1. 内容交互部分，mashup特征，已选择的services，待测service的表示 交互；
2. 相似度计算不用HIN
3. 隐式交互部分 加入已选择的attention

**多通道？？一个attention模块，在多种空间（内容，历史信息）中都可以使用**

1. 显式交互部分 考虑已选择的，归一化。

1.K个相似度最大，具有代表性。

Learning to rank。

2.HIN的embedding。各种属性。各种连接边。（我们只有一种连接边）

深入地分析了人机交互的推荐问题。明确提出在选择一个服务时，应当考虑用户需求，该服务和用户已经选择过的服务的复杂交互。并且利用内容信息和历史调用信息，从多个角度，分析了它们的交互。

为了获取context\_services的有效表示，设计了一种基于attention机制的，整合多个service的表示方法，使得最终表示能够突出对选择行为影响较大的service，忽略无关service的影响。

可组合性？

实验：

**引言部分中可以考虑加入的一些点：**

用户在搭建一个mashup以满足一个复杂的功能需求时，会调用多个services。每个service可以实现特定的功能，相互之间存在着复杂的关系，比如互补（功能互异，联合起来满足复杂功能），替代（功能相同或相似）等等。例如，一个能够为旅客预定酒店的mashup，可能需要调用高德地图api，酒店详情查询api，支付宝支付api，微信支付api等插件。在根据酒店情况选择酒店之后，需要支付账单完成预定，所以酒店详情查询api和支付宝付款api功能上互为补充。而付款时可以选择微信和支付宝，所以这两个api互为替代，都可以单独完成付款需求。这些services组合起来，构成了mashup，可以满足用户的复杂需求。

在真实的服务推荐场景中，用户在与推荐系统交互的过程中，**动态**地完成mashup搭建。我们根据实际使用体验，归纳出一个真实且普遍的使用场景：当用户试图搭建一个新的mashup时，首先向系统中输入其功能需求。然后，系统根据其需求推荐一个services列表，接着该用户会从中选择一些services。然后，系统根据用户需求和已选择的services给出推荐列表，接着用户从中选择services继续完善mashup的搭建，这个步骤循环下去，直至用户需求被满足，mashup搭建完成。

我们在之前的研究中已经研究过，如何为尚未选择任何组件services的新用户进行推荐的冷启动问题。而本文将研究以下应用场景：在用户在线构建mashup的过程中，当他已经选择了几个services（referred as to context services）时，服务推荐系统如何继续为用户推荐services，帮助他完成mashup的搭建？~~需要注意的是，本文只关注用户已经选择了一部分services的情景，而不关注用户尚未选择任何services的场景。这是因为，后者实际上是前者的选择services的数目为0的一种特例，并且我们之前已经研究过后者【】。~~

在上例中，假如用户已经选择了高德地图api，那么一种可能的情况是，即便其他的地图api也能够满足用户关于地图的需求，系统也不需要再推荐这些api给该用户，因为高德地图api已经很好地满足了用户关于地图方面的需求，并且调用额外的替代品会产生额外费用。可见，用户选择的一个候选service的行为，不仅跟需求相关，而且在很大程度上受已经选择的services的影响。所以，在这种情境下，一个好的推荐系统除了考虑用户需求，也应当即时利用这些context services：通过对待搭建mashup，context services和candidate service之间的复杂关系正确地建模，使得candidate service能够连同context services一起，共同满足用户的复杂需求。

基于模型的协同过滤推荐系统通常提出一个模型并基于历史数据训练模型，最后使用该模型为存在历史行为的用户推荐item。一种流行的框架是，首先获取一个存在历史信息的user和item在同一个空间的特征表示，然后再基于它们的交互预测该user对item的评分，例如基于矩阵分解算法的模型，神经协同过滤模型等【MF,NCF】。但是在本文的在线会话式服务推荐场景中，这些方法~~在获取待搭建mashup的特征表示时~~遇到了一些困难。一方面，它们在模型训练阶段便得到了存在历史信息的mashup的表示。但是，对于模型应用阶段才出现的搭建mashup的需求来说，它们很难获得该mashup的特征表示。即便理论上可以基于用户在线动态选择的mashup的组件services对该mashup建模，但这种操作需要更新整个推荐模型，十分消耗资源和时间，这对于在线交互式推荐系统而言是不可忍受的。~~所以，在实际应用中，context services在这类推荐系统的在线预测阶段无法被利用，用户本质上仍然是一个没有调用历史的新用户。~~另一方面，遗憾的是，基于模型的协同过滤算法以及衍生的各种算法无法为没有特征表示的新mashup做推荐。所以，这类算法不能应用在在线会话场景中。

