向量化技术和使用余弦相似度计算推荐分数：

当需要向量化包含导师信息的大段文字和用户输入的一句话时，可以考虑以下向量化技术：

大段文字的向量化：对于包含导师信息的大段文字，可以使用词袋模型（Bag-of-Words）的方法进行向量化。一种常用的实现是使用CountVectorizer或TF-IDF（Term Frequency-Inverse Document Frequency）Vectorizer。这些技术可以将大段文字转换为基于词频或TF-IDF权重的向量表示。这种向量化方法适用于大段文字，不考虑单词之间的顺序和语义关系，但可以捕捉到文档中单词的重要性。

用户输入的一句话的向量化：对于用户输入的一句话，可以考虑使用词嵌入（Word Embeddings）的方法进行向量化。Word2Vec、GloVe或FastText等预训练的词嵌入模型可以将单词映射为低维度的连续向量表示，其中向量的位置和方向编码了单词的语义关系和上下文信息。这种向量化方法可以捕捉到句子中单词之间的语义关系，并且在许多自然语言处理任务中表现良好。

因此，针对大段文字和用户输入的一句话，可以采用不同的向量化技术。对于大段文字，可以使用词袋模型（如CountVectorizer或TF-IDF）进行向量化。对于用户输入的一句话，可以使用预训练的词嵌入模型（如Word2Vec、GloVe或FastText）进行向量化。这样可以根据不同文本类型的特点，获得适合的向量表示，用于后续的相似度计算或推荐任务。

在一开始使用countvectorizer实验时，算法给固定的几个导师计算出很高的分数，例如Sharib Ali 和 Noorhan Abbas，他们也是文本内容最多的导师之一。这是因为CountVectorizer使用词频进行向量化，文本内容多的导师关键词也更多，所以他们经常会得到优先推荐。这种情况也说得过去，因为当某项技术或者背景被导师多次提到时，用户也在检索这项技术时，确实应该得到更高的分数。但是当用户输入的内容更加广泛时，这些导师也会得到优先推荐，因为他们的文本也比别人多包含了很多主语，连接词和动词。还有很多诸如，Research，Profile postgraduate students等与计算机技术无关的词语，所以删除这些单词可以减少干扰。

上面的代码使用了 CountVectorizer 将文本向量化，它主要基于词频进行向量表示，而不是使用预训练的词向量模型。

相比于使用预训练的词向量模型，**CountVectorizer** 的优点是简单易用，不需要下载额外的模型文件，并且可以自动处理词频信息。然而，它的缺点是无法捕捉到词语之间的语义关系，因为它只关注了词的出现频率。

使用预训练的词向量模型（如 Word2Vec 或 FastText）可以更好地捕捉词语之间的语义关系，因为这些模型是通过大规模文本数据训练得到的。预训练的词向量模型能够将相似语义的词语映射到向量空间中的相似位置。这使得在计算相似度时能够更准确地捕捉到词语之间的语义相似性。

因此，如果你关注词语之间的语义关系并希望在推荐中考虑到这些关系，使用预训练的词向量模型是更好的选择。而如果只是简单地基于词频进行向量表示，**CountVectorizer** 是一个简单有效的方法。

根据你的需求和数据特点，你可以根据实际情况选择适合的方法。

Genism pre-trained model 并不好

我测试了 fasttext-wiki-news-subwords-300

因为这些模型通常是在非常大的文本语料库上进行训练的，所以它们可以捕获很多复杂的词语关系和语义信息。

但是这些模型的问题是无法在计算机科学这个特定领域处理任务，并且有**词汇不匹配问题**：你的数据集可能包含一些预训练模型的词汇表中不存在的词汇，尤其是一些专有名词、短语或新词。

删除主语和动词会影响word2vec的哪方面判断

Word2Vec模型通过词语的上下文来理解和表示词语的含义。如果你从文本中删除主语和动词，那么将影响Word2Vec模型的以下几个方面：

1. **上下文理解**：动词和主语常常是句子的关键组成部分，它们帮助确定句子的行动和行动的主体。移除它们可能会改变或混淆模型对上下文的理解。
2. **语义关联**：Word2Vec模型的一个优点是它能够理解词语之间的关联和相似性。例如，模型可能会理解"cat"和"dog"有相似的上下文，因为它们都可能与"pet"、"animal"等词语一起出现。如果你删除主语和动词，这可能会改变这种关联和相似性的表示。
3. **句法信息**：Word2Vec模型能够捕捉一些句法信息，比如单词的词性、语态等。如果你删除主语和动词，模型可能无法有效地捕获这些信息。

