。

**Lyric-to-Melody Generation**要求根据给定的歌词创建合适的旋律。旋律的评估依据是： （1）和谐度（Harmony，即HMY.）：评估旋律的整体质量。

（2）旋律-歌词兼容性 （Melody-Lyric Compatibility，即MLC.）：检查生成的旋律与给定歌词的契合程度。

**Melody-to-Lyric Generation** 旨在制作与所提供旋律相匹配的歌词。歌词的评估依据： （1）流利度（Fluency，即FLN.）：考虑生成的歌词的语法正确性和语义连贯性。

（2） 旋律-歌词兼容性 （MLC.）：检查生成的歌词与给定旋律的拟合程度。

**Song Continuation**涉及在旋律和歌词上扩展给定的歌曲片段。研究人员从以下方面评估延续质量：

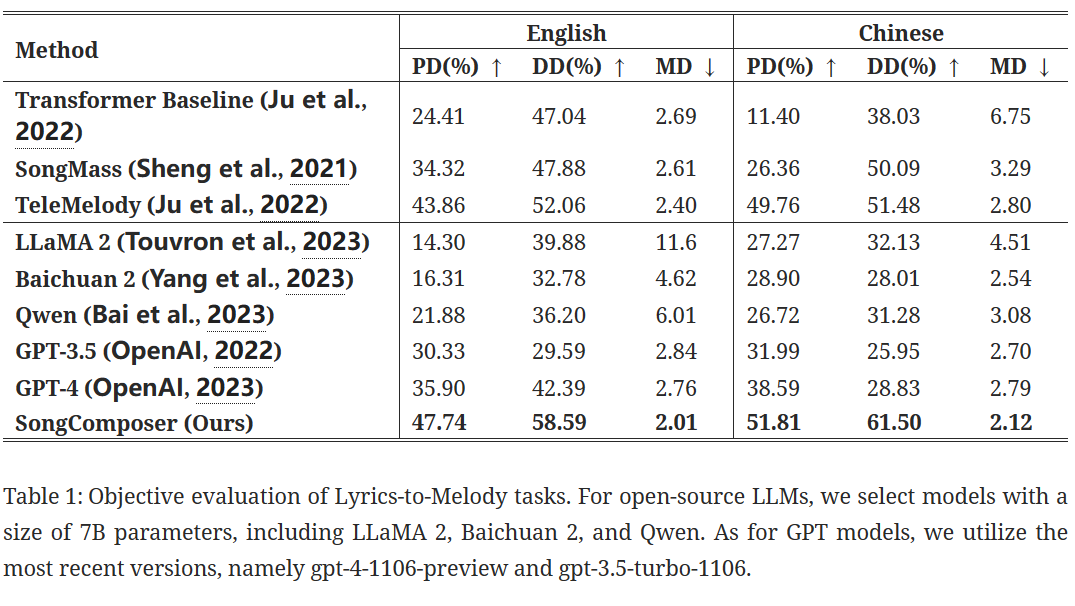
（1） 整体质量 （Overall Quality，即OVL.）：根据其音乐吸引力来衡量生成的歌曲的整体质量。

