Welcome

MSRA Researcher新手，现阶段从事多模态大模型预训练方向。对NLP感兴趣或对博客内容有疑惑及意见建议的，欢迎评论或添加我微信。此外如果有需要内推的同学，也欢迎来骚扰我。联系方式详见concat页面。

前往GitHub (https://github.com/Dod-o)

梳理|对话系统中的DST

6月 9, 2019 (http://www.pkudodo.com/2019/06/)

梳理|对话系统中的DST

作者 Dodo (http://www.pkudodo.com/author/root/) 在 (http://www.pkudodo.com/2019/06/09/1-12/)
对话系统 (http://www.pkudodo.com/category/%e5%af%b9%e8%af%9d%e7%b3%bb%e7%bb%9f/) 标签
DST (http://www.pkudodo.com/tag/dst/), SLU (http://www.pkudodo.com/tag/slu/),
对话系统 (http://www.pkudodo.com/tag/%e5%af%b9%e8%af%9d%e7%b3%bb%e7%bb%9f/)

最近看了任务型对话中的SLU和DST部分，感觉里面方法结构很多，论文看多了以后脑袋就乱了。所以写篇文章给自己梳理一下，这篇文章**先写DST部分**，其他部分以后会再更新出来。

也正是因为这样，所以这篇文章的**正确性还有待考究**。有很大可能性会在未来的某个时间对文章的内容做一些修改。大家看一看就好了。

chatbot对话类型

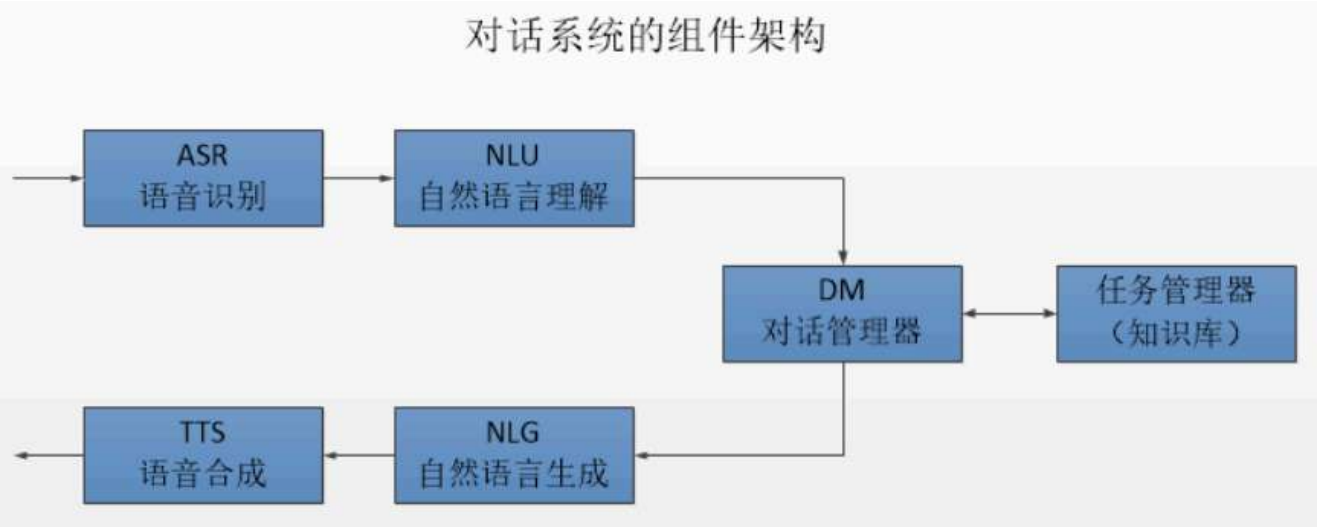
chatbot对话分为任务型、闲聊型、推荐型、知识问答型。这里讨论一下任务型和闲聊型。

任务型：以小米的小爱同学为例，通常负责为用户执行任务，例如订票、查阅信息、控制家电等。这类任务需要维护一个系统状态，知晓当前用户的意图、完成任务还需要的信息等，最后确定用户实际任务需求并执行灯舞。

闲聊型：以微软的小冰为例，通常在开放域环境下聊天，即没有特定的领域、任务，满足用户的情感交流需要。

任务型对话

任务型对话的处理方式有pipeline和端到端两种结构，pipeline定义了数个模块，以一条line的形式串联起来共同完成一个任务，如下图所示。端到端的代表为memory network，至于这块我还没有看相关论文，因此不作展开。



(http://www.pkudodo.com/wp-content/uploads/2019/06/pipeline.png)

pipline模块

其核心模块组成是NLU->DM->NLG，先通俗来讲，NLU负责对用户输入进行理解，随后进入DM模块，负责系统状态的追踪以及对话策略的学习，控制系统的下一步动作，而NLG则配合系统将要采取的动作生成合适的对话反馈给用户。

其中若用户的输入是语音形式，则在NLU的输入前需再添加一个ASR语音识别模块，负责将语音信号转换为文本信号。DM对话管理器内部又可分为DST（对话状态追踪）和DPL（对话策略学习），DST（对话状态追踪）根据用户每一轮的输入更新当前的系统状态，而DPL则根据当前的系统状态决定下一步采取何种动作。NLG将语言生成后，若用户采用语音交互方式，则还需要TTS（语音合成）模块将文本转换为语音。

pipline工作原理直观理解

ASR：这部分在论文中大部分都是一笔带过，因为它的任务比较单一，只负责将语音转换为文本信号。不过值得一提的是，有些论文提起ASR的输入并不是唯一的，因为语音识别可能会存在一定错误，因此一般会输出多个可能的句子，每个句子同时附带一个置信度，表示这个句子正确的概率。这种方式在论文中被称为N-best，及前N个最有可能的句子。