请注意，这些影响在一定程度上取决于你的具体任务和应用。有些任务可能并不需要强大的上下文理解或语义关联。然而，如果你的任务依赖于理解句子的具体含义或者词语之间的关系，那么删除主语和动词可能会对模型的性能产生负面影响。

word2vec-google-news-300

推荐的效果可以接受，但是由于是通过上下文理解词语的含义，这个模型对于计算机领域的单词有些不理解，可能会误将某单词理解成其他单词，推荐出不符合用户输入的导师。

总体来说，Word2Vec因为通过上下文理解词义，他理解的词义可能与真实情况相反。

例如text analysis是一个东西，但是他可能接受到text analysis后，给你推荐介绍里包含其他单词+analysis的导师，导致出现歧义。

**Word2Vec**：这种方法将文本中的每个词转化为向量，而且这个向量可以捕捉到一些语义特性，比如相似的词会有相似的向量。Word2Vec实际上包含两种模型：CBOW（连续词袋模型）和Skip-gram模型。在CBOW模型中，我们根据上下文预测当前词，而在Skip-gram模型中，我们用当前词预测上下文。Word2Vec的主要限制是它只考虑单个词，不考虑文本的整体语义。

**Doc2Vec**：Doc2Vec（也称为段落向量或句子向量）可以被看作是Word2Vec的扩展，不仅可以将词转换为向量，也可以将较大的文本单元（如句子、段落或整个文档）转换为向量。这使得我们能够考虑到更多的上下文和词序，从而更好地捕捉文本的整体含义。

在许多任务中，例如文本分类或情感分析，Doc2Vec通常会比Word2Vec表现得更好，因为它可以考虑更大的上下文和词序信息。然而，哪种方法最好取决于具体的应用场景和数据。

使用Doc2Vec将导师信息当做一个整体向量化后在将用户输入使用**Doc2Vec计算余弦相似度。**

**经过实验发现，使用Word2Vec的模型都会吧amy lowe当做software engineering and web application的第一位推荐，因为她直接在档案里连续地描述了这两个单词，但他档案的描述只有一句话。另一方面，她并不在**Doc2Vec的前几位，而Doc2Vec推荐的前几位相对拥有更长的文字描述，他们也具有软件和web开发背景，但并没有直接，连续地描述**software engineering and web application这句话，所以**Doc2Vec对于长文本拥有更强的语义理解性。

**设计了两种文本清理方法，第二种在第一种的基础上使用了spacy过滤了动词，连接词，主语，只保留了名词、形容词、副词和专有名词。**

**1. 原以为会对CountVectorizer，因为它只关心词语是否出现，以及出现的频率，而不考虑词语在其他文档中出现的频率。因此，它可能过度强调在所有文档中频繁出现的词语（例如“the”、“is”等常见词）。所以删除掉这些单词后，能减少无用词语对推荐系统的干扰。**

**实验后发现，清理主语动词连接词后与未清理的结果差别很小，只是在导师推荐的排名上有区别.**

**并且经过分析后发现导师信息和用户输入全部使用CountVectorizer能达到最好的效果。因为CountVectorizer更适合主题非常特定的文档。**

**2. TF-IDF 的基本思想是：如果某个词或短语在一篇文章中出现的频率高（Term Frequency，即TF），并且在其他文章中很少出现（Inverse Document Frequency，即IDF），则认为此词或者短语具有很好的类别区分能力，适合用来分类。**

**"TF"（Term Frequency）表示词条在文档 d 中出现的频率，TF 的主要思想是：如果某个词在文章中出现的次数越多，那么其对文章的表达能力就越强。但是，如果一个词在文章中出现的频率很高，但在其他所有的文章中也都出现，那么这个词就不能很好地反映文章的特性，因此需要 IDF 来调整。**

**"IDF"（Inverse Document Frequency）表示词条在所有文档中出现的频率的倒数。IDF 的主要思想是：如果包含词条 t 的文档越少, IDF 越大，则说明词条具有很好的类别区分能力。**

**所以TF-IDF（词频-逆文档频率）向量化方法对高频出现的词语进行了惩罚，但如果一个词在多数文档中都频繁出现（如主语、动词、连接词等），那么它的重要性将被大大降低。TF-IDF认为，如果一个词在文档中频繁出现，但在其他文档中很少出现，那么这个词可能就很重要，或者至少可以有效地区分这个文档和其他文档。**