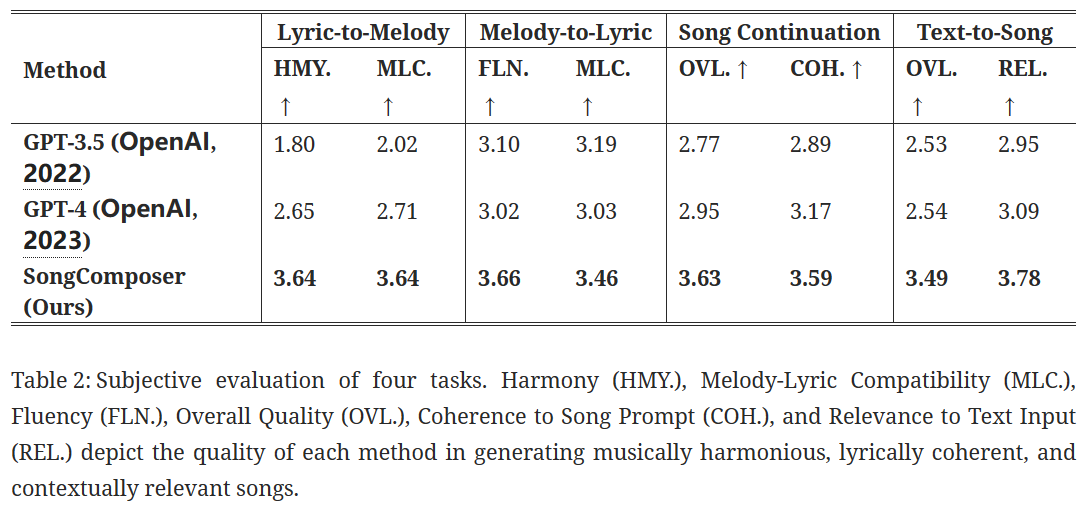
（2）与歌曲提示的连贯性（Coherence to the Song Prompt，COH.）：分析续写部分与提供的歌曲提示的吻合性，评估旋律、歌词和其他音乐元素的连贯性。