NLU：语言理解模块，用户语音转文本后称为用户Utterance，NLU负责对用户Utterance进行领域/意图分类及槽值对填充。其中领域和意图分类是为了让系统明白用户的对话所处领域及意图，方便后续调用相应的model去识别（并不是一个model跑遍所有的领域意图，就好像树一样，根据领域/意图的分类，在树中找到对应的model。当然model结构可能是一样的，不同的是训练采用的数据是对应领域/意图的）。完成一个任务需要去弄清楚一些条件，比如说点一杯咖啡，根据领域/意图分类，系统判断用户的意图是点咖啡，此时系统需要弄明白是什么咖啡，甜度怎么样等等。所以会检索点咖啡所需要的槽值对，这个时候后台检索发现完成这个任务需要弄明白{咖啡类型=?，甜度=?}（其中咖啡类型和甜度被称为槽（slot）），这个时候系统会反馈回去问用户咖啡类型是什么？甜度是多少？用户反馈后，NLU再对用户Utterance内容进行识别，发现咖啡类型是摩卡，即槽"咖啡类型"的值（value）为"摩卡"，这个过程被称为槽填充。同时这个过程也很容易被人联想到命名实体识别这一方法。当然了，考虑到系统的准确率，**一般情况下生成的槽值对也会附带一个置信度**，也就是说，对于一个slot，可能并不会只输出一个value。

DST：DST（对话状态追踪）归属于DM对话管理器中，负责估计用户的当前轮的目标，它是对话系统中的核心组成部分。在工作过程中维护了一个系统状态（我认为这个系统状态其实就是各个槽对应的值以及相应的概率），并根据每一轮对话更新当前的对话状态（各个槽值对）。这里我说一些自己的感觉，但可能不太对，我认为直观上来

看，SLU输出了slot-value，但是是不确定的，也就是说可能会输出{咖啡类型=摩卡}–0.8 {咖啡类型=拿铁}–0.2，SLU认为这次用户要求的是摩卡的概率是0.8，是拿铁的概率为0.2，并没有输出一个确定值。所以DST需要结合当前的用户输入（即SLU输出的槽值对）、系统上一时刻的动作（询问需要什么类型的咖啡）以及之前多轮对话历史来判断咖啡类型到底是哪个，最后计算得到{咖啡类型=摩卡}–0.9，认为是摩卡的概率为90%，这是DST评估后认为咖啡类型的当前状态。当然还有很多其他的槽，可能甜度还没有问过，所以{甜度=none}，等待DPL去询问用户。这些所有的槽值对的状态，被统称为当前的系统状态，每个轮次结束后都会对当前的系统状态做一次更新。

DPL：对话策略管理是根据DST输出的当前系统状态来判断还有哪些槽需要被问及，去生成下一步的系统动作。这部分和NLG的论文都还没看，所以不展开了。

NLG：DPL生成下一步的系统动作后，生成相应的反馈，是以文本形式的。

TTS：若用户是语音交互，则将NLG输出的文本转换为对应语音即可。这部分与ASR差不多，功能相反而已。

DST：对话状态追踪

DST归根结底最终要的还是评估判断当前的用户目标、维护当前系统状态。因此对于slot-value需要确定其匹配情况。因此一般都是对于一个slot建立一个多分类模型，分类数目是slot对应的value数目。

常用方法：DNN、RNN、NBT、迁移学习（迁移学习部分还没看，后续会更新到文章末尾）

论文一：Deep Neural Network Approach for the Dialog State Tracking Challenge

这篇论文核心观点是使用n-gram滑动窗口，同时手工构造了12个特征函数来抽取特征，随后将所有特征送入DNN，最后对slot的所有可能value计算概率，概率最高的即为slot对应的value。每一个slot都会有一个对应的model，因此如果该intention内有n个slot需要填充，则系统内有n个该model。

12个特征如下图所示，比较好懂就不翻译了，抽取到的特征涵盖了该轮SLU的槽值对输出、系统上一时刻动作、多轮对话历史。所以总体上还是满足了DST对历史对话及当前输入的考量。

1. **SLU score**; the score assigned by the SLU to the user asserting s=v.
2. **Rank score**; $1/r$ where r is the rank of s=v in the SLU n-best list, or 0 if it is not on the list.
3. **Affirm score**; SLU score for an `affirm` action if the system just confirmed s=v.
4. **Negate score**; as previous but with `negate`.
5. **Go back score**; the score assigned by the SLU to a `goback` action matching s=v.
6. **Implicit score**; 1− the score given in the SLU to a contradictory action if the system just implicitly confirmed s=v, otherwise 0.
7. **User act type**; a feature function for each possible user act type, giving the total score of the user act type in the SLU. Independent of s & v.
8. **Machine act type**; a feature function for each possible machine act type, giving the total number of machine acts with the type in the turn. Independent of s & v.
9. **Cant help**; 1 if the system just said that it cannot provide information on s=v, otherwise 0.
10. **Slot confirmed**; 1 if s=v' was just confirmed by the system for some v', otherwise 0.
11. **Slot requested**; 1 if the value of s was just requested by the system, otherwise 0.
12. **Slot informed**; 1 if the system just gave information on a set of bus routes which included a specific value of s, otherwise 0.

