针对数百万用户的基于嵌入的新闻推荐总结

本文尝试对Yahoo Japan团队等提出的Embedding-based News Recommendation for Millions of Users的方法做一个总结，主要包括以下内容：

1、任务目标和任务难点

2、传统推荐方法简介及为什么不适用于新闻推荐

3、新闻推荐方法的基本思路

4、word-based方法

5、Embedding-based方法

6、试验结果

7、总结

一、任务目标，任务特殊性

1、任务目标

针对广大的网络用户提供个性化的新闻推荐服务

2、任务的特殊性 ：

（1）、新闻有很强的时效性，因此旧的文章会在很短的时间内被新的文章所替代

（2）、系统要在用户访问的数百毫秒的时间内做出响应。

二、传统推荐方法简介及为什么不适用于新闻推荐

1、传统方法简介

经典的推荐系统的方法包括基于隐含特征的矩阵分解技术（SVD）的方法和基于相似度的协同过滤的方法。

基于隐含特征的矩阵分解方法 ：

基于隐含特征的矩阵分解的方法的基本精神是用户和商品都包含某些内在的属性（这些内在属性就是隐含特征），当用户与商品的内在属性契合度比较高时，那么用户可能就会喜欢该商品。

具体操作就是对用户-商品 的评分矩阵进行矩阵分解（SVD），分别获得用户和商品在属性空间的表示，然后在该属性空间内计算相似度，然后将与用户相似度高的商品推荐给用户。

但是直接进行SVD分解有两个主要问题：

1、用户-商品矩阵中可能很大，进行SVD分解的计算复杂度非常高

2、用户-商品矩阵中存在大量零值（这并不代表用户对该商品兴趣很低，可能只是没有看过，直接SVD分解，不可靠）

实践上采用梯度下降法最优化以下目标函数：

$L = \sum_{i,j}(r^i r^j + b_i + b_j - n_{ij})^2$(其中$r^i$代表第i个用户在属性空间的表示，$r^j$为商品在属性空间的表示，$n_{ij}$表示用户-商品矩阵中第i行第j列的值，$b_i$，$b_j$代表偏置项)

基于相似度的协同过滤方法 可以分为基于用户的协同过滤方法和基于物品的协同过滤的方法。

基于用户的协同过滤方法 ：

基本精神：给用户推荐和他兴趣相似的其他用户喜欢的物品

根据用户-商品矩阵，计算目标用户与其他用户之间的相似度（两个行向量的余弦相似度），并进行归一化操作，将归一化的相似度作为权重，针对目标用户没有看过的物品，计算其他用户对目标物品的评分的加权平均和作为目标用户的对目标商品的评分，推荐评分高的物品给该用户。

基于物品的协同过滤 ：给用户推荐和他之前喜欢的物品相似的物品（与基于用户的协同过滤方法精神基本一致，一般在用户多，而物品相对少时使用）

2 、 为什么传统推荐方法不适用于新闻推荐

基于隐含特征的矩阵分解的方法在获得用户和商品在属性空间的表示后，推荐速度非常快（只需要执行类似内积的相似度计算，然后排名即可），但是要获得属性空间的表示，需要进行SVD分解或梯度下降，由于新闻的时效性，旧的文章会在很短的时间内被新的文章所替换，因此，无法构建稳定的用户-商品矩阵，也无法获得用户对商品的准确评分，而且SVD计算复杂度高，因此基于隐含特征的矩阵分解方法不适用于新闻推荐。

基于相似度的协同过滤的方法与基于矩阵分解方法面临的问题相似，1、由于新闻文章很强的时效性，无法构建稳定的用户-商品矩阵，也无法获得用户对商品的准确评分，2、需要计算大量的相似度，计算成本高，无法实现快速的响应。因此也不适用于新闻推荐。

三、新闻推荐方法的基本思路

虽然矩阵分解的方法不适用于新闻推荐，但是只要获得用户和商品在属性空间的表示后，就可以非常快速实现推荐。因此，如果我们能够快速的获得针对新闻推荐的用户和文章在属性空间的表示（即分布式表示），就可以实现快速的新闻推荐。因此，新闻推荐的基本思路如下：

1、根据文章的内容得到文章在属性空间的编码表示

2、根据用户的访问历史获得用户偏好在属性空间的编码表示

3、根据文章和用户偏好的编码表示，计算相似度得分，选出得分较高的文章（由于用户请求和显示需要在数百毫秒之内完成，相似度的计算必须是轻量级的）

4、根据某些特征(比如预期的页面浏览次数和每篇文章的新鲜度)等，重新对选出的文章进行排序

5、根据文章之间的相似度，删除排名靠前但内容重复的文章，这一步非常重要，因为相似的文章往往具有相似的评分，如果不进行取重，那么推荐的文章的多样性就会减少，用户的满意度会下降。

具体删除步骤如下：对排序后的文章，从第二篇开始，贪心的计算与其前边所有文章的相似度（采用余弦相似度$cos(A,B) = \frac{AB}{|A||B|}$）,如果相似度超过某一设定的门限值，就进行删除。

接下来首先介绍一种word-based的baseline方法，说明其存在的问题。然后说明本文详细介绍提出 Embedding-based的方法。后两步操作对于每种方法都相同，因此对于不同的方法主要介绍前三步，即获得文章和用户偏好在属性空间中的编码以及相似度的计算，分别使用a和u_t来表示文章和用户偏好在属性空间编码（用户在不同时间段，看作某些文章后，偏好会发生改变，因此u_t是访问历史的函数，即：$u_t = F(a_1^u, a_2^u \dots a_t^u)$ ，a_i^t为用户u访问

的第i篇文章），使用$R(u_t, a)$表示用户偏好和文章的相似度

4、word-based 方法

假设总共有V个词。

（1）文章编码 ：采用词集模型表示，使用文章中包含的单词的集合来表示文章，即如果文章中存在某个词，那么词集模型对应的项就为1。具体如下

$a \in (0,1)^V$ ，$(a)_v = \begin{cases} 1, 文章中包含v \\ 0, 其他情况 \end{cases}$

a是一个V维的二值向量（每一项取0或1），如果文章中包含某个词，则该维为1，否则为0。

（2）用户偏好编码 ：同样采用词集模型，使用浏览历史的所有文章中单词的集合表示用户偏好，如果浏览历史的文章中包含某一词，则在向量中用1表示，具体如下：

$u_t \in (0,1)^V$，$(u_t)_v = (F(a_1^u, \dots a_t^u))_v max_{1 \le t' \le t}(a_{t'}^u)_v$