**Text-to-Song Generation** 根据文本描述生成一首完整的歌曲，在音乐和抒情上捕捉其精髓。评估的重点是：

（1） 整体质量 （OVL.）：根据其音乐吸引力来衡量生成的歌曲的整体质量。

（2） 与文本输入的相关性 （Relevance to the Text Input，即REL.）：检查生成的歌曲与输入文本的一致性和从中得出的相关性。

总的来说，对于每项任务，一个指标评估所制作样本的整体音乐质量，而另一个指标衡量相应任务的具体挑战。

****

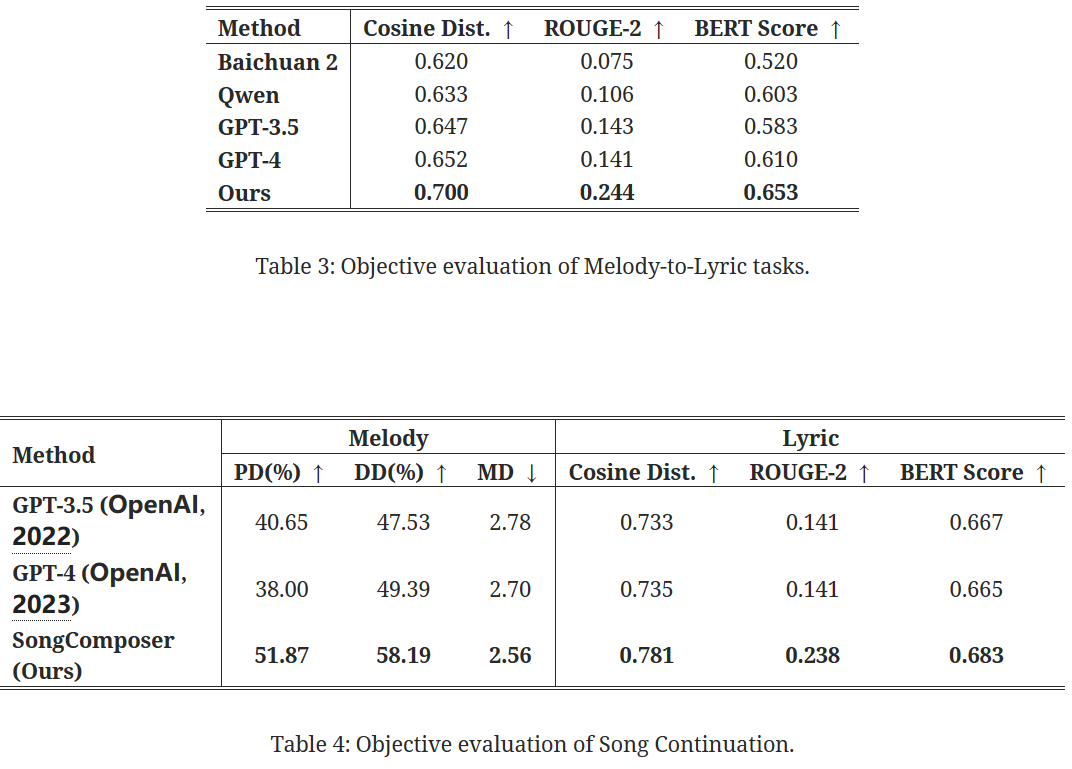
**5.3结果**

研究人员主要测试和比较研究人员的方法与现有的LLM。 对于替代性的LLM 基线，研究人员采用少样本（few-shot）提示方法，提供样本示例来提示 LLM 并按照给定的指令生成所需的输出。详情见附录C。 研究人员通过 GPT-4 和 GPT-3.5 的 API 收集输出。此外，研究人员还评估了其他典型的 LLM，其权重是从 Hugging Face 社区获得的。

**从歌词到旋律的生成：**研究人员客观地从英文和中文两方面评估歌词到旋律的能力。如表 1 所示，研究人员的方法在音高分布相似度 （PD） 和持续时间分布相似度 （DD） 方面比 GPT-4 等高级语言模型高出 10 分以上，并且在两种语言中都实现了 0.6 低的旋律距离 （MD）。它展示了研究人员在将音乐知识无缝集成到 SongComposer 方面的成功。此外，研究人员的方法超越了强大的传统模型TeleMelody的性能，这是一个两阶段的歌词到旋律生成系统，采用精心设计的音乐模板来提高生成旋律的质量和可控性。有趣的是，研究人员是通过直接预测下一个token实现这一超越的。研究人员还将主观评价制成表2。值得注意的是，与 GPT-3.5 和 GPT-4 相比，SongComposer 在和声和旋律歌词兼容性方面都表现出显着的优势，这表明其在将歌词线索转化为相应旋律表达方面的先进能力。

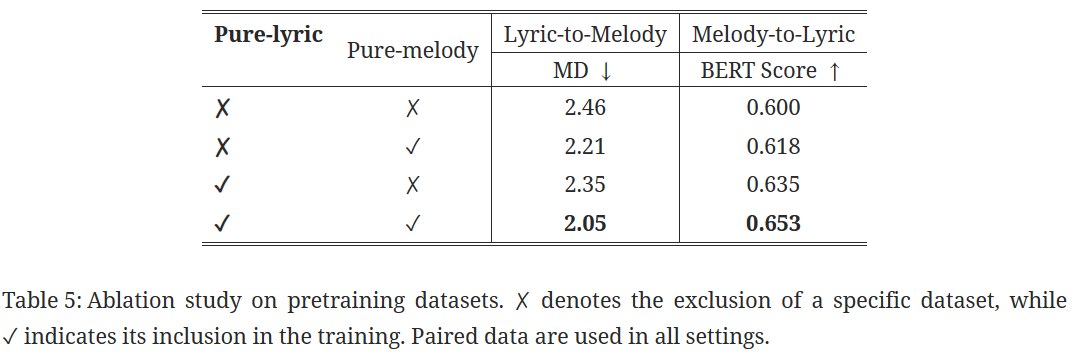
**旋律到歌词的生成：**如表3所示，客观评估表明，研究人员的方法在生成与原始歌词非常匹配的歌词方面是有效的。 在表 2 所示的主观评估中，SongComposer 在旋律到歌词生成领域的表现优于领先的 GPT 模型。用户反馈始终支持研究人员的 SongComposer 的卓越能力，可以制作出不仅更流畅而且高度相关的歌词，有效地捕捉和补充潜在的旋律节奏。