在基于模型的协同过滤算法无法应用的情况下，另一种普遍的方案是基于内容的方法。~~现有的基于内容的方法一般用在新用户的推荐场景中。在推荐服务时，~~它们仅仅考虑用户需求和candidate service之间的功能交互，倾向于推荐那些与需求的~~内容~~相似度比较高的services，也就是更可能满足需求的services【例如基于LDA的】。在本文的研究场景中，这些传统的基于内容的方法无法处理context services的信息，忽略了context services的在 在线会话式服务推荐中扮演的重要作用，因此也存在着一些缺陷。

另外，也有一类算法在用户输入功能需求之后，推荐一个完整的services set或者package，其中各个services之间是功能互补的【】。这种算法虽然考虑了services之间的关系，使得生成的services集合能够尽可能满足用户的复杂需求，但是它假设了各个组件services的功能是互异互补的，显然这种假设不能对用户实际需要的services之间的复杂关系有效建模。而且，这种方法在用户输入需求之后直接生成一个services组合结果，不适用于本文的研究场景，无法根据用户动态选择的context services优化推荐结果。

问题形式化定义

服务推荐系统的仓库可以表示为,其中M是mashup的集合，S是service的集合。Mashup是由多个service构成的，它们之间存在着调用关系。我们可以把这种关系当做隐式反馈数据，把历史调用信息转化为一个mashup-service调用矩阵MS，其中第m行第s列的取值为：。每个mashup或service都存在能够描述其功能的内容信息，包括词序列形式的文本和离散的标签，即content=（text，tag）。

对于上述服务推荐系统，给定一个搭建新的mashup需求MReq（也就是描述用户的功能需求的text和tag），和用户在线实时选择的组件服务 context\_services，怎么从用户未选择的服务集合（S- context\_services）中生成一个服务列表SL?

在大规模topN推荐系统中，一般主要分为两个环节：matching stage，生成跟用户相关的候选item列表；生成推荐列表的ranking stage，它又分为预测用户对每个item的评分或是选择概率，然后选择出评分最高的N个item（DIN）。

本文的研究重点是第二个环节的第一个阶段，这也是整个推荐过程的难点。我们的目标是生成一个概率预测模型，它能够预测在用户根据***MReq***已经选择***context\_services***的条件下，调用一个候选服务***s***的可能性，即。

问题分析&模型概况（根据启发设计模型）

我们认为，***MReq，context\_services***和s之间存在着复杂的交互关系，它们影响或决定了用户最终的选择行为。因此，本文中，我们设计了一种基于多种交互的深度学习模型，预测用户选择一个候选服务的概率。如图所示，我们的模型可以分为上下两层。下层由内容交互部分和近邻交互部分和组合交互部分组成，它们分别基于内容信息和历史调用信息，学习***MReq***, ***context\_services***和***s***之间的多种交互向量。最后，这些交互向量被输入到上层的多层感知机（MLP）中，输出***MReq***在***context\_services***的条件下，调用***s***的概率。

从功能的角度出发，用户调用多个services以搭建mashup的原因是每个service可以实现特定的功能，组合它们可以满足用户复杂的功能需求。所以，我们在选择一个services时，不但要考虑到需求和待测service的功能，还要考虑到已选择的services已经实现了哪些功能。换句话说，我们需要分析***MReq***，***context\_services***和***s***之间的功能交互。这也是我们设计内容交互部分的原因。具体的，我们首先基于特征提取器从***MReq***，***s和context\_services***中每个服务的内容信息中提取它们各自的特征，然后使用注意力模块整合***context\_services***中每个服务的特征，得到***context\_services***的有效表示。接下来我们拼接***MReq***，***s和context\_services***的特征表示，再把它们放入到一个MLP中，以充分学习它们复杂的功能交互，最终得到一个内容交互向量。