（3）相似度计算 :文章和用户偏好之间的相似度采用简单的内积的方式得到，即：
$R(u_t, a) = u_t^T a = \sum_{v \in V}(u_t)_v (a)_v$

该方法没有需要学习的参数，只需要简单的统计词集和计算内积即可，因此，非常高效。

但上述方法有两个主要的问题：

1、**稀疏表示的问题** ：词集模型无法衡量词之间的相似性，无法考虑语义信息，只能实现字面的对比，只有文章和浏览历史中存在相同的词时，相似度才会增加，因此无法很好的处理同义词和拼写错误等问题。

2、**intensity** ：将浏览历史记录看作一个词集，因此关于浏览历史的顺序和频率信息将丢失。而且非常容易受到混入浏览历史记录中的噪音的影响，鲁棒性较差，实践上计算相似度和删除重复项都表现不是很理想。

5、Embedding-based方法

针对word-based方法存在的问题，Embedding-based方法做了两个主要的改进：

1、使用一个密集向量对文章和用户偏好进行编码，可以更好的捕获相似性（并且保证可以迅速计算得到）

2、使用循环神经网络，对用户的浏览历史进行编码（用户相对比较固定，因此可以将用户的状态存储，在模型不变的情况下，在浏览完某篇文章后，执行一步状态更新即可）。

如果用户是一个新用户，那么就要面对冷启动的问题，原文中没有讲到这个，这里简要说明一下：基本精神就是**通过不同的维度获取用户的基本特征，从而进行粗粒度的推荐** ，主要有以下几个思路：

1、**利用用户在其他地方已经沉淀的数据进行冷启动** ：在腾讯等大公司，有许多相关的产品，因此可以打通各个产品的日志系统，提取用户特征，从而实现比较好的冷启动。

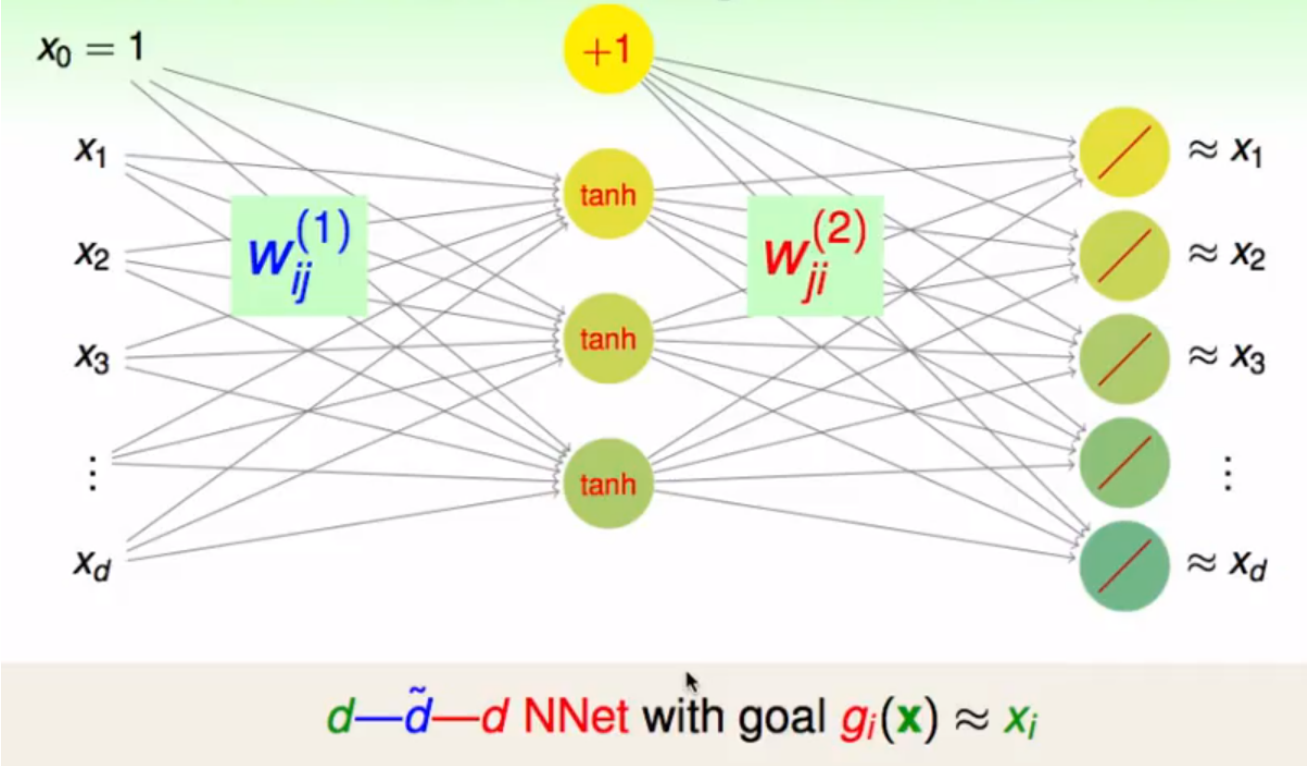
2、**利用用户的手机等兴趣偏好进行冷启动** ：安卓系统比较开放，比如可以通过分析用户安装的应用，了解用户的兴趣。

3、**制造选项** ，让用户选择自己感兴趣的点后，及时生成粗粒度的推荐

文章相对独立，可以进行独立编码，为了保证能够迅速计算得到，在模型训练好后进行编码时，不需要其他文章。原文采用的方法时一个降噪自编码器的变体。为了介绍该模型中采用的方法，我们首先介绍传统的自动编码器，然后介绍降噪自动编码器，最后介绍本文中介绍的降噪自动编码器的变体。

传统自动编码器 ：

自动编码器最早被提出是用于解决深度神经网络的权值初始化问题，深度学习是一种表示学习方法，每一层的权重代表了某种特征转换，自动编码器的基本精神是：在训练之前，我们不知道输入中那些特征对于目标是有意义的，因此，我们希望能够保留输入中的所有信息，因此，好的权重可以很好保存输入的原始信息（information preserving encoding），可以在下一层中对原始数据进行重构。