(http://www.pkudodo.com/wp-content/uploads/2019/06/特征.png)其model架构如下图所示。model对历史对话的信息重点放在前T轮，也就是t-T+1到当前轮次t，对于每个轮次都根据特征函数抽取了信息。此外T轮之前的信息，作者可能认为参考意义不是很大，因此直接将t-T+1轮之前的特征直接做了一个总和，单独作为一个输入。如果当前轮次t<T，那么该输入的所有特征值都为0。

之后将所有特征函数送入3层隐层的DNN网络，最后得到一个实数表示概率值，该概率值体现了当前slot-value匹配的置信程度。随后使用其他value再次输入网络，得到相应的概率值。最后得到slot所有可取value的概率情况，所有概率用来更新当前的系统状态信息。

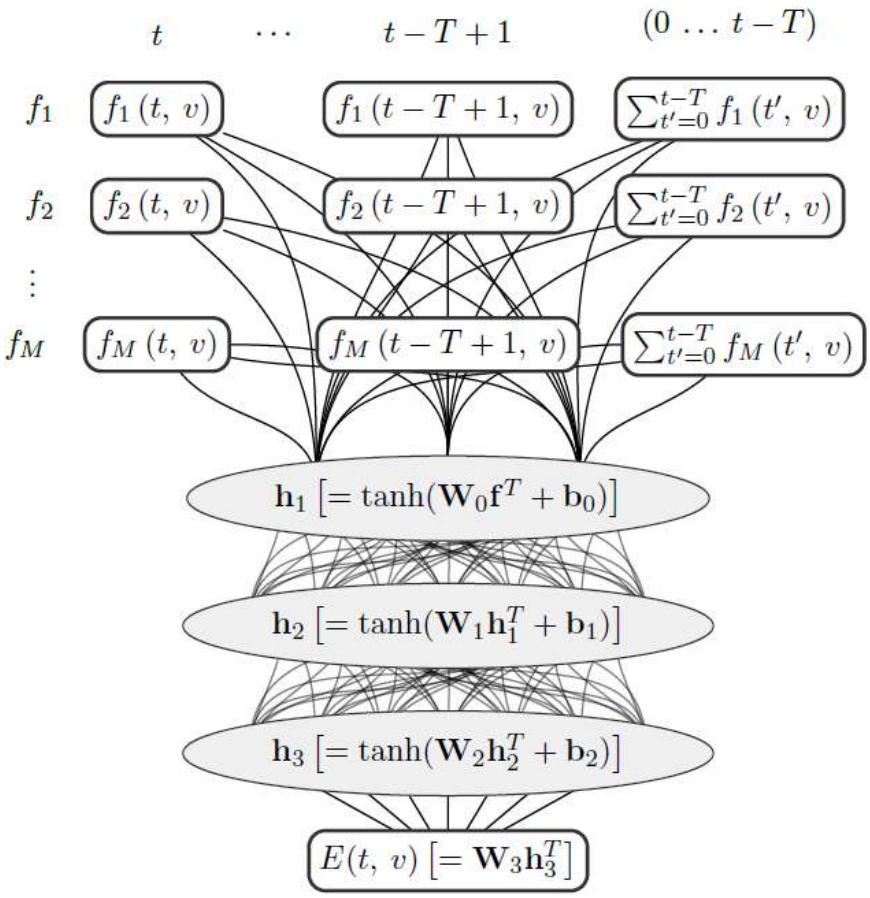


Figure 1: The Neural Network structure for computing $E(t, v) \in \mathbb{R}$ for each possible value v in the set $S_{t,s}$. The vector $\mathbf{f}$ is a concatenation of all the input nodes.

(http://www.pkudodo.com/wp-content/uploads/2019/06/网络结构1.png)总的来说这篇论文的亮点我认为在于构造了12个合理的特征函数，这个的工作量还是蛮大的。不过缺点也是很多model的通病，就是对于每一个slot，都需要有足够的训练集去拟合。这对于经常缺少训练集的工业界来说，是一个问题。后面的论文中有一些试图在解决这一问题。

论文二：Word-Based Dialog State Tracking with Recurrent Neural Networks

这篇文章的立意是考虑到ASR输出用户Utterance后需要再通过SLU，随后才进行DST。可是ASR可能会出错，SLU也可能会出错，这样会造成一个error传播。因此作者设计的model直接以ASR的输出作为DST的输入，绕过了SLU部分。这种策略目前在paper中也比较常见，一般来说效果也确实比添加SLU模块的要高一些。

然后和上一篇文章相似的是，它也选择去抽取一些特征，但特征是输入RNN网络（上一篇文章是DNN）。不过在特征的抽取上，两者的方法差异还是蛮大的。

特征构造如下图所示，它从ASR和Machine Act两个方面去构造特征，之后将对应的特征串联，最后得到f、fs、fv三个特征向量，其中f是直接对ASR结果和对话动作取n-gram的结果，fs和fv是对f的结果进行tag替换得到的泛化恩正。fs是对槽提取的特征，适用于该槽下对应的所有的值，fv是对每个value进行提取，属于每个值得特有特征。这样有助于模型的泛化，当遇到训练集中没有出现过的同义词时，同样可以发挥作用。