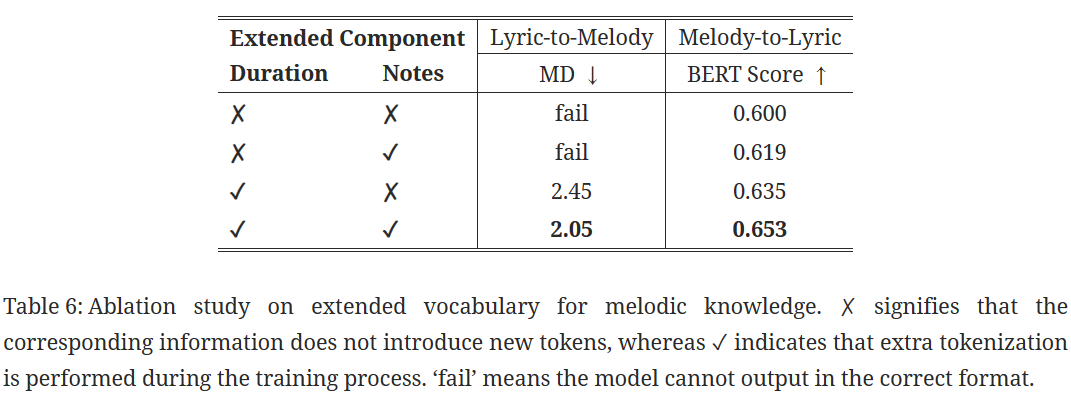
**歌曲续写：**如表 4 和表 2 所示，SongComposer 不仅可以单独生成高质量的歌词和旋律，而且擅长将两者融合在一起，这体现在它的高 COH 得分上，展示了有效协调歌词和旋律元素的能力。

**文本到歌曲生成：**研究人员还展示了文本到歌曲生成的主观评估，如表 2 所示。这突显了 SongComposer 将文本提示转化为歌曲的惊人能力，这些歌曲不仅具有音乐连贯性，而且有效地捕捉了文本中嵌入的主题和情感。

**5.4消融研究**

在消融研究中，研究人员分别展示了歌词到旋律和旋律到歌词任务的旋律距离 （MD） 和 BERT 分数。

**预训练数据集：**为了探索专门数据集对模型学习的影响，研究人员使用具有纯抒情和纯旋律数据集不同组合的配对数据进行训练实验。表 5 表明，省略纯抒情和纯旋律数据集会显着降低表现，强调了基础旋律和抒情知识在培训早期阶段的重要作用。单独集成任何一个数据集都可以在任务中提供显着的改进。具体来说，纯歌词数据集主要增强了旋律到歌词任务的性能，而纯旋律数据集则为旋律生成提供了更实质性的提升。这一观察结果与直观的理解一致，即每个数据集都丰富了模型对其各自模态的理解。最后，同时使用这两种类型的数据集会产生最有利的结果，显示出协同效应，突出了在预训练期间结合抒情和旋律数据的价值，以获得更全面的学习成果。

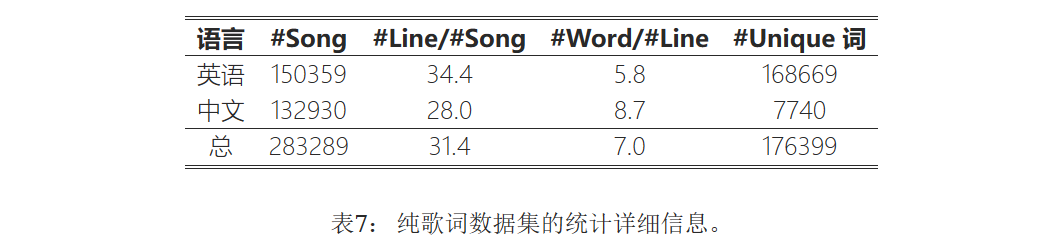
**旋律知识的扩展标记：**为了了解旋律知识的专用标记是否有利于 SongComposer 的性能，研究人员在表 6 中对不同的标记组合进行了消融。研究人员评估了它们对模型处理和生成音乐内容的能力的影响。 首先，对于持续时间标记的消融，研究人员没有应用离散数字，而是直接将原始的基于浮点数的秒数输入到 SongComposer 。 缺少特殊时间标记导致模型无法为歌词到旋律任务生成正确的结果。 相反，无论是否引入音符标记，结合离散持续时间标记化都会大大提高旋律到歌词任务的性能，将 BERT 分数提高 0.03 以上。验证了持续时间离散化对掌握旋律结构的重要性。 其次，音符标记的采用在两项任务中都产生了显着的改进。尽管该模型可以识别音符表示的字符串，但对它们进行特殊标记化可进一步提高清晰度并优化学习过程。