我们借鉴了user-based的协同过滤的思想设计了近邻交互部分。user-based的协同过滤算法中，利用相似用户对一个item的评分计算该用户对目标item的评分。它的假设是相似用户对相同item的偏好相同。受此启发，我们可以认为，如果两个mashup*的功能很*相似，那么在已经调用了相同的services(即相同的***context\_services***)的情况下，它们接下来对相同的候选service ***s***的评分行为也比较相似，也就是说，它们可能***context\_services***与和***s***之间的交互比较相似。所以我们可以基于历史调用信息，借助近邻mashup ***NM***（仓库中与***MReq***具有相似功能需求的mashup），***context\_services***和***s***之间的交互，推测出***MReq, context\_services***和***s***之间的交互。这部分中，首先将MS转化为图，然后使用了一种graph embedding的方法——node2vec去处理它，得到M和S中每个对象/元素的向量表示。然后，我们一方面基于***NM中***的表示以及它们与***MReq***的相似度，得到***MReq***的加权表示；另一方面使用注意力模块整合每个已选择的service的表示得到***context\_services***的表示。然后，最后，我们拼接***MReq***，***context\_services***和***s***的同一个空间的向量表示，使用一个MLP学习它们之间的复杂交互。

组合交互部分中，我们考虑了待测service与context services之间的可组合性对用户选择行为的影响。我们认为，如果在***NM***的组件services中，有很多存在于context services中的service与某个待测service一起出现过，说明它们之间的可组合性比较强，这种交互关系也会促进对这个待测service的选择行为。受此启发，我们首先根据近邻mashup中是否同时调用过context services中组件service和s的情况，构造了一个能够衡量context services和s的可组合性的稀疏向量，然后使用一个MLP处理它，学到一个向量来刻画 context services和s在组合性方面的交互情况。

需要注意的是，在内容交互和近邻交互部分中，我们使用特征提取器和node2vec得到的都是单个的已选择的services的向量表示。在每部分中，为了学习***MReq***，***s和context\_services***的交互，一种直观的思路是使用多个MLP分别学习 ***MReq***，***s***和每个已选择过的service的交互情况，但是这种方法需要消耗大量的计算资源，复杂度较高。所以我们首先使用一个向量表示***context\_services，***然后只需要使用一个***MLP***就可以学习它们的交互。具体地，在计算它们的整体表示时，我们设计了一个基于attention的模块，**我们设计了一种基于attention机制的方法，有效整合每个已选择service的特征表示，重点关注那些对下个service的选择影响比较大的services**，忽略无关service。

下面，我们先介绍每个部分，最后再详细介绍我们设计的用于整合单个已选择service表示的注意力模块。

3.2 内容交互

***MReq***，***context\_services***和***s***在功能上存在着复杂的关系。一方面，***s***在功能上跟***context\_services***关系复杂，可能是相互补充，或者相互替代等等。而另一方面，***context\_services***和***s***的功能组合起来，共同满足用户的复杂需求***MReq***。在考虑一个候选时，用户会考虑s和MReq和context\_services之间的功能交互。所以，在CI部分，我们从功能的角度，对***MReq***，***context\_services***和***s***的交互建模。

这部分可以分为两个步骤。第一步，特征提取。我们从***MReq***，***context\_services***和***s***的内容信息中提取特征向量。具体的，我们首先使用【】提出的深度学习技术，提取***MReq，s，***以及已选择的每个service的特征向量，然后基于注意力模块整合得到***context\_services***的表示。第二步，对它们的交互建模。我们没有基于一种先验假设去预定义他们之间的交互，而是基于数据训练一个MLP，对他们的复杂交互建模。

3.2.1特征提取

服务仓库中，mashup或service的功能描述，也就是内容信息，主要可以分为两种类型：word sequence（比如描述） and separate word set（比如tags）.在使用一个向量表示一个mashup或service时，由于这两种内容信息的形式不同，所以我们首先使用两种方法分别处理它们。**最后，我们将提取到的这两种特征拼接，作为描述一个mashup或service的功能的特征表示。**

Before extracting features by deep learning techniques, we need to use a dense vector to represent each term that appears in the content information of existing mashups and services. To this end, we first transform these terms into sparse binary vectors with one-hot encoding, e.g., [0, 0, …, 1, …, 0]. Next, we feed the vectors into an embedding layer and map each term to a dense vector or an embedding. More specifically, the embedding layer can be viewed as a lookup table, and the embedding of a term is indeed its corresponding weights in the embedding layer.在得到每个term的embedding vector之后，我们可以将包含多个词的内容信息（word sequence and separate word set形式），转化为矩阵：