先看第一层，ASR输出两个句子并附加置信度，第一层特征抽取采用1-gram、2-gram、3-gram，并且抽取出来的词的置信度跟随ASR的输出，若出现了相同的item，则把概率相加即可。Machine Act也作同样处理，只不过概率全部为1。最后抽取到的特征ASR部分和Machine Act单独相加，再串联形成特征f。不过这里存在一个问题，怎么

将特征表示为向量？这事在论文中没有说明，不过论文中提到最后得到的向量是3500维的，因为我认为是one-hot形式，3500为词典的大小，同时对应位置的值为inten对应的概率值。下面两层也是一样的，就不再说明了。

第二层对槽进行提取，使用<slot>和<value>替换掉n-gram的结果，同时相同的item也需要概率相加。使用<slot>和<value>这种同样的符号来替代是为了提高模型泛化性，与第一层同样串联处理得到fs。

第三层同理，不再展开。

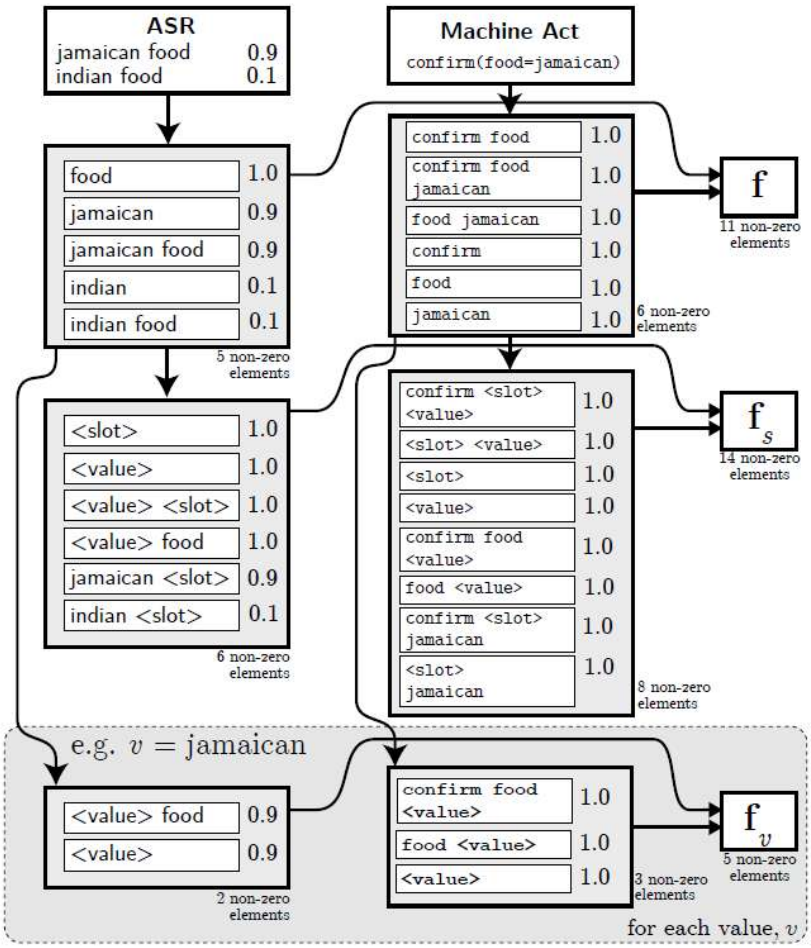


Figure 1: Example of feature extraction for one turn, giving $\mathbf{f}$, $\mathbf{f}_s$ and $\mathbf{f}_v$. Here $s = \text{food}$. For all $v \notin \{\text{indian, jamaican}\}$, $\mathbf{f}_v = \mathbf{0}$.

(http://www.pkudodo.com/wp-content/uploads/2019/06/特征1.png)三个特征向量构造出来后，输入RNN，

RNN内部结构如下所示。需要说明的是，DST的model如非特殊说明，单个model都是针对单个slot的，不同的slot需要多个model，差别只在于训练数据的不同。

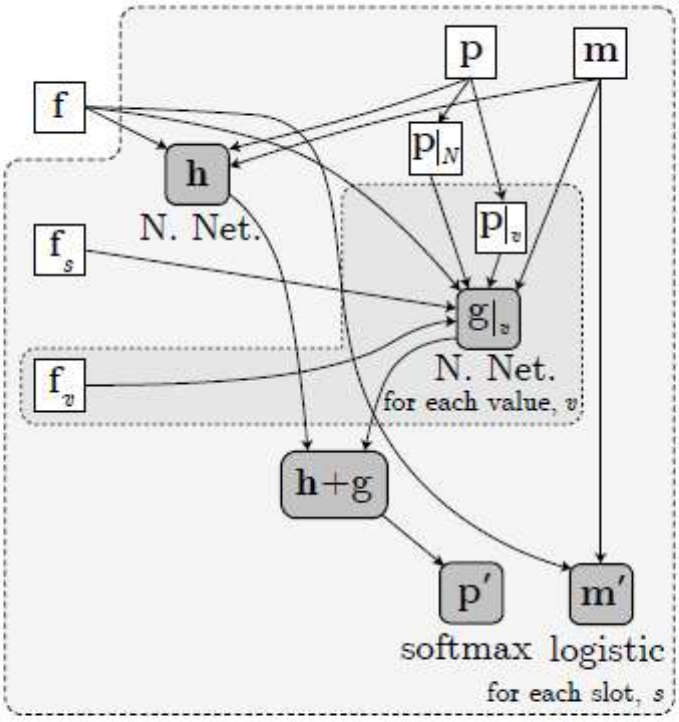


Figure 2: Calculation of $\mathbf{p}'$ and $\mathbf{m}'$ for one turn

(http://www.pkudodo.com/wp-content/uploads/2019/06/rnn.png)