自动编码器的基本架构如上：输入与输出都是一个d维向量，中间层是一个d'维的向量。具体公式如下：

$h = s(Wx + b)$,其中W是一个$d' * d$的矩阵

$y = s(W'h + b')$

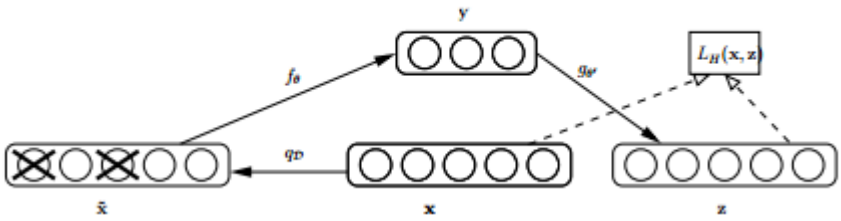
为了控制模型复杂度，一般要求$W' = W^T$

由于我们的目标是使得x与y尽可能接近，因此最直觉的损失函数为均方误差$L(x, y) = |x - y|^2$

降噪自动编码器:

降噪自动编码器是传统的自动编码器的一个拓展，其动机来源于人类能够从部分观察中恢复全部的信息，比如我们能够识别部分遮挡或损坏的图像，另外我们能够组织形成与多种形式信息（如图像和声音等）高级的概念，并且能在某些模式丢失的情况下也能记住它。降噪自动编码器的精神是**利用部分破坏的输入信息也能产生几乎相同的表示**，从而提升自动编码器的鲁棒性。

降噪自动编码器一个非常简单的方法就是在训练时，向输入数据中加入一些高斯噪音。Bengio[2]等认为添加噪音的训练等价于正则化，相当于对函数进行了平滑，不会获得显著的性能上的显著的提升。Bengio[2]提出的方法是从一个破坏的，部分的输入中重构原来的信息，**可以使得性能获得非常显著的提升**。结构如下图所示：



首先根据输入x进行一个部分的破坏得到$\tilde{\boldsymbol{x}}$,$\tilde{\boldsymbol{x}}$ ~ $q_D(\tilde{\boldsymbol{x}}|\boldsymbol{x})$,具体操作如下：对于每一个输入x，根据跟定的超参数$\boldsymbol{p}$，使得x中的pd维的项为0，其余项保持不变。

然后将$\tilde{\boldsymbol{x}}$接入原来的自动编码器中,具体的操作如下：

$$\boldsymbol{h} = \boldsymbol{s}(\boldsymbol{W}\tilde{\boldsymbol{x}} + \boldsymbol{b})$$

$$\boldsymbol{y} = \boldsymbol{s}(\boldsymbol{W}'\boldsymbol{h} + \boldsymbol{b}')$$

Bengio[2]等从流型学习的角度给出了降噪自动编码器的解释，如下图：

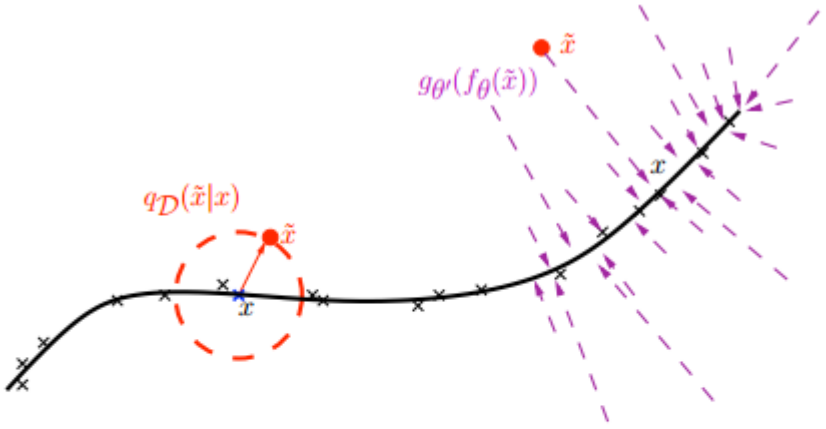


Figure 2. **Manifold learning perspective.** Suppose training data (×) concentrate near a low-dimensional manifold. Corrupted examples (•) obtained by applying corruption process $q_D(\tilde{X}|X)$ will lie farther from the manifold. The model learns with $p(X|\tilde{X})$ to "project them back" onto the manifold. Intermediate representation Y can be interpreted as a coordinate system for points on the manifold.

假设训练数据在x集中在低维流型区域附近，那么进行部分破坏得到的$\tilde{\boldsymbol{x}}$将远离流型区域。降噪自动编码器尝试根据$\tilde{\boldsymbol{x}}$重构x，因此相当于在流型区域内从远离流型区域的点向流型区域进行转换。因此去噪自动编码器可以看作是一种流型学习的方式，因此，可以认为y捕捉了输入信息x的主要特征。

Bengio提出了另外一种损失函数：**重构交叉熵(reconstruction cross entropy)** ：

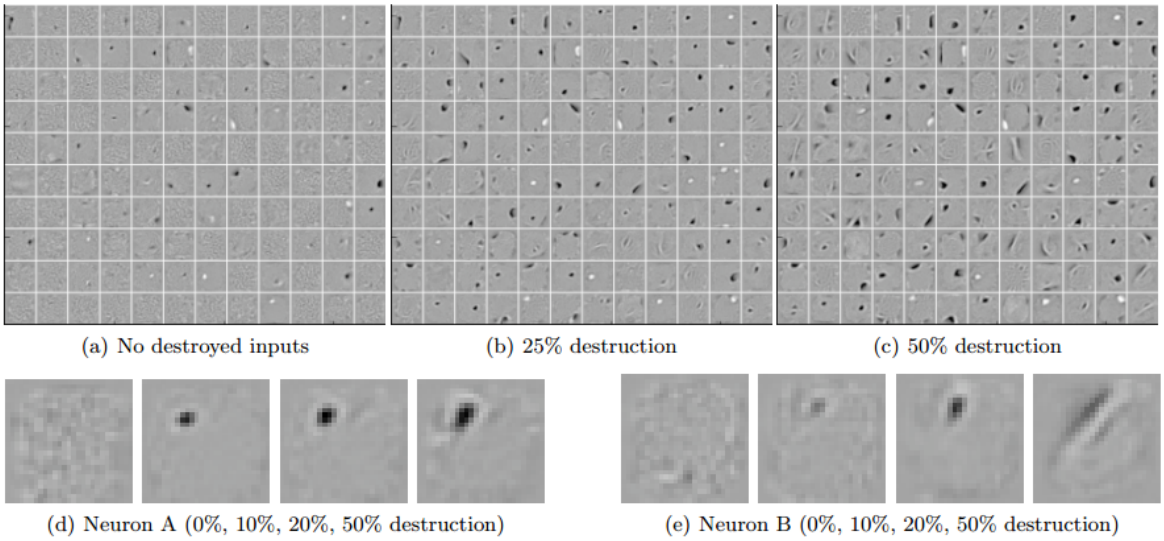
$$\boldsymbol{L_R}(\boldsymbol{x}, \boldsymbol{y}) = -\sum_{k=1}^{d}[\boldsymbol{x_k log y_k} + (1 - \boldsymbol{x_k})\boldsymbol{log}(1 - \boldsymbol{y_k})]$$