(1)

对于word sequence形式的内容信息，我们使用了我们在【】中提出的text\_inception的方法来提取它的特征。受inception结构的启发，在text\_inception中，我们首先使用并行堆叠的卷积操作提取词序列中的局部特征，提高了效率和非线性；然后使用A global average pooling (GAP) layer突出有效特征，最后使用一个MLP实现非线性转换。细节可查阅文献【】。这个过程可以表示为：

对于separate word set形式的内容信息来说，term相互之间没有顺序，所以我们不能使用 用于处理词序列的text inception处理它。我们注意到，这些term一般是用户可读的词汇，一般存在预训练的，具有可加性的word embedding表示，例如glove，word2vec模型。因此，为了得到separate word set的定长的feature vector ，我们检索并平均每个词汇的embedding：

(8)

where is the embedding of the -th term in the set and is the size of the set.

对于一个mashup或者service，我们首先分别从它的word sequence and separate word set形式的内容信息中提取它的特征，然后拼接这两种特征，得到一个能够表示它的功能的特征向量：

(9)

where denotes the concatenation operation.

至此，我们可以将***MReq***，***s***和已选择的每个service的自然语言形式的内容信息表示为实值的特征向量：,，（）。

我们已经分析过，为了高效地学习***MReq***，***s和context\_services***之间的功能交互，我们首先需要整合各个已选择services的特征向量列表，使用一个特征向量表示***context\_services***的内容信息。我们设计了一个注意力模块完成这项工作。它根据已选择service c对s的影响程度计算一个权重，最后对各个service的特征加权求和。这个权重系数由c和s共同决定，当c对s的影响越大时，它的权重越大。我们在3.X部分详细系数这个过程。这里我们暂时先把它简单记做：

其中，是注意力模块的简称，X是用户已选择的services的数目。是service 相对于s的影响权重。

3.2.2 内容特征交互

接下来，我们对***MReq***，***context\_services***和***s***在功能上的交互关系建模。由于它们之间关系复杂，贸然假设和定义很可能不贴合实际情况，而DNN在理论上可以接近一切函数【】，所以我们使用一个MLP处理***MReq***，***context\_services***和***s***的功能特征的拼接结果，得到一个体现它们交互的低维向量。这个过程可以描述为：

(10)

3.3近邻交互

除了内容信息，服务推荐系统中还存在着历史调用信息，这也是一种能够有效提高推荐效果的信息。运用历史调用信息的一种常用的框架基于这种信息得到已存在的mashup和service在同一个空间的特征表示，然后再基于它们的交互，预测这个mashup对这个service的评分【MF,NCF】。~~首次启发，在会话式服务推荐场景中，我们可以考虑context services的影响，基于mashup，已选择services和待测service的交互做出预测。~~但是在本文的会话式在线推荐情景中，这种方法遇到了一个关键的困难： 它们很难即时处理用户实时选择的services（***context\_services***），也就无法得到当前待搭建mashup的特征表示，也便无法直接学习MReq与s在这个特征空间的交互情况。

user-based的协同过滤算法本质上基于近似用户与item的交互预测目标用户与item的交互。受此启发，我们打算借助近邻mashup ***NM***，***context\_services***和***s***之间的交互，推测出***MReq, context\_services***和***s***之间的交互。针对上述问题，一种直观的想法是，先使用多个MLP学习***MReq***的每个近邻mashup与***context\_services***和***s***在同一个特征空间（它们的特征表示都可以基于历史信息得到）的交互，再使用一个MLP推测出***MReq***与***context\_services***和***s***在这个特征空间的交互情况。但是这种方法消耗的资源太多，复杂度太高。所以我们采取了另一种替代的策略：我们首先基于近邻mashup的特征表示以及它们与***MReq***的相似度，加权得到***MReq***的特征表示，然后只需要使用一个MLP就可以学习待搭建***MReq***与***context\_services***和***s***在这个特征空间的交互。这种方式在本质上和基于attention机制整合每个已选择service的特征的方法有共通之处。总结一下，近邻交互部分的主要步骤（关键技术）跟内容交互类似，主要分为特征获取和特征交互。*~~首先基于历史调用信息得到仓库中的mashup和service在同一个空间的特征表示；然后一方面衡量~~****~~MReq~~****~~和已有的mashup的相似度，并基于这种相似度为它寻找近邻mashup~~* ***~~NM~~****~~，再基于~~****~~NM~~****~~的特征得到~~****~~MReq~~****~~的特征；另一方面，使用attention模块整合各个已选择的特征得到context services的特征表示。特征交互则使用MLP学习待搭建mashup与~~****~~context\_services~~****~~和~~****~~s~~****~~在这个特征空间的交互。~~*下面我们分别阐述这些步骤。