**综上，TF-IDF在许多情况下可能会提供更好的结果，因为它能够减少常见词语的影响，并突出显示可能更具区分性的词语。但是，哪种方法更好可能取决于特定的应用场景和数据集。**

**TF-IDF提供了一种更纯净的方法，不用对原文档删除词语，得到了和CountVectorizer一样优秀的效果。D**

实验1：

使用Bart总结Dr Ban Al Jassani的内容，将总结的内容变为用户输入的一句话，用这句话来运行不同的推荐算法，若能准确地将Dr Ban Al Jassani作为第一名推荐出来，即证明这个算法有效。

结果

自训练Word2Vec，Doc2Vec无法将Ban成功推荐，Google Word2Vec将Ban放到了第三名。而Jaccard相似度算法将，CountVectorizer, TFIDF成功将Ban放到了第一名。

(Jaccard相似度是用于比较两个集合相似度的一种方法。它的定义是两个集合的交集大小除以两个集合的并集大小。在这个案例中，每个文本（导师描述和用户输入）都被视为一个词的集合，然后计算这些集合之间的Jaccard相似度来衡量文本之间的相似性。)

通过词频进行向量化的推荐系统基本上无法分辨计算机领域同一单词的区别，例如Visual Basic和Visual C++,都有visual这个单词，但是推荐系统看到我输入了这个单词后经常会把档案中拥有CV或提到visual analytics等单词的导师推荐给我。根据语义向量化的推荐系统也会在这个问题上犯错，他们无法也很难分辨visual这个单词在计算机语言/IDE的应用和在计算机学科和技术的应用的区别。我表现最好的word2vec模型是谷歌根据新闻上的文字训练的，如果能得到根据计算机领域专门训练的模型，那么就能解决这个问题。

文本生成—总结：

brat生成、T5生成、TextRank都是用于生成文本摘要的方法，但它们在实现和原理上有一些区别：

1. **brat生成**：brat生成是一种基于规则的文本摘要方法。它使用一系列预定义的规则和模式，通过匹配关键词、短语或语法结构等方式来生成摘要。这种方法通常需要手动编写规则，并且对于不同的文本类型和任务需要进行定制化。brat生成的优点是灵活性高，可以根据需求进行精细的控制，但也容易受限于规则的准确性和覆盖范围。
2. **T5生成**：T5生成是基于预训练的语言模型，如T5（Text-To-Text Transfer Transformer）的生成能力。T5模型通过在大规模文本数据上进行训练，学习到语言的结构和语义信息，然后可以用于生成文本摘要。它可以自动学习文本的特征，并根据给定的输入生成相应的摘要。T5生成的优点是适用于不同类型的文本和任务，并且不需要手动编写规则。然而，T5生成可能需要更多的计算资源和时间来进行模型训练和生成。
3. **TextRank**：TextRank是一种基于图的摘要生成算法。它通过将文本中的句子构建成图的形式，并使用图中的节点和边的权重来评估句子的重要性。TextRank根据句子之间的相似性和连接关系来计算句子的权重，并选择具有最高权重的句子作为摘要的一部分。TextRank的优点是简单有效，并且不需要额外的训练数据。但是，它可能无法捕捉到文本的语义信息，并且对于较长的文本可能存在信息损失。

TextRank算法本身并不直接控制生成的摘要长度，它仅根据句子之间的相似度构建图，并使用PageRank算法对句子进行排序。因此，生成的摘要长度取决于你从排序后的句子中选择的句子数量。

总体而言，不同的生成方法在实现和效果上有所区别。选择适当的方法取决于具体的需求、数据和任务要求。一般来说，基于预训练

T5-large的训练时间是Bart-large的两倍还多，大小也是bart的两倍

<https://zhuanlan.zhihu.com/p/420090646>

尽管都用的large模型，使用同样的max min，但T5生成的句子长度总比bart短，并且从内容质量上来说T5也更差，例如：

Since Dr Walkley's profile includes the name of Evangelia Antonopoulou, so T5 wrongly made Evangelia Antonopoulou as the subject of the summary. While Bart did not make this mistake.