如果x是一个二值向量(binary vector，新闻推荐系统对于文章的表示训练时也采用了这种损失函数,输入x因为词集模型，可取的值为[0,1])，那么损失函数就为给定伯努利参数y条件下输入x的负的log-likelihood，其物理意义就是通过调整参数y，使得出现在训练集中x有最大的可能性 。

另外关于这个损失函数Bengio[2]从一个生成模型的角度推导了上述损失函数，并指出最大化上述目标等价于最大化一个特殊的生成模型的变分边界（variational bound），关于这个推导，由于时间关系，还没有认真去研究，因此，这里就不再说明。

由于$\tilde{\boldsymbol{x}}$ 是x随机部分失活得到的，y是有$\tilde{\boldsymbol{x}}$得到的，因此y不是x的确定性映射。因此自动编码器无法学到全等映射，因此我们可以不用限制$\tilde{\boldsymbol{d}} < \boldsymbol{d}$。

由于该方法试图从一个被破坏的输入学习一个很好的映射，因此这会使模型捕捉数据中不变的特性，因此可能捕捉的更有趣的信息，Bengio[2]等的实验中证明，在训练中加入破坏性的噪音时，权重可以捕捉更多合理的特征，而且，随着噪声水平的增加，能够捕捉到更大尺寸的结构。如下图所示。



(a) No destroyed inputs (b) 25% destruction (c) 50% destruction

(d) Neuron A (0%, 10%, 20%, 50% destruction) (e) Neuron B (0%, 10%, 20%, 50% destruction)

Figure 3. **Filters obtained after training the first denoising autoencoder.**
(a-c) show some of the filters obtained after training a denoising autoencoder on MNIST samples, with increasing destruction levels ν. The filters at the same position in the three images are related only by the fact that the autoencoders were started from the same random initialization point.
(d) and **(e)** zoom in on the filters obtained for two of the neurons, again for increasing destruction levels.
As can be seen, with no noise, many filters remain similarly uninteresting (undistinctive almost uniform grey patches). As we increase the noise level, denoising training forces the filters to differentiate more, and capture more distinctive features. Higher noise levels tend to induce less local filters, as expected. One can distinguish different kinds of filters, from local blob detectors, to stroke detectors, and some full character detectors at the higher noise levels.

降噪自编码器的变体（对输入文章内容的分布式表示）：

为了介绍新闻推荐算法中的降噪自编码器的变体，为了方便说明，这里对去噪自动编码器做一个简单的说明：

$\tilde{x} = q(\tilde{x}|x)$

$h = f(W\tilde{x} + b)$

$y = f(W'h + b')$

$\theta = argmin_{W,W',b,b'} \sum_{x \in X} L_R(y, x)$

其中$x \in X$是原始输入向量，$q(.|.)$是破坏分布。随机破坏向量$\tilde{x}$从$q(.|x)$中得到。隐藏层表示h，是$\tilde{x}$通过神经网络得到，参数为W，b。同理，表示向量y通过W'和b'计算得到，$L_R(.,.)$为上一节提到的重构交叉熵损失函数，通过学习这些参数来最小化y和x的重构误差。

不管是传统的自动编码器还是去噪自动编码器，每个样本都是独立训练，中间层的表示h仅与其对应的输入有关。但是根据之前的论述，我们希望中间层的表示h帮助我们衡量用户与文章匹配的相关性以便进行匹配，衡量不同文章之间的相似度以便进行删除操作。因此我们不仅希望h包含输入x的信息，我们还希望如果x_0与x_1相似，h_0与h_1的内积$h_0^T h_1$应该比较大。为了实现该目标，使用一个三元组(x_0, x_1, x_2)作为训练集的输入，通过改变目标函数来保护他们之间的相似性，如下图所示：

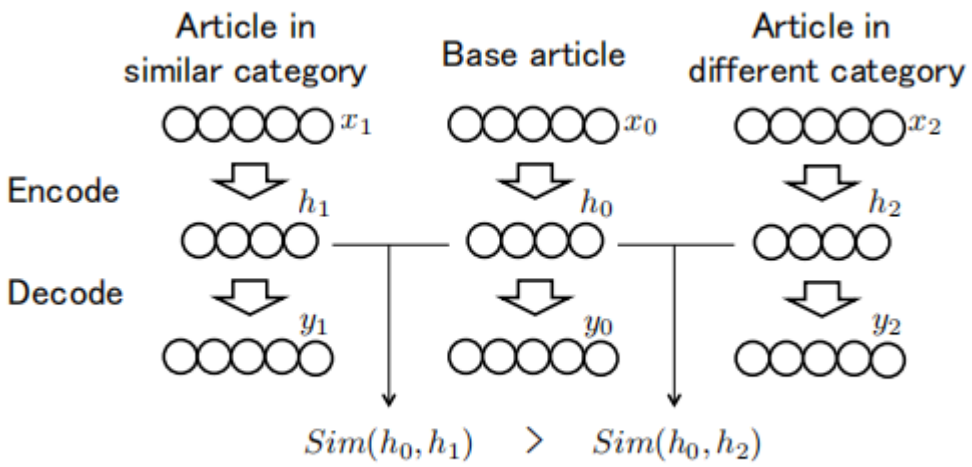


Figure 2: Encoder for triplets of articles

具体实现如下：

$\tilde{x}_n = q(\tilde{x}_n|x_n)$

其中x_n为对应文章的词集模型对应的向量（如果文章中存在某词，那么该维对应的值为1，否则为0）。

$h_n = f(W\tilde{x}_n + b) - f(b)$

保证$x = 0 \rightarrow h = 0$，意味着没有可用信息的文章与任何其他文章都不相似

$y_n = f(W'h_n + b')$

损失函数为 ：$L_{W,W',b,b'} = \sum_{x_0,x_1,x_2 \in T} \sum_{n=0}^{2} L_R(y_n, x_n) + \alpha L_T(h_0, h_1, h_2)$