特征获取环节，我们需要获取***MReq***与***context\_services***和***s***特征表示。我们首先将历史调用关系矩阵MS转化为图，然后使用node2vec处理它，进而可以得到仓库中存在的mashup和services（包括s）的特征表示。相比于传统的矩阵分解的方法，node2vec能够捕捉更复杂的非线性关系。

我们基于近邻mashup的特征表示计算***MReq***的特征表示。其中一个关键问题是如何衡量一个mashup与***MReq***的功能相似度。具体的，我们首先计算它们的两种形式（词序列和词集合）的内容信息的特征的相似度，然后整合它们。

Since the features of the word sequence of and are real-valued vectors, we employ the commonly-used Cosine similarity to calculate their similarity, i.e., .

|  |  |  |
| --- | --- | --- |
|  |  | (10) |

The similarity between the separate word set of and that of , , is computed in the same way. The weighted sum of the two similarities is regarded as the content similarity between and . In this study, and are not set to fixed values, and they act as learnable parameters in the process of model training.

|  |  |
| --- | --- |
|  | (11) |

然后，We select most similar mashups of to build its neighbor mashups according to their content similarities.

最后, we calculate the weighted representation of mashup using the following equation:

|  |  |  |
| --- | --- | --- |
|  |  | (12) |

where is a neighbor mashup of , is the content similarity between and , and is the representation of obtained by node2vec.

在获取context\_services的特征表示的时候，像我们使用attention模块整合各个services的特征表示。这个过程可以简单表示为：，具体细节我们在部分再讲。

3.3.4近邻交互

至此，我们得到了***MReq***，***context\_services***和***s***在同一个特征空间的表示，我们把它们拼接，使用一个MLP学习它们在这个特征空间的交互：

(17)

where denotes the representation of obtained by node2vec.

组合交互

隐式交互组件中，我们基于mashup，已选择service和待测service在同一空间中的表示，学习它们的隐式交互。除了这种策略，我们还从近邻mashup中的历史信息中，该用户已选择的services，与待测services之间的更直接的历史交互中，分析它们的可组合性，进而预测该用户在已经选择这些services的情况下，继续选择该待测service的可能性。

我们首先根据近邻mashup与待测service的调用关系构建了一个能够反映用户已选择的services与待测service之间的可组合性的稀疏向量。该向量的具体情况如下列公式：

(14)

(14)

where is an value that measures每个近邻mashup中，与s共同出现的context services占该mashup的所有组件services的比例，这个比例可以衡量s和context services的可组合性：在某个近邻mashup中，越多的context services中的service与s一起出现，意味着context services和s的可组合性就越强，也意味着用户在当前的context services环境下，越可能调用s。

Since the number of neighbor mashups is , the dimension number of is also .

然后，我们将上述 从所有近邻mashup的调用历史中获取的，表示当前的context services和s的组合关系的向量，输入到一个MLP中。这样，我们就可以学习到从可组合关系角度考虑的，context services和s的交互向量。这个过程可以表示为：

(15)

**Attention块**

我们已经分析过，已选择的services在推荐过程中扮演着重要的作用。我们需要对用户需求，待测service，已选择services的交互关系建模。并且，在我们的框架中，我们使用了一个MLP处理它们的向量形式的特征表示建模。并且在此之前，我们已经获取了需求，待测service和每个已选择service的表示。所以，问题的关键在于，我们如何整合已经提取到的每个已选择的service的表示（比如基于内容信息得到的表示**，基于历史信息得到的表示**），得到一个特征向量表示以表示***context\_services？***

~~基于上述内容特征提取方法，我们可以分别提取内容特征。~~

~~接下来我们需要整合这些已选择services的特征向量列表，使用一个特征向量表示以表示~~***~~context\_services~~***~~的内容信息。~~