**6结论**

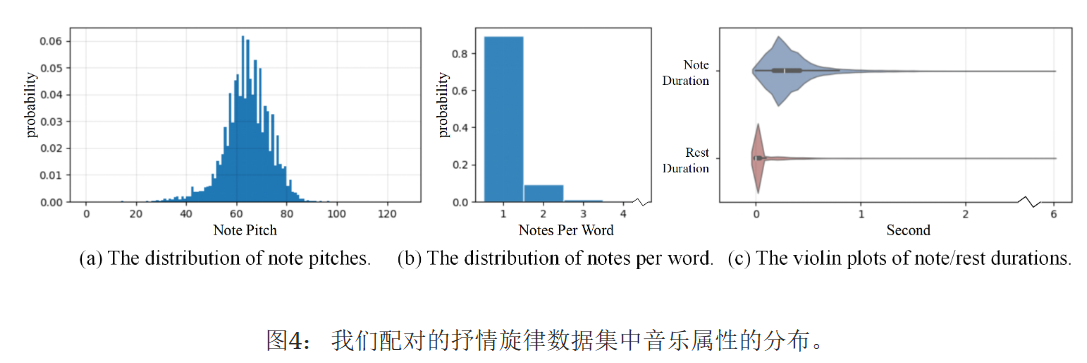
在本文中，研究人员提出了SongComposer，这是一种新颖的大型语言模型，它利用符号歌曲表示来生成旋律和歌词。 与之前的音乐 LLM 不同，这些 LLM 将音乐处理为量化的音频信号，导致效率低下和多样性有限，SongComposer 利用符号符号以更类似于人类作曲技术的方式制作音乐。研究人员集成了一种新颖的元组设计，将音符的音高、持续时间和休止符持续时间与歌词连接起来，确保精确的音乐理解和对齐。此外，研究人员还创建了 SongCompose-PT，这是一个包含歌词、旋律及其组合的综合数据集，用于大规模的预训练。在被专注于【指令遵循】能力的 10K问答对训练后，SongComposer 在各种任务中表现出色，例如将歌词转换为旋律、将旋律转换为歌词生成、歌曲延续和文本到歌曲创作，在性能上甚至超过了 GPT-4 等高级 LLM 模型。 总之，研究人员相信 SongComposer 可以为 LLM 在音乐领域的创造性应用开辟新的途径。

**附录A 数据集详细信息**

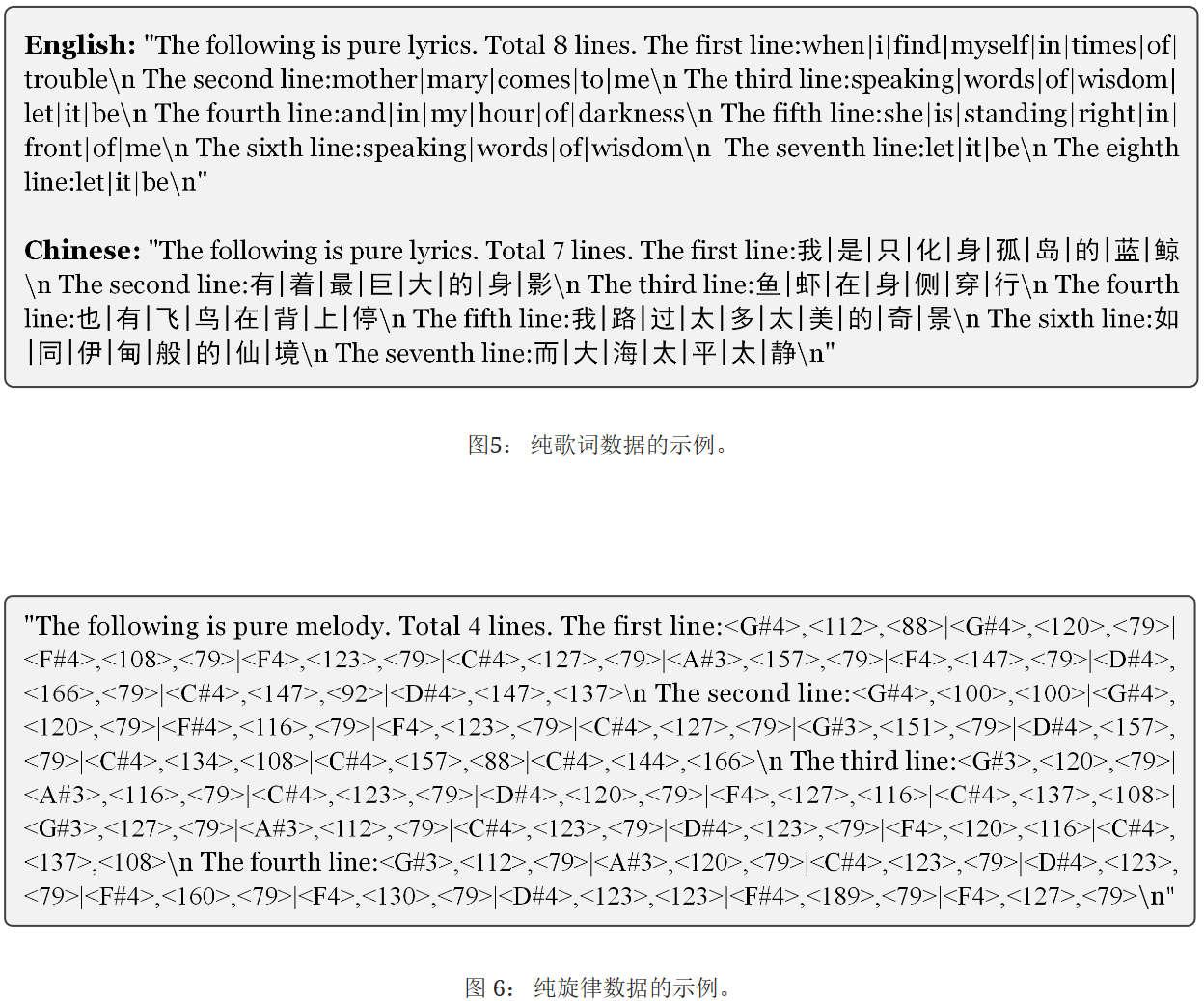
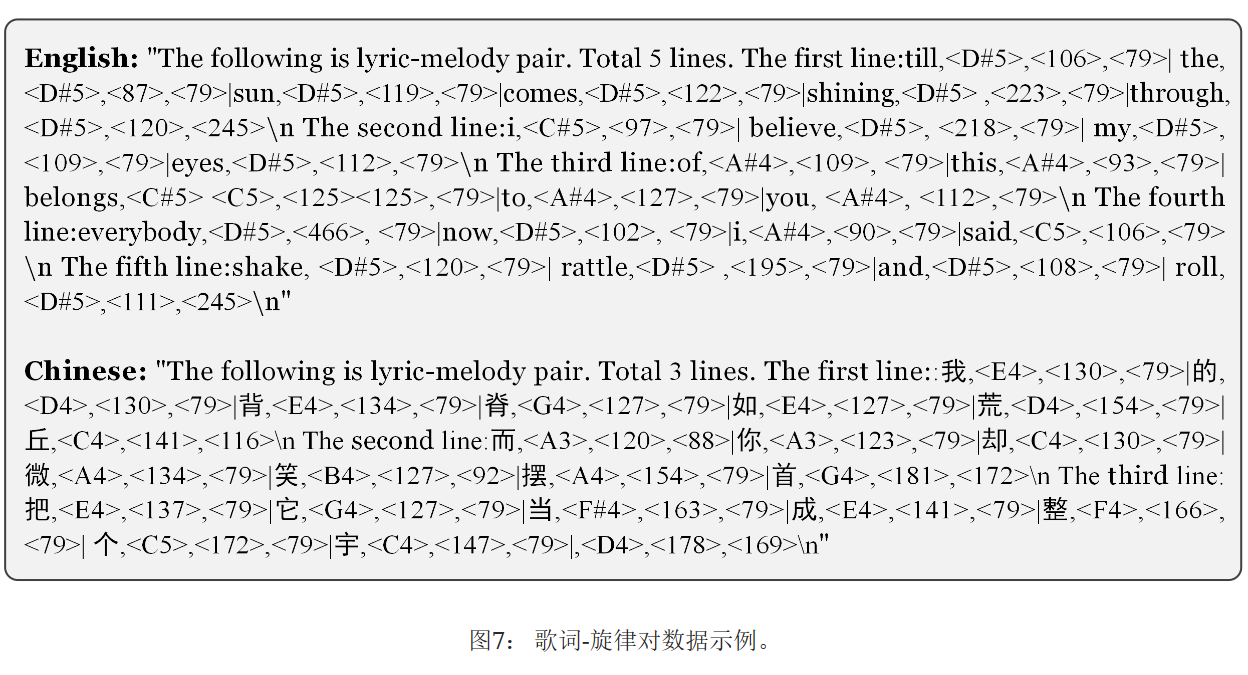
**A.1纯歌词数据集**

对于歌词清理过程，研究人员（1）删除非英文和非中文片段，（2）删除特殊符号、标点符号和空格，（3）消除一些多余和无意义的重复。 通过这些努力，研究人员从283K首歌曲中编译了一个高质量的歌词数据集，其中英文150K，中文133K。表 7 提供了数据集的详细细分，包括语言分布、每首歌曲的平均行数、每行的单词数和唯一单词的数量。

**A.2成对的抒情旋律数据集**

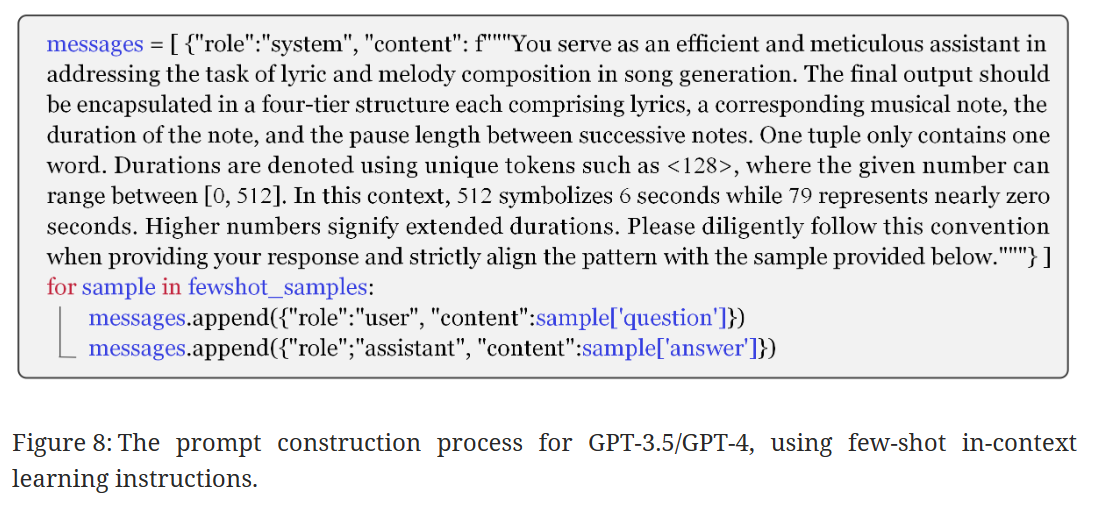
研究人员还对图4所示的成对歌词-旋律数据集进行了统计分析。研究人员发现大多数音高数字在 50 到 80 的范围内，大多数单词与一个音符配对，大约 10% 的单词对应于两个或多个音符。在检查音符持续时间时，研究人员观察到它们主要在 0 到 1 秒之间变化，休止符持续时间主要为零，反映了简洁的音乐结构。