其中$L_T(h_0, h_1, h_2) = log(1 + exp(h_0^T h_2 - h_o^T h_1))$

$L_R(x, y) = -\sum_{k=1}^{d}[x_n^k log y_n^k + (1 - x_n^k) log(1 - y_n^k)]$ 为重构交叉熵

$L_T(h_0, h_1, h_2)$ 的目标是为了保留文章之间的相似性。由于x_0与x_1更相似，因此也希望h_0与h_1的内积更大，即$h_0^T h_1 > h_o^T h_2$,下面我们来分析上述目标函数，当$h_0^T h_2 >> h_o^T h_1$时，L_T近似为$h_0^T h_2 - h_o^T h_1$,因此目标就是使得$h_0^T h_2 - h_o^T h_1$尽可能小，当$h_0^T h_2 << h_o^T h_1$时，L_T接近于0，很难再减小，已经达到目标，不再惩罚，重点是实现y是对x的重构。α是平衡两者之间的超参数。

因此，通过上述训练后，最终得到的h包含两个粒度的信息，1、输入文章x的信息 2、x与其他文章之间的相似度的信息。

激活函数为sigmoid函数，采用minibatch 的梯度下降法进行训练。

需要说明的是，由于在训练时，我们随机使x的pd维的为0，因此，与dropout类型，测试时，不会对输入x进行随机pd维的失活，因此，为了保证中间层在训练时有掩码噪声和测试时没有掩码噪声分布相同，需要乘上$1 - p$

使用上述方法生成的h表示文章有三个应用：

1、计算user-state。

2、衡量用户和文章的相关性

3、衡量文章之间的相似度，以便进行删除操作。

符号说明

上一节介绍了文章内容的表示方式，接下来说明对于用户偏好的分布式表示。为了下述表述方便，先对符号进行一些说明。

A：文章的集合 $a \in A$ 代表一篇文章

$\{a_t^u \in A\}_{t=1,...T_u}$ 表示使用者$u \in U$的浏览记录，浏览表示用户访问了文章所在网页的URL。

session表示用户访问了了推荐服务并点击推荐列表中的一篇文章。

当使用者u点击推荐服务中的一篇文章（发生某种情况）时，他/她将立即访问所点击的文章的网址（发生浏览）。因此在a_t^u和a_{t+1}^u之间只有一个会话。该会话被简称为s_t^u,然而使用者u可以不使用我们的服务访问该URL，因此s_t^u并不总是存在的。

