Когнитивный ассистент

February 10, 2020

1 Описание задачи

В данной работе мы описываем методику конструирования целеориентированного режима когнитивного ассистента (далее КА). Нас интересует построение интерактивной диалоговой системы, т.е. системы, способной анализировать запросы пользователя, задавать ему вопросы и правильно на них отвечать, которая помогает пользователю в построении его собственного плана достижения заранее (в самом начале диалога) обозначенной цели. А именно, цель пользователя является достаточно конкретной (напр., купить автомобиль), и КА должен в процессе диалога максимально конкретизировать цель (в случае автомобиля данной конкретезацией может быть выбор между отечественным автомобилем и иномаркой, между подержанным и новым и т.д.) и провести пользователя по всем этапам достижения цели давая рекