**第二部分：方法 - 描述程序设计和详细方法，并附上一些图表。**

在自然语言处理任务中，数据预处理是关键步骤。对于 VADER 算法，首先需导入相关库，通过analyzer = SentimentIntensityAnalyzer()创建情感分析器实例。然后从外层（语言）到内层（训练集等）遍历数据，在这个过程中，提取出分词后的数据（tokenized\_data）以及原始未分词数据（original\_data）。针对每个分词后的句子，利用创建的分析器通过analyzer.polarity\_scores(sent)获取情感得分，将这些得分添加到sentiment\_scores列表中，最后汇总各数据集类型的结果字典，以语言为键，形成包含所有语言及各数据集类型情感分析结果的字典，从而完成 VADER 算法的数据预处理。

而对于 Word2Vec 算法，同样要先遍历不同语言的数据，从外层到内层进行操作。在训练阶段，将句子数据用于模型训练，训练结束后得到训练好的 Word2Vec 模型。在使用时，依据给定的单词列表 “words”，借助模型的词向量字典获取每个单词对应的词向量，并以列表形式返回。此外，为了可视化效果，可选取部分单词的词向量，在训练后的 Word2Vec 模型中获取其对应的向量，再通过 t - SNE 等降维方法，将高维向量映射到二维或三维空间，制作散点图等可视化图表，以便直观观察单词向量间的关系，辅助理解数据特征，进一步提升数据预处理的效果和对数据的理解程度。

图表：

Vader：表格

描述已自动生成

Word2Vec：  
图表

中度可信度描述已自动生成

**第三部分：实施细节 - 总结程序实施的详细信息。**

**VADER:**

\*\*整体功能\*\*：

该函数使用VADER模型对通过`load\_sentiment\_data`函数加载的数据进行情感分析，并以特定的字典结构返回分析结果。

\*\*步骤及作用\*\*：

1. \*\*初始化分析器\*\*：

- 代码行：`analyzer = SentimentIntensityAnalyzer()`

- 作用：创建一个VADER情感分析器实例，用于后续对句子进行情感分析，获取情感得分。

2. \*\*遍历语言数据\*\*：

- 代码段：`for lang, lang\_data in data\_dict.items():` 及内部相关循环

- 作用：

- 外层循环遍历加载数据字典中的每种语言（如印尼语、英语、爪哇语等）。

- 对于每种语言，内层循环将进一步处理该语言的训练、验证、测试数据。

3. \*\*处理各数据集类型\*\*：

- 代码段：`for set\_type in ['train', 'valid', 'test']:` 及内部相关操作

- 作用：

- 针对每种语言，分别处理其训练集、验证集和测试集数据。

- 首先提取出分词后的数据（`tokenized\_data`）和原始未分词数据（`original\_data`）。

4. \*\*分析句子情感并记录得分\*\*：

- 代码段：`for sent in tokenized\_data:` 及内部相关操作

- 作用：

- 遍历分词后的句子数据。

- 对每个句子，使用之前创建的分析器通过`analyzer.polarity\_scores(sent)`获取情感得分，并将得分添加到`sentiment\_scores`列表中。

5. \*\*构建返回结果字典\*\*：

- 代码段：`analyzed\_lang\_data[set\_type] = (sentiment\_scores, original\_data)` 及相关操作

- 作用：

- 将分析得到的情感得分列表和原始数据组成元组，作为对应数据集类型（训练、验证、测试）的值。

- 然后将各数据集类型的结果字典汇总到以语言为键的字典中，最终返回这个包含所有语言及各数据集类型情感分析结果的字典。

**Word2Vec：**

\*\*整体功能\*\*：

这段代码主要实现了基于机器翻译数据训练Word2Vec模型，并能从训练好的模型中获取指定单词的词向量。

\*\*步骤及作用\*\*：

**\*\*`train\_word2vec\_on\_mt\_data`函数\*\*：**

1. \*\*数据准备\*\*：

- 代码段：`for lang, lang\_data in mt\_data\_dict.items():` 及内部循环

- 作用：遍历由`load\_mt\_data`函数加载的机器翻译数据字典，针对每种语言的训练、验证、测试数据集，提取分词后的数据并进一步处理成单词列表形式，将所有这些单词列表合并到`all\_sentences`中，为训练Word2Vec模型准备数据。

2. \*\*模型创建与参数设置\*\*：

- 代码行：`model = Word2Vec(sentences=all\_sentences, vector\_size=100, window=5, min\_count=1, workers=4)`

- 作用：使用`gensim.models.Word2Vec`创建Word2Vec模型实例，设置相关参数，如`vector\_size`指定词向量维度为100，`window`设置上下文窗口大小为5，`min\_count`表示单词最少出现次数为1才会被纳入训练，`workers`设定训练使用的线程数为4。

3. \*\*模型训练\*\*：

- 代码行：`model.train(all\_sentences, total\_examples=len(all\_sentences), epochs=10)`

- 作用：使用准备好的所有句子数据对模型进行训练，指定总示例数为`all\_sentences`的长度，训练轮数为10，让模型学习单词的分布式表示。

4. \*\*返回训练好的模型\*\*：

- 代码行：`return model`

- 作用：将训练好的Word2Vec模型返回，以便后续使用。

**\*\*`get\_word\_vectors`函数\*\***：

1. \*\*获取词向量\*\*：

- 代码行：`return [model.wv[word] for word in words]`

- 作用：从传入的训练好的Word2Vec模型中，根据给定的单词列表`words`，通过模型的词向量字典`wv`获取每个单词对应的词向量，并以列表形式返回，列表中的元素为表示词向量的numpy数组。

**第四部分：实验 - 评估方法并分析结果，包括定性和定量分析，附上表格、图表和讨论。**

**VADER:**

定性分析：

在测试集上应用做出的混淆矩阵，可以看到，在英语测试集上，模型的准确率比较高。

图表, 树状图

描述已自动生成 图表, 树状图

描述已自动生成 图表

描述已自动生成

* 定量分析：
* Accuracy（准确率）：衡量模型预测正确的样本数占总样本数的比例，反映模型整体预测准确程度。
* Precision（精确率）：在预测为正例的样本中，真正例所占的比例，体现模型对正例预测的精准度。
* Recall（召回率）：实际为正例的样本中，被模型正确预测为正例的比例，反映模型对正例的捕捉能力。
* F1（F1 分数）：综合考虑精确率和召回率，是两者的调和平均数，用于全面评估模型在正负例预测上的平衡表现。

文本

描述已自动生成