第一部分是用户Utterance、记忆、向量f的结合，串联后输入网络，得到向量h。

$$\mathbf{h} = \text{NNet}(\mathbf{f} \oplus \mathbf{p} \oplus \mathbf{m}) \in \mathbb{R}^N$$

(http://www.pkudodo.com/wp-content/uploads/2019/06/第一部分.png)之后将fs、fv也输入到rnn中，扩展RNN所使用的特征。

$$\mathbf{g}|_v = \text{NNet}\left(\begin{array}{c}\mathbf{f} \oplus \mathbf{f}_s \oplus \mathbf{f}_v \oplus \\ \{\mathbf{p}|_v,\ \mathbf{p}|_N\} \oplus \mathbf{m}\end{array}\right) \in \mathbb{R}$$

(http://www.pkudodo.com/wp-content/uploads/2019/06/第二部分.png)将h合g结合，使用softmax得到概率分布，生成新的状态（也就是slot对应的各个value的值），并同时更新记忆。

$$\mathbf{p}' = \text{softmax}([\mathbf{h} + \mathbf{g}] \oplus \{B\}) \in \mathbb{R}^{N+1}$$

(http://www.pkudodo.com/wp-content/uploads/2019/06/第三部分.png)

$$\mathbf{m}' = \sigma(W_{m0}\mathbf{f} + W_{m1}\mathbf{m}) \in \mathbb{R}^{N_{\text{mem}}}$$

(http://www.pkudodo.com/wp-content/uploads/2019/06/第四部分.png)此外论文还提到了一些小技巧，这些不属于model本身范围内，不再作讨论。

论文三：Neural Belief Tracker: Data-Driven Dialogue State Tracking

这篇文章也是将SLU合并到了DST当中，除了结构外，文章本身很值得一读。在介绍自身的同时对当前的一些主流方法简单提了一些，可以起到一个思维梳理的过程。

先把model框架贴出来。

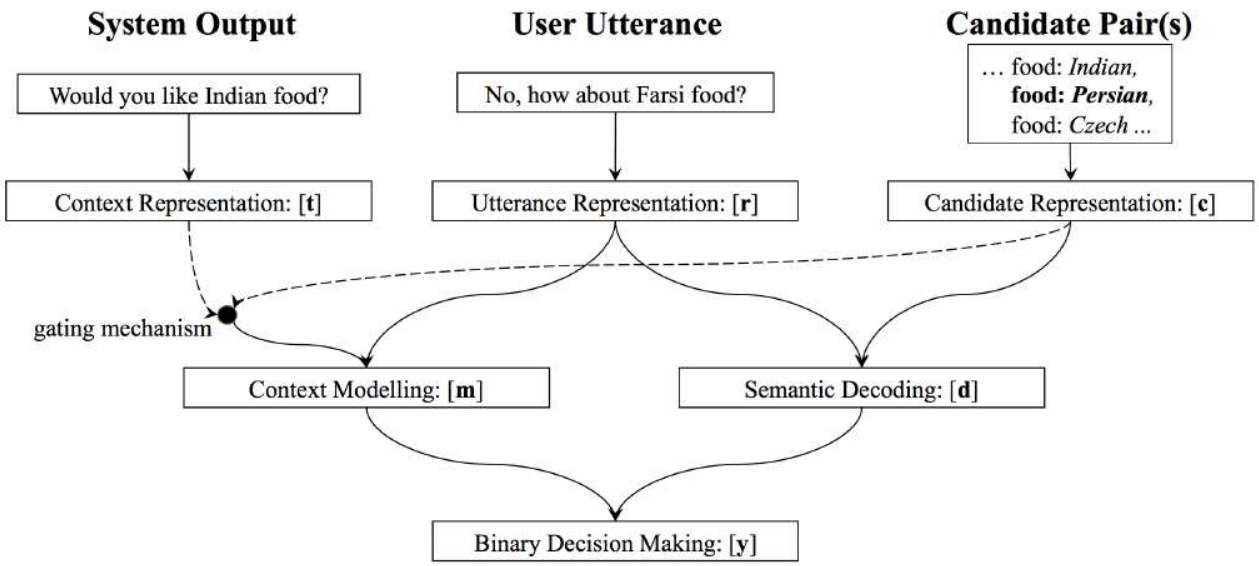


Figure 3: Architecture of the NBT Model. The implementation of the three representation learning subcomponents can be modified, as long as these produce adequate vector representations which the downstream model components can use to decide whether the current candidate slot-value pair was expressed in the user utterance (taking into account the preceding system act).

