算法说明文件

统一数据准备阶段

由于两者的输入要求不同，这里先对不同的算法进行输入数据的准备，使用`data_generatior.py`文件进行此步骤。统一的操作为：

- 读取`lmtraining.txt`，按照英文单词的词粒度进行分割。
- 使用en_core_web_sm词库中的停用词*(stop_words)*库进行停用词的去处，来增加文章蕴含的语义信息量。
- 使用collections中的Counter函数来得到文段中过低频的词语*(过低频词也很难学习到足够的语义信息，没有存在的意义)*，一般设置为5-10，由于词库过大，为了压缩到可以在可用资源上运行的大小设置为了100。
- 用过滤后的列表生成词库字典，标为words_dict[word] = i，i为索引数字。
- 用词库将去掉了停用词的列表*(注意不是去掉过低频词的，因为要避免不在一个句子的词语由于中间词全部被删除了而错误的靠近在一起)*转化为token列表，如果不在词库里*(过低频词)*，就设置为一个特殊词-1，以便后续处理。
- 把字典和token列表分别存在`words_dict.txt`和`numerical_list.txt`中，为后续使用准备。

对于sgns需要不同形式的数据，在另外处理，同样使用`data_generatior.py`文件：

- 遍历每一个token列表的单词，然后读取其两边`K=2`长度的单词作为目标，如果不在边界上，并且该数值不为-1(过低频词)则生成(单词，目标)的单词对，把单词对组成的列表储存在`sgns_training_data.txt`中。

SVD方法

训练流程

1. **数据准备**：
   - 从`words_dict.txt`文件中读取词典，该词典将每个词映射到一个唯一的整数ID。
   - 从`numerical_list.txt`文件中读取并分割为一个数字列表，这个列表代表文本中单词的顺序和出现的词汇ID。
2. **构建共现矩阵**：

- 初始化一个零矩阵`vocab_matrix`，其尺寸为词典大小的平方，用于存储词汇间的共现频次。
- 遍历数字列表，设置窗口大小`K=5`，为每个单词与其邻近单词（窗口内）的共现次数进行计数。

3. **执行SVD**：
   - 对共现矩阵进行奇异值分解，得到矩阵`U`（左奇异向量），`S`（奇异值向量），`V`（右奇异向量）。
   - 选取前`k=100`个奇异值及其对应的奇异向量进行降维。
4. **计算和输出相关数据**：
   - 计算非零奇异值的总数。
   - 计算选取的前`k`个奇异值之和。
   - 计算所有奇异值之和。
   - 计算选取的奇异值之和与全部奇异值之和的比例。
5. **降维数据的使用**：
   - 使用降维后的数据`vec_sta`（由`U_reduced`和`S_reduced`计算得到的向量）计算单词间的余弦相似度。
   - 将相似度结果添加到`wordsim353_agreed.txt`文件中，生成新文件`svd_output.txt`。

关键参数和执行细节

- **设备选择**：根据系统是否支持CUDA，选择`cuda`或`cpu`进行计算，以加速处理。
- **矩阵操作**：利用PyTorch进行矩阵运算，确保所有的数据结构都转移到相应的设备（CPU或GPU）上。
- **降维的选择**：选用了K=100，以下为此参数下的特征信息：
  - 非零奇异值的数量：11534
  - 选取的前100个奇异值之和：864735.5000
  - 全部奇异值之和：1845735.0000
  - 选取的奇异值之和与全部奇异值之和的比例：0.4685

Skip-gram with Negative Sampling（SGNS）方法

训练流程

1. **数据处理**：
   - 从`sgns_training_data.txt`中读取训练数据，每行包括一个输入词和一个上下文词的索引。
   - 数据被转换为Tensor格式，以便在PyTorch中进行操作。
2. **数据加载**：
   - 使用`DataLoader`来批量处理数据，批量大小设为256，同时进行数据的随机洗牌以提高模型泛化能力。
3. **训练过程**：
   - 总共进行10轮训练（`num_epochs=10`），每轮遍历所有的数据批次。
   - 每个批次数据通过模型前向传播生成对数概率，计算损失，并通过反向传播更新模型参数。
   - 使用`tqdm`库显示训练进度，包括每个epoch的总损失和每10000个批次的平均损失。

模型使用与评估

- **模型保存与加载**：
  - 训练完成后，模型的参数被保存到`model_skipgram.pth`。
  - 加载模型并设置为评估模式，准备进行词向量的提取和相似度计算。
- **相似度计算**：
  - 从`wordsim353_agreed.txt`读取词对，使用加载的模型提取对应词向量，就是只过嵌入层。
  - 计算词对之间的余弦相似度，并将结果写入到`sgns_output.txt`。

模型构建与参数设置

1. **模型定义**：
   - 使用PyTorch框架定义了一个`SkipGramModel`类，继承自`nn.Module`。
   - 类中包含两个主要部分：一个嵌入层`self.embeddings`用于生成词向量，和一个线性层`self.linear`用于预测上下文词。
   - 输入词通过嵌入层转换为词向量，然后通过线性层生成对应的分数，最终通过对数Softmax得到预测上下文词的对数概率。

- 训练前所有的参数均为默认初始化的值。

2. **参数配置**：
   - `vocab_size`：词典大小，由词典文件`words_dict.txt`确定，此处为11533。
   - `embedding_dim`：词向量的维度，设定为100维，这是一个常用的维度，平衡了表示能力和计算复杂度。
   - `loss_function`：使用负对数似然损失函数（Negative Log Likelihood Loss, NLLLoss），专门用于多分类问题中的对数概率。
   - `optimizer`：选择随机梯度下降（SGD）作为优化器，初始学习率设为0.01。

结果生成

将已经生成的数据整合按照格式要求生成`2021213688.txt`。