会话与呈现给用户u的列表相对应，使用一个文章的列表$\{s_{t,p}^u\}_{p \in P}$来表示s_t^u

P是是本次会话屏幕上实际显示的推荐列表的位置集合。其中P_+表示被点击的位置，P_-表示没有点击的位置。

用u_t表示使用者依赖于$a_1^u, \dots a_t^u$的状态，代表了刚刚访问了a_t^u后的偏好。

$R(u_t, a)$代表用户状态u_t与文章a的相关性，代表了是时间t时使用者u对文章a的感兴趣程度。

下图说明了符号之间的关系。

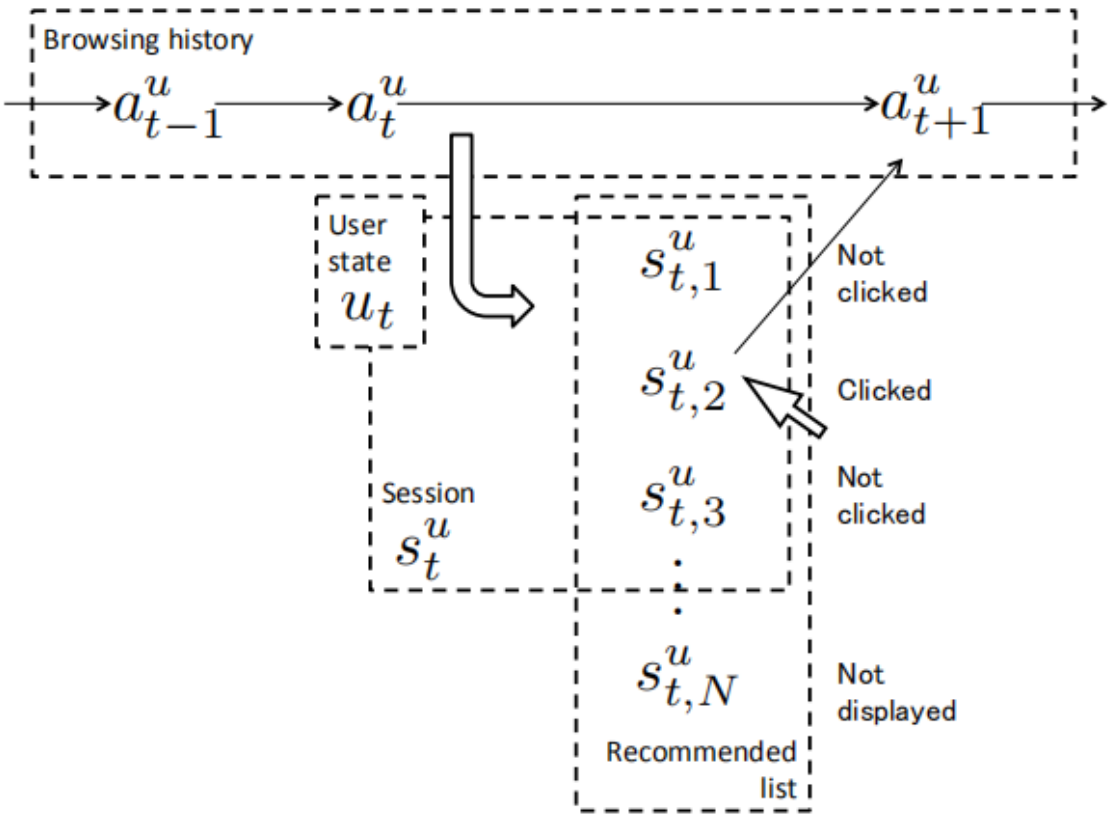


Figure 3: Browsing history and session

还需要确定使用者的状态函数$u_t = F(a_1^u, \dots a_t^u)$和相相似性数$R(u_t, a)$。

相似性函数 ：$R(u_t, a)$

由于在真实的大流量的新闻分发系统中响应时间可能非常短（需要保证在推荐服务列表显示之前的很短的时间内被计算出来），因此相似性函数$R(u_t, a)$必须是一个非常简单的函数，$R(u_t, a) = u_t^T a$（没有需要学习的参数）。

目标函数

$R(u_t, s^u_{t,p+})$表示用户状态与在会话中点击了的文章$s^u_{t,p+}$的相似度，因此要比用户在会话中没有点击的文章$s^u_{t,p_}$相似更高。因此我们希望在任何会话s^u_t中，对于任何的$p_+ \in P_+, p_ \in P_$ 都有：$R(u_t, s^u_{t,p_+}) > R(u_t, s^u_{t,p\_})$。因此，我们的目标可以描述为使$R(u_t, s^u_{t,p_+}) > R(u_t, s^u_{t,p_})$的概率尽可能的大。因此，我们的目标函数可以定义为：

$$\sum_{s^u_t} \sum_{p_+ \in P_+ p\_ \in P\_} = -\frac{log(\sigma(R(u_t, s^u_{t,p+}) - R(u_t, s_{t,p^u\_})))}{|P_+|P\_||}$$

其中sigmoid函数是为了表示概率，取log是为了避免概率乘积超出下限，分母项是实现归一化处理（总共有|P+|*P-||项的log 概率的叠加）

另外考虑到当文章垂直排列时，点击的概率与显示的位置有关，我们还需要添加一个偏置项$B(p_+, p_)$

因此，最终的目标函数为$\sum_{s^u_t} \sum_{p_+ \in P_+ p_ \in P_} = -\frac{log(\sigma(R(u_t, s^u_{t,p+}) - R(u_t, s_{t,p^u_}) + B(p_+, p_)))}{|P_+|P_||}$

其中B为一个需要学习的参数，原文中没有给出具体说明，这里也不在说明。

接下来唯一剩下的问题就是用户状态函数的定义

用户状态函数 ：$u_t = F(a^u_1, \ldots a^u_t)$

用户状态函数比较复杂，它是$a^u_1 \ldots a^u_t$的函数，原文中介绍了两类模型：衰减模型和循环模型，循环模型又可进一步分为基于RNN,基于LSTM和基于GRU的模型，他们都采用分布式表示的密集向量表示。

衰减模型 ：与基于词的模型不同，衰减模型的动机是通过加权平均来组织浏览历史，增加最近的浏览的权重，考虑了一定程度的浏览顺序。

数学表示为：$u_t = \alpha . \frac{1}{\sum_{1 \le t' \le t} \beta^{t-t'}} \sum_{1 \le t' \le t} \beta^{t-t'} a^u_t$

$0 \le \beta \le 1$是一个标量超参数，代表着与时间相关的衰减强度。如果β是1，则为简单的求平均值，因此没有考虑浏览顺序。

此模型中的训练参数仅为α。

循环模型 ：

循环模型可以分为基于RNN,基于LSTM和基于GRU的模型。

基于RNN的 模型 ：

$u_t = \phi(W_{in} a^u_t + W^{out} u_{t-1} + b)$ ，ϕ为激活函数，这里使用tanh函数，训练参数包括W^{in}, W^{out}，b和初始化状态u_0，u_0是一个不依赖于u的初始值。通过mini-batch的梯度下降法进行学习，如在循环神经网络中的论述，基本的RNN模型存在梯度消失于梯度爆炸问题，当序列变长时，学习变得困难，因此需要门控的循环神经单元LSTM和GRU。

关于LSTM和GRU在循环神经网络部分做了详细说明，这里就不再详述，只简单说明本试验中的一些差异。

基于LSTM的模型 ：

与一般的LSTM不同，这里计算门时，不仅使用了上一层memory信息，还使用了前一步的隐藏层信息。

$$
\begin{aligned}
gi_t &= \sigma(W_{gi}^{in} a_t^u + W_{gi}^{out} u_{t-1} + W_{gi}^{mem} h_{t-1}^u + b_{gi}) \\
gf_t &= \sigma(W_{gf}^{in} a_t^u + W_{gf}^{out} u_{t-1} + W_{gf}^{mem} h_{t-1}^u + b_{gf}) \\
enc_t &= \phi(W_{enc}^{in} a_t^u + W_{enc}^{out} u_{t-1} + b_{enc}) \\
h_t^u &= gi_t \odot enc_t + gf_t \odot h_{t-1}^u \\
go_t &= \sigma(W_{go}^{in} a_t^u + W_{go}^{out} u_{t-1} + W_{go}^{mem} h_t^u + b_{go}) \\
dec_t &= \phi(W_{dec}^{mem} h_t^u + b_{dec}) \\
u_t &= go_t \odot dec_t,
\end{aligned}
$$

需要训练的参数包括多个的W,b以及初始状态u_0, h_0

基于GRU的模型：

该模型与一般的GRU不同的是，在GRU的输出部分，又接了一个全连接层。

$$
\begin{aligned}
gz_t &= \sigma(W_{gz}^{in} a_t^u + W_{gz}^{mem} h^{t-1} + b_{gz}) \\
gr_t &= \sigma(W_{gr}^{in} a_t^u + W_{gr}^{mem} h^{t-1} + b_{gr}) \\
enc_t &= \phi(W_{enc}^{in} a_t^u + W_{enc}^{out} (gr_t \odot h^{t-1}) + b_{enc}) \\
h_t^u &= gz_t \odot enc_t + (1 - gz_t) \odot h_{t-1}^u \\
dec_t &= \phi(W_{dec}^{mem} h_t^u + b_{dec}) \\
u_t &= dec_t.
\end{aligned}
$$

原文中分析了LSTM和GRU在通过分析$|h_t^u|$的上限 分析了LSTM和GRU在梯度爆炸问题上的表现。

$$
sup_u |h_t^u|_{\inf} = \begin{cases} |h_0^u|_{\inf} + t sup_x |\phi(x)|, in, LSTM \\ max(|h_0^u|_{\inf}, sup_x |\phi(x)|), in, GRU \end{cases}
$$

GRU与LSTM上的一个差别是GRU中的updata gate将memory中原信息的擦出与新信息的写入联动起来了，只有当要擦出旧的信息时，才将新的写入，因此，GRU的h_t^u 会有一个上界,而LSTM可以一直写入，因此其上界可能非常大，但是仅仅通过上述说明，就认为基于GRU的模型比基于LSTM的模型具有更高的解决梯度爆炸问题的能力，我认为是有待商榷的。

六、试验结果

1、离线度量

文章中比较了三个基于单词的模型和第五个基于分布式表示的模型，如下图：

Table 1: Model descriptions

Name	Description	Section
BoW	The simplest word-based model	4.2
BoW-Ave	Word-based model that uses average function instead of max in Eq.4	4.2 $+\alpha$
BoW-Dec	Word-based model that uses decayed average function with $\beta = 0.9$ similar to that introduced in Section 4.3	4.2 $+\alpha$
Average	Decaying model with $\beta = 1$ (no decaying)	4.3
Decay	Decaying model with $\beta = 0.9$	4.3
RNN	Recurrent model using simple RNN unit	4.4.1
LSTM	Recurrent model using LSTM-based unit	4.4.2
GRU	Recurrent model using GRU-based unit	4.4.3

关于训练集，测试集的分配以及训练参数等细节，这里就不详述，

接下来主要介绍一些度量的参数和试验结果和结论。

度量参数 ：

评估一个推荐模型的质量是一个比较麻烦的问题，如果使用常用的指标准确率，召回率等，不能很好的表示相关性，评价结果很差。Zygmunt 等提出把推荐当作一个排名任务，可以更好的评估推荐系统的注重相关性。基本精神都是**相关度越高，结果排在前面越好** 。

为了更方便的表述，首先说明一些符号表示：

S :所有待评价sessions的集合

$$c_{s,i} = \begin{cases} 1, clicked, i \\ 0, otherswise \end{cases}$$

定义了三个指标：

AUC是与目标函数直接相关的评价指标，MRR和nDCG是常用的排名指标。

1、AUC（Area under the curve）,即ROC(Receiver operating characteristic)曲线下的面积。

AUC描述的是把正样本排在负样本前的概率 ，只要正样本排在负样本前面就可以得分，没有加权。

ROC分析的是二元分类模型（像我们之前的描述一样，可以将$R(u_t, s^u_{t,p_+}) > R(u_t, s^u_{t,p_-})$看作一个二分类问题），将FPR（$FPR = \frac{FP}{FP+TN}$伪阳性率）作为X轴，TPR（$TPR = \frac{TP}{TP+FN}$真阳性率）为Y轴，根据已经产生的结果，设置不同的阈值，可以得到不同的坐标，这些坐标连成的线为ROC曲线，其下面积为AUC，面积越大，效果越好。

$$AUC = \frac{1}{|S|} \sum_{s \in S} \frac{|(i,j), i<j, c_{s,i}=1, c_{s,j}=0|}{|i|c_{s,i}=1||j|c_{s,j}=0|}$$

2、MRR(mean reciprocal rank) ：其核心是推荐结果的好坏，跟第一个正确答案的位置有关，第一个正确答案越靠前，结果越好,推荐时，如果点击的位置排在第n位，那么此次MRR分数就为$\frac{1}{n}$ 。

$$MRR = \frac{1}{S} \sum_{s \in S, c_{s,i}=1} \frac{1}{min_i}$$

3、nDCG(normalized discounted cumulative gain)：

AUC中只要正样本排在负样本前面就可以得分，nDCG做了加权，它不仅希望正样本排在负样本之前，而且正样本越靠前，得分越高（通过除以$log2^{i+1}$实现）

$nDCG = \frac{1}{S}\sum_{s\in S}\frac{\sum_i \frac{c_{s,i}}{log_2(i+1)}}{max_\pi \sum_i \frac{c_{s,i}}{log_2(\pi(i)+1)}}$ π是任意分布的位置,分母项表示最佳排序的得分，是为了归一化处理，如果分子项越接近最佳排序，则nDCG得分为1。

试验的结果 ：

将测试数据集分成10个子数据集，并计算每个子数据集的每个度量标准。表2中的值是子数据集的每个度量的平均值，以及基于t分布估计的每个度量的99%置信区间。

Table 2: Results from offline experiments. Values indicate average of metrics and 99% confidence intervals in ten split test sets.

	AUC	MRR	nDCG
BoW	0.582 ± 0.003	0.300 ± 0.003	0.446 ± 0.002
BoW-Ave	0.579 ± 0.004	0.310 ± 0.003	0.452 ± 0.002
BoW-Dec	0.560 ± 0.004	0.297 ± 0.004	0.442 ± 0.003
Average	0.608 ± 0.003	0.313 ± 0.003	0.457 ± 0.002
Decay	0.596 ± 0.003	0.302 ± 0.002	0.449 ± 0.001
RNN	0.612 ± 0.004	0.309 ± 0.004	0.455 ± 0.003
LSTM	0.648 ± 0.004	0.344 ± 0.004	0.481 ± 0.003
GRU	**0.652 ± 0.003**	**0.347 ± 0.004**	**0.484 ± 0.003**

由上表可以看到：

1、基于LSTM和GRU的模型显著好于其他模型，有非常显著的提升。而且每个评测中中GRU都表现最好。

2、Decay 和Bow-Dec比Average 和BoW-ave更差，说明访问历史的信息不能被简单的表示为衰减。

3、RNN在AUC评测上有轻微的提升，但是LSTM和GRU显著好于其他，这是因为使用LSTM和GRU中的门结构可以更好的表达浏览顺序之间更复杂的关系（可以更好的解决长期依赖问题）。