(http://www.pkudodo.com/wp-content/uploads/2019/06/DBN.png)model中可以看到一共有三个输入，System Output（上一时刻系统动作）、User Utterance（用户输入）、Candidate Pairs（候选槽值对）。model要做的就是根据系统之前动作及用户当前输入，判断候选槽值对中那个value才是真正的value。

User Utterance的表示

user utterance最终应当表示成向量形式，论文给了DNN和CNN两种抽取方式。

首先介绍一下两者共同的地方，首先都将用户utterance中的词用词向量表示，之后使用1-gram、2-gram、3-gram抽取词向量，下式中u为对应的词向量，n为gram的长度。

$$\mathbf{v}_i^n = \mathbf{u}_i \oplus \ldots \oplus \mathbf{u}_{i+n-1}$$

(http://www.pkudodo.com/wp-content/uploads/2019/06/1.png)

NBT-DNN

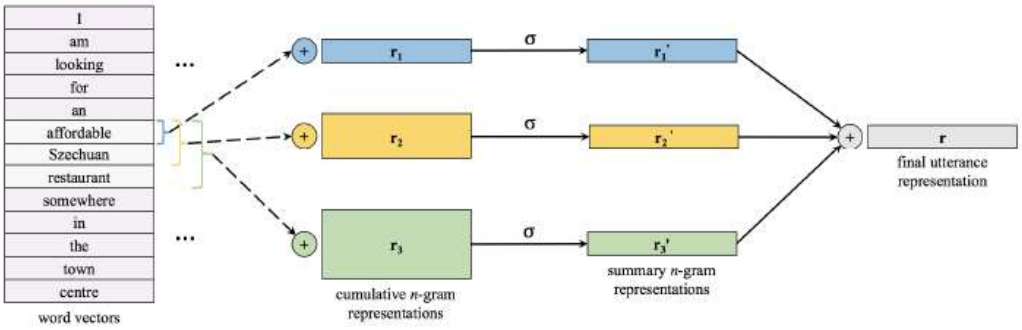


Figure 4: NBT-DNN MODEL. Word vectors of n-grams ($n = 1, 2, 3$) are summed to obtain *cumulative* n-grams, then passed through another hidden layer and summed to obtain the utterance representation $\mathbf{r}$.

(http://www.pkudodo.com/wp-content/uploads/2019/06/NBT-DNN.png)抽取到n-gram后，对应于不同的n，生成三个词向量矩阵。

$$\mathbf{r}_n = \sum_{i=1}^{k_u - n + 1} \mathbf{v}_i^n$$

(http://www.pkudodo.com/wp-content/uploads/2019/06/2.png)之后由于三个向量的维度不一致，因此使用一层nn将维度转换到相同。

$$\mathbf{r}'_n = \sigma(W_n^s \mathbf{r}_n + b_n^s)$$

(http://www.pkudodo.com/wp-content/uploads/2019/06/3.png)得到的三个向量直接相加即可，最后的输出r就是最终的用户utterance向量。不过这种方式存在严重的缺陷，它对于句子中的组成部分并没有设计权重，所以句子中例如停用词或其他需要被忽略的词并没有得到处理，最终得到的句向量噪音很大。这种方式只适合句子较短的情况下，所以论文又提出了基于CNN的方式，可以将attention着重放在对句子有帮助的地方。

NBT-CNN

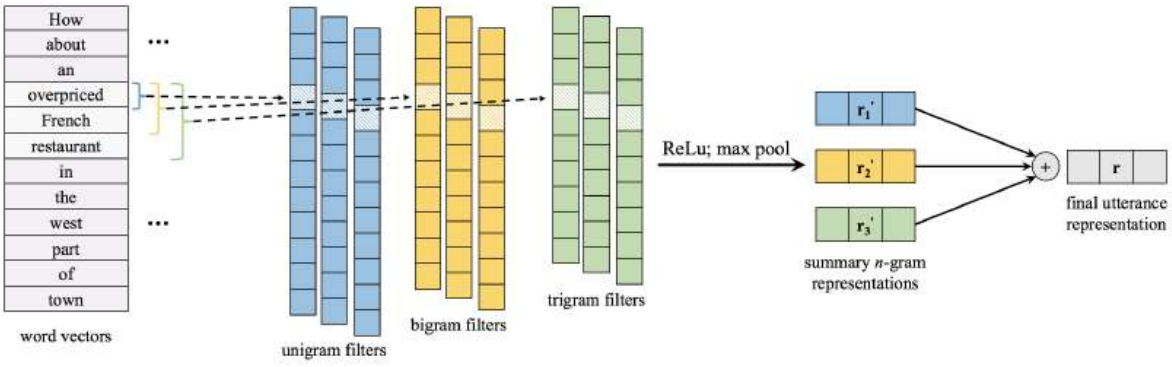


Figure 5: NBT-CNN Model. L convolutional filters of window sizes $1, 2, 3$ are applied to word vectors of the given utterance ($L = 3$ in the diagram, but $L = 300$ in the system). The convolutions are followed by the ReLU activation function and max-pooling to produce summary n-gram representations. These are summed to obtain the utterance representation $\mathbf{r}$.