**附录B元组格式示例**

在预训练阶段，研究人员引入了三种类型的数据，它们都采用统一的元组结构进行格式化。每个数据条目都以短语“以下内容是......”开头。来介绍内容。在每个条目中，每行都以提示开头，例如“第一行/第二行/第三行：”，并以换行符 （“ \n”） 结尾以指示分隔。如图 5 所示，研究人员展示了纯歌词的中文和英文实例。纯旋律的结构如图 6 所示。对于抒情旋律对，双语版本如图 7 所示。这种基于元组的方法能够系统地表述各种数据类型。

**附录C基线建设**

**C.1GPT**

研究人员调用 GPT API 来检索基线结果。研究人员利用少样本（few-shot）提示来提供一个模板并指示模型效仿。伪代码如图 8 所示。

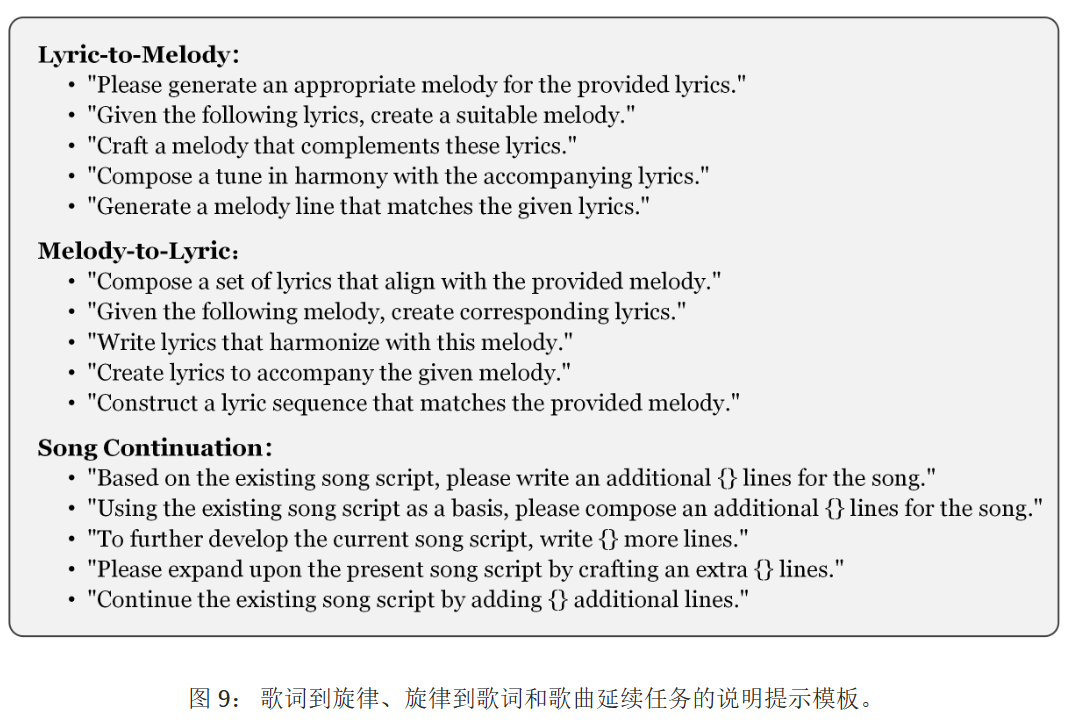
**C.2开源LLM**

对于开源 LLM，研究人员选择基础模型作为所有候选人的公平比较。LLM 的提示结构如下

“system messages: ​Q​1→A​1，Q​2→A2，Q​3→"

其中系统消息与 GPT 的消息相同，Q1,Q​2,和A​1,A​2是研究人员希望模型执行的任务的示例。研究人员指示模型生成A​3作为此提示的延续。

**附录D说明模板**

研究人员在图 9 中为三个不同的任务提供了提示模板，这些模板是编译 QA 对的基础。例如，在歌词到旋律的任务中，研究人员从指令提示开始，例如“请为提供的歌词生成适当的旋律”。然后按照提示播放歌曲的纯歌词版本。对于歌曲延续任务，研究人员将另外指定研究人员希望模型扩展歌曲的行数。