一种直观的方案是直接拼接所有的**。但是，这种方案存在着一些问题。context中已选择的services的数目是不确定的。为了得到定长的context 特征表示，需要使用padding等技术。但是无论如何处理，拼接处理会增加特征表示的维度，增加模型的参数量，降低模型效率。另外，在context中选择的services之间实际上是无序的，也就是说，如果用户改变选择services的顺序， context的表示应当是相同的。但是使用这种方法时，改变顺序则会产生不同的特征表示结果，一方面不合理，另一方面也增加模型学习的难度【】。**

**另外一种流行的解决方案是使用pooling方法处理每个service的表示，最终得到定长的特征表示【**Deep neural networks for youtube recommendations/Wide & deep learning for recommender systems**】。**

**其中**是已选择的第i个services的向量表示,X是已选择的services的数目。

最常用的pooling方法是sum和平均池化，它们分别对已选择services的特征列表进行元素求和/平均操作。

这种方法能够给解决拼接方法的遇到的两个问题，但是仍然存在着一些问题。我们来分析一个案例。假设用户需要开发一个能够帮助游客做旅行准备的mashup。假设该用户已经选择了酒店预定api，天气查询，地图，支付宝付款api，~~那么在评价一个微信支付api时。~~一种可能的情况是，考虑到预定酒店之后，一般需要在线支付，并且用户已经选择了同类型的在线支付api——支付宝api，所以用户可能会选择微信支付api。这个选择过程中，对用户的选择行为起到主导作用的实际只有预定酒店api和支付宝付款api，而天气查询api的影响不大。换句话说，下个services的选择，受到每个已选择的services的影响程度是不同的。所以计算context的特征表示时，每个已选择services的权重大小不同。然而，使用pooling方法实际上是按照同样的权重整合所有选择的services的特征表示。 另一方面， 假如该用户在相同的context services的条件下衡量一个旅游线路规划的api，此时context的特征表示应当和上面衡量微信支付api时的有区别：后者应当突出已选择的services中跟支付有关的特征，而前者则应当重点关注跟出行相关的特征，比如需要考虑到地点和天气等因素。而在pooling方法中，用户在选择下一个services时，无论待测的service是什么，context services的特征表示是固定的，实际上这种固定维度的表示很难有效地表示出已经选择过的services中中的丰富特征，比如它们已经实现的复杂功能。

*用户的选择行为受相关api的影响比较大，而几乎不受无关api的影响。*

*我们只需要关注跟这个待测service相关的一些已选择services，而忽略无关的services，这样，可以更有效地表示出已选择services实现的功能。*

**受此启发**，~~在借助用户已选择的services信息，衡量一个待测item是否满足用户需要时，~~我们设计了一个基于attention的整合已选择services的表示的方法。**通过注意力机制，我们重点**关注那些跟待测service相关的（功能相似，可以相互替代；或者功能相异，可以互为补充）**，对选择过程影响比较大的**services~~实现的功能~~，**而忽略无关的service~~特征~~**。与pooling的方法不同，我们的方法根据已选择service和待测service的相关程度adaptively calculate the representation vector。所以我们的方法中，不同services的对context services的特征表示的贡献权重不同，同时计算的context的特征表示随着待测item变化而变化。

整体上，我们使用各个已选择service的特征向量的加权和，去表示context services。

其中 在形式上是一个已经选择的service i针对待测service s的权重，它的物理意义是service i与待测service s的相关程度，也可以理解为service i对用户选择的决策行为的影响程度，对待测service被选的贡献程度。

可以表示service i，service s的相关程度的权重，，是由service i，service s的特征向量共同决定的。我们认为，两个特征向量的元素乘的结果，一定程度上可以衡量它们的相似度；而两个特征向量的元素减的结果，一定程度上可以衡量它们的差异（有时可以理解为功能上的互补程度）。这两个运算结果是可以帮助i和s的相关程度建模的先验知识。考虑到i影响选择s的行为的方式或者原因不确定，比如，可能是因为s和i之间功能互补，也可能是因为它们的功能相似，我们把这两种运算结果，与service i，service s的特征向量拼接后，一共输入到MLP中，自动学习i与s的相关程度，进而得到一个相关性score。~~所以首先我们基于一个MLP处理这两个向量以及对它们的数值操作结果。~~这个过程可以表示为：

其中， 是两个特征向量的元素乘的结果；而是两个特征向量的元素减的结果。

最后，我们再将这个score输入到一个softmax层中计算得到一个最终权重标量：