(http://www.pkudodo.com/wp-content/uploads/2019/06/NBT-CNN.png)结构还是比较简单的，n-gram抽取向量之后做卷积，最后再maxpool提取信息，最后得到用户utterance向量。细节可以看一下论文，就不多展开了。

采用这种方式使得最后得到的utterance向量具有一定的代表性和侧重性。

语义解码

结合用户语义与槽值对进行处理，论文中提到这一步主要是因为用户utterance和槽值对是有直接交互的。它决定了用户utterance包含的内容是否与候选槽值对有相关性。

$$\mathbf{c} = \sigma\left(W_c^s(\mathbf{c_s} + \mathbf{c_v}) + b_c^s\right)$$
$$\mathbf{d} = \mathbf{r} \otimes \mathbf{c}$$

(http://www.pkudodo.com/wp-content/uploads/2019/06/4.png)

最后得到了向量d，不过其实在形式上，第二个式子很像那种相似度计算的形式（小声叨叨）。

上下文模型

这个时候用户utterance和候选槽值对已经处理了，还剩下上一时刻的系统输出。也就是说DST还得明白上一时刻系统要求用户提供什么信息了。

这里系统定义了两种操作：

1. 系统请求：上一时刻系统要求用户提供某种信息，例如"您大概需要什么价位的呢？"这样子系统必须提供价格槽用于填充，而不是地区或者食物类型等等。
2. 系统确认：上一时刻系统要求用户确认某种信息，例如"您是需要土耳其食物是吗？"，用户需要回到是或者不是。

可能会有疑问，只需要知道上一时刻的系统输出就可以了吗？当然最好还是能够所有历史动作及对话历史都考虑到，但由于这样会使系统过于复杂，因此论文在策略上使用马尔科夫形式，只考虑前一时刻的系统动作。

$$\mathbf{m_r} = (\mathbf{c_s} \cdot \mathbf{t_q})\mathbf{r}$$
$$\mathbf{m_c} = (\mathbf{c_s} \cdot \mathbf{t_s})(\mathbf{c_v} \cdot \mathbf{t_v})\mathbf{r}$$

(http://www.pkudodo.com/wp-content/uploads/2019/06/5.png)

决策生成

最后所有特征都已转换完成，直接进入一个二分类决策model，最后使用一个softmax输出slot-value匹配的概率。网络内部很简单，就是添加一个隐层就可以了。

$$\mathbf{y} = \phi_2\left(\phi_{100}(\mathbf{d}) + \phi_{100}(\mathbf{m_r}) + \phi_{100}(\mathbf{m_c})\right)$$

(http://www.pkudodo.com/wp-content/uploads/2019/06/6.png)

下标100表示最后得到的维度是100，y最后的维度是2维，分别是肯定或者否定的概率。

写在最后

DST当时给我最大的困扰是系统状态究竟是什么以及它到底核心任务是什么。

我觉得可以再写一写梳理一下以及再写一些自己的困惑，未来如果我能够解答了，会在下面重新更新的。（这部分可能理解还是错的，但是是我目前认为自己能够给出的比较合理的解释）

DST系统状态：因为任务型对话中slot-value配对是比较重要的，但是系统并不能保证给出的slot-value对已经是正确的。所以SLU在做槽值对匹配的时候，只能在给出槽值对的同时再给出一个概率来表示置信度。

> 可是我看了一些SLU的论文，paper中并没有说会输出置信度。比如说用命名实体识别方法，paper中直接给出了slot-value。感觉这个事情很奇怪。所以后来我认为根据DST采取的策略不同，可能会对置信度没有要求。但这两周陆陆续续也看了将近十几篇论文，DST的核心仍然在维持一个系统的置信状态。这件事是我接下来想去搞懂的一个点。

因为每一个slot都有很多个value可以被填充进去，那么在每一轮对话中需要更新slot中所有value候选值的概率。所有slot和其对应的value概率，汇总起来可以被称为系统状态（我觉得）。

DST核心任务：DPL的论文还没有看，就目前来看，DST主要工作就是更新系统状态，试图捕捉用户的真实意图（意图通过槽值对体现）。后续DPL可能会根据当前的系统状态去合理推断下一步应该做什么，才能保证所需要的槽全部被填充，同时所有的槽值对都是可信的。

Dodo

8条评论

匿名 **发布于7:05 上午 - 1月 28, 2023**

Hi pkudodo.com owner, Thanks for the well-structured and well-presented post!

()

回复

()

匿名 **发布于7:58 下午 - 1月 9, 2023**

Hello there! This is my 1st comment here so I just planned to give A fast shout out and say I actually delight in examining your articles or blog posts. Are you able to suggest another blogs/Web sites/forums that handle the identical subjects? Thanks a ton! sasilu.se/map6.php blanda finbetong file?¶r hand

回复

()

匿名 **发布于7:53 下午 - 1月 9, 2023**

Soon after I at first left a remark I appear to acquire clicked to the -Notify me when new responses are added- checkbox and Any longer every time a remark is extra I acquire four e-mail with the similar remark. Potentially You will find a usually means you can easily take away me from that services? Several thanks!

回复

()

匿名 **发布于7:36 下午 - 1月 9, 2023**

I used to be advisable this weblog by my cousin. I'm not sure irrespective of whether this submit is prepared by him as no person else know this kind of in-depth about my trouble. You're unbelievable! Many thanks!

回复

()

匿名 **发布于11:10 上午 - 10月 31, 2022**

Hello there great website! Does running a blog much like this acquire a huge quantity work? I've virtually no knowledge of Personal computer programming even so I was hoping to begin my very own site during the around long term. In any case, When you have any ideas or tips for new blog entrepreneurs remember to share. I am aware this is off issue but I simply needed to inquire. Lots of thanks!