2、在线度量

为了比较性能，使用GRU模型来处理主要的流量，对于1%的用户使用传统的BoW模型。

在线度量的4个指标：

Sessions :一位用户每天使用推荐服务的平均次数

Duration(持续时间) ： 用户每次使用推荐服务的平均时间（以秒为单位）。用户查看推荐列表的总时间以及点击该文章后阅读文章的时间。

Clicks :每次会话的平均点击次数（与第四节中的$|P_+|$）

Click through rate ：点击次数/显示的文章数。 如果文章的检索效率低，这些值会下降，用户要花更多的时间去探索推荐列表。

最重要的评价是一个人的总的持续时间（Session * Duration）

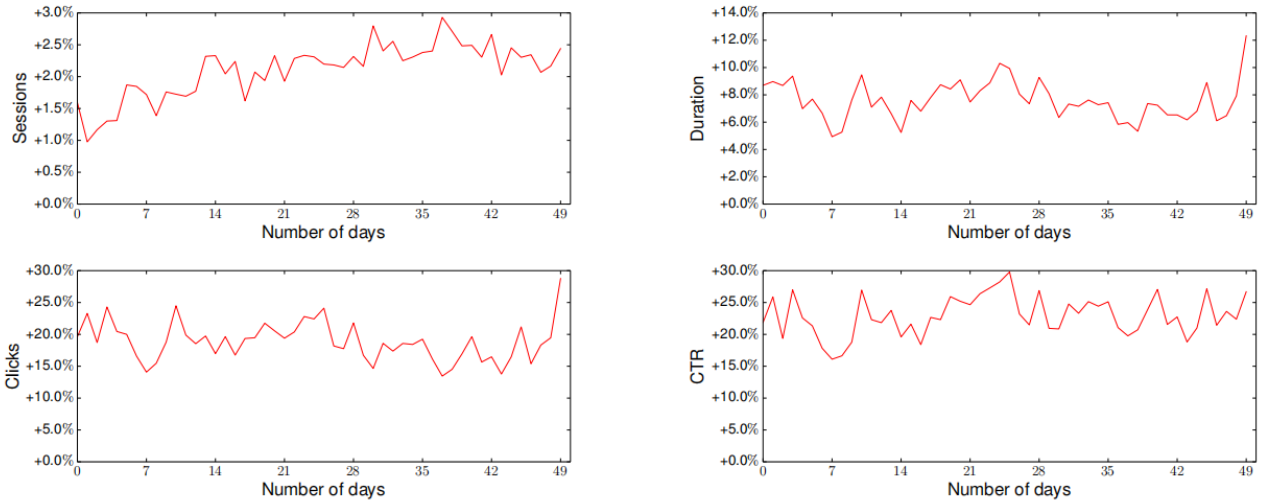


Figure 6: Transition in lift rate for metrics.

上图绘出了每个度量指标每日的提升（基于GRU的模型 相对于BoW模型），所有的度量指标在基于GRU的模型上都获得显著的提升。持续时间，点击率和CTR从新模型应用第一天就表现出一个稳定的提升率。**Sessions** 在开始表现出相对较小的改善率，但逐渐获得更多改善。这意味着一个好的推荐模式会首先增加点击次数，而多次点击体验会鼓励用户更频繁地使用该服务。

Table 3: Average metric lift rates in 7th week and those by user segments.

Metric	ALL	Heavy	Medium	Light
Sessions	+2.3%	+1.1%	+1.0%	+1.8%
Duration	+7.8%	+4.9%	+13.3%	+17.4%
Clicks	+19.1%	+14.3%	+26.3%	+42.3%
CTR	+23.0%	+18.7%	+29.8%	+45.1%

上表描述了对于每个度量指标，总的平均提升率和每个细分用户平均提升率。

细分为三类，使用者的划分率比例大约为：Heavy:Medium：Light = 3 :2 :1。

Heavy:在前一周访问超过五天的用户。

Medium:上一周访问在两天到五天的

Light：上周访问小于两天的

所有的细分度量都得到了提升。轻用户（Light）表现出特别大的改善。因此，可以假设：使用词袋模型表示，那些有很少访问历史的用户受特征稀疏的影响非常大。而且由于突然兴趣引起的浏览噪音严重影响了推荐的准确性，采用分布式表示和GRU的门结构可以很好的改善上述问题。

Table 4: Average daily percentage of user composition in 7th week for both buckets.

Bucket	ALL (%)	Heavy (%)	Medium (%)	Light (%)
Control	100.0	50.0	33.9	16.1
Proposed	100.0	51.4	33.6	15.0
0th week (ref.)	100.0	49.7	34.3	16.0

除了每个细分市场内指标的改进之外，还发现用户从轻质到中等以及中到重的转变，这也解释了为什么Sessions的总体增长率高于表3中每个部分的增长率。

面向大规模基于深度学习的服务部署的调整

将机器学习应用于实际新闻发布系统时，使用最新数据保持更新模型非常重要。Bow模型由于非常简单，每三个小时可以使用最新的文章数据和会话数据来更新模型。但GRU无法做到频繁的更新，主要原因如下：

1、模型训练模型时间很长。实践上，基于GRU的模型训练需要超过一周。

2、如果我们更新文章表示的模型，我们将不得不重新计算所有得到的文章。

3、如果我们更新使用者的表示模型，我们将不得不重新计算使用者的状态从过去的第一次浏览开始。在实践中，只能放弃对一段时间内旧历史的重新计算。

4、用户偏好表示和文章表示必须同步更新。

基于词的模型对时髦词（新词）很敏感，只要停止更新几天就会变坏。然而，基于分布式表示的模型在模型更新的频率下保持足够的准确性。如果我们三个月不更新GRU模型，它的结果就会和每三个小时更新的模型有相同的准确率。

7、总结

本文总结的方法使用分布式表示的基于嵌入的方法，以三步端到端的方式实现。

1、使用一个降噪自动编码器的变体对文章进行分布式表示

2、对访问历史序列使用RNN生成用户偏好的表示

3、考虑到效率，使用用户和文章的分布式表示的内积为用户匹配和列出文章。

我们发现，即使在具有大规模流量的真实新闻分发系统中，该方法也很有效，因为在设计时就考虑到了它的具体实现。目前该方法已经完全纳入了Yahoo JAPAN网页在智能手机上的所有流量，它每天向数百万用户推荐各种文章。也可以将该方法应用于广告等其他领域。