（尽管我的原文本包含大小写，但T5生成的模型没有首字母大写，需要我手动将其转大写）

主题建模：

LDA主题分类-推荐算法

首选使用一种称为主题模型的无监督学习方法对导师进行分类。主题模型（例如Latent Dirichlet Allocation，简称LDA）是一种用于文本数据的常见方法，可以自动发现文档中的主题。

需要先使用CountVectorizer向量化，所以必须先删除主语，动词，连接词等会造成干扰的词语，尽量只保留关键字。

之后使用LDA算法总结出10-30个主题（我选择了25个），为每个导师打上他们对应主题的标签。之后将用户输入也进行LDA主题化，寻找到这25个主题中最匹配的导师并输出。

这个方法我使用了原文本和bart总结后的文本进行实验，我原一位bart总结后的文本去除了一些无用词，只保留有效信息，更适合LDA算法，但是由于篇幅较少，所以会导致每个主题的关键词偏少。

使用原文本的LDA算法推荐效果更好。这种方法与余弦相似度算法相比，只会推荐有限个数的导师，并且更加精准，但是如果用户输入的内容不符合总结的主题，那么将无法找到匹配的导师。

1. **LDA**：是基于概率的生成模型，它假设每个文档的主题是由Dirichlet分布产生的，而每个主题中的词是由多项式分布产生的。这种方法可以生成清晰的主题-词分布和文档-主题分布，但是可能在处理短文本或不均衡分布的数据时表现不佳。
2. **NMF**：是一种基于线性代数的因子分解方法，它将数据矩阵分解为两个非负矩阵的乘积，适合用于文本数据，因为文本数据通常可以被表示为非负的词频或TF-IDF值。NMF假设数据是多元高斯分布的，它可以发现隐藏的线性关系，因此可能更适合发现复杂的、跨多个文档的主题。

至于为什么NMF在某些情况下可能会比LDA更准确，这取决于数据的性质和具体的应用场景。NMF的一个优点是它能够更好地处理稀疏数据和噪声，因此对于一些文本数据集，NMF可能能够提供更清晰、更一致的主题。然而，这并不意味着NMF总是优于LDA，因为在某些情况下，LDA可能会提供更好的结果。

我的文本中稀疏数据和噪声很多，只能人工清理，但清理不完全，所以LDA的推荐效果更好。

BERTopic

**BERTopic**是一种用于主题建模和文档聚类的Python库。它结合了BERT编码器的能力和UMAP的降维能力，以及HDBSCAN的聚类能力，来创建和可视化文档的主题。

BERTopic的主要工作流程如下：

1. **文本嵌入**：首先，BERTopic使用Transformers库中的预训练BERT模型（或其他预训练模型，如RoBERTa、DistilBERT等）来将文本转化为向量（也就是嵌入）。
2. **降维**：然后，它使用UMAP（Uniform Manifold Approximation and Projection）算法进行降维。UMAP是一种用于降维的非线性技术，它在保留全局数据结构的同时，也能保留局部结构。
3. **聚类**：在降维后，它使用HDBSCAN（Hierarchical Density-Based Spatial Clustering of Applications with Noise）进行聚类。这是一种基于密度的聚类方法，不需要预先设定聚类数量，适合处理噪声和异常值。
4. **主题创建**：最后，BERTopic使用类内的词频（Class-based TF-IDF）来创建主题。

BERTTopic使用**min\_topic\_size**来控制每个主题必须包含的最小文档数。

但是我发现如果我不设置**min\_topic\_size**，他会使用默认值10，这样他只能给我的所有导师分成三类1,0和-1（无效类，噪声类），仅仅两个有效分类无法完成推荐任务。当我将**min\_topic\_size设置为2时，我会获得12个分类，但最终的推荐结果不尽人意。因为降低min\_topic\_size**，可能会导致过拟合，因为一些只包含极少数文档的主题可能只是噪声或者偶然的聚类，而并非真实的主题。

我掌握的导师文本数量总体来说太少，并且质量参差不齐，所以在这些文本下训练BERTTopic很难得到可用的分类结果。在huggingface上有一些已训练好的BERTTopic模型，其中包括对IMDB电影的分类，对学术论文内容的分类，他们都是经过上万篇文章训练而成，很遗憾，hugging face并没有对计算机领域的个人信息的分类模型。所以BERTTopic并不适用于我的推荐系统。