回复

匿名 **发布于3:20 上午 - 10月 31, 2022**

Quite! This was a really fantastic post. Many many thanks for furnishing these specifics

()

回复

匿名 **发布于4:13 上午 - 9月 8, 2022**

bunadisisj nsjjsjsisjsmizjzjjzjzz zumzsert

()

回复

匿名 **发布于10:12 上午 - 11月 23, 2021**

p895xne

()

回复

发表评论

内容(链接请去除http协议头，否则会被误判为垃圾评论)

设置显示昵称(选填)

发送信息

近期文章

- 论文|记忆网络之Memory Networks (http://www.pkudodo.com/2019/06/14/1-13/)
- 梳理|对话系统中的DST (http://www.pkudodo.com/2019/06/09/1-12/)
- 论文阅读|How Does Batch Normalizetion Help Optimization (http://www.pkudodo.com/2019/05/23/1-11/)
- 学习规划|机器学习和NLP入门规划 (http://www.pkudodo.com/2019/03/20/1-10/)
- 面试体会|微软、头条、滴滴、爱奇艺NLP面试感想 (http://www.pkudodo.com/2019/03/10/1-9/)

文章归档

- 2019年6月 (http://www.pkudodo.com/2019/06/)
- 2019年5月 (http://www.pkudodo.com/2019/05/)
- 2019年3月 (http://www.pkudodo.com/2019/03/)
- 2018年12月 (http://www.pkudodo.com/2018/12/)
- 2018年11月 (http://www.pkudodo.com/2018/11/)
- 2018年10月 (http://www.pkudodo.com/2018/10/)
- 2016年9月 (http://www.pkudodo.com/2016/09/)

标签

DST (http://www.pkudodo.com/tag/dst/)

K近邻 (http://www.pkudodo.com/tag/k%e8%bf%91%e9%82%bb/)

LSI (http://www.pkudodo.com/tag/lsi/)

memory network (http://www.pkudodo.com/tag/memory-network/)

MQTT (http://www.pkudodo.com/tag/mqtt/)

NLP (http://www.pkudodo.com/tag/nlp/)

s3c2440 (http://www.pkudodo.com/tag/s3c2440/)

SLU (http://www.pkudodo.com/tag/slu/)

SVM (http://www.pkudodo.com/tag/svm/)

VSM (http://www.pkudodo.com/tag/vsm/)

体会 (http://www.pkudodo.com/tag/%e4%bd%93%e4%bc%9a/)

决策树 (http://www.pkudodo.com/tag/%e5%86%b3%e7%ad%96%e6%a0%91/)

分类器 (http://www.pkudodo.com/tag/%e5%88%86%e7%b1%bb%e5%99%a8/)

实现 (http://www.pkudodo.com/tag/%e5%ae%9e%e7%8e%b0/)

对话系统 (http://www.pkudodo.com/tag/%e5%af%b9%e8%af%9d%e7%b3%bb%e7%bb%9f/)

感想 (http://www.pkudodo.com/tag/%e6%84%9f%e6%83%b3/)

感知机 (http://www.pkudodo.com/tag/%e6%84%9f%e7%9f%a5%e6%9c%ba/)

支持向量机 (http://www.pkudodo.com/tag/%e6%94%af%e6%8c%81%e5%90%91%e9%87%8f%e6%9c%ba/)

文章相似度 (http://www.pkudodo.com/tag/%e6%96%87%e7%ab%a0%e7%9b%b8%e4%bc%bc%e5%ba%a6/)

最大熵 (http://www.pkudodo.com/tag/%e6%9c%80%e5%a4%a7%e7%86%b5/)

朴素贝叶斯 (http://www.pkudodo.com/tag/%e6%9c%b4%e7%b4%a0%e8%b4%9d%e5%8f%b6%e6%96%af/)

机器学习 (http://www.pkudodo.com/tag/%e6%9c%ba%e5%99%a8%e5%ad%a6%e4%b9%a0/)

爬虫 (http://www.pkudodo.com/tag/%e7%88%ac%e8%99%ab/)

统计学习方法 (http://www.pkudodo.com/tag/%e7%bb%9f%e8%ae%a1%e5%ad%a6%e4%b9%a0%e6%96%b9%e

统计方法学习 (http://www.pkudodo.com/tag/%e7%bb%9f%e8%ae%a1%e6%96%b9%e6%b3%95%e5%ad%a6%e

综述 (http://www.pkudodo.com/tag/%e7%bb%bc%e8%bf%b0/)

网易云歌单 (http://www.pkudodo.com/tag/%e7%bd%91%e6%98%93%e4%ba%91%e6%ad%8c%e5%8d%95/)

规划 (http://www.pkudodo.com/tag/%e8%a7%84%e5%88%92/)

详解 (http://www.pkudodo.com/tag/%e8%af%a6%e8%a7%a3/)

逻辑回归 (http://www.pkudodo.com/tag/%e9%80%bb%e8%be%91%e5%9b%9e%e5%bd%92/)

逻辑斯蒂回归 (http://www.pkudodo.com/tag/%e9%80%bb%e8%be%91%e6%96%af%e8%92%82%e5%9b%9e%e

面试 (http://www.pkudodo.com/tag/%e9%9d%a2%e8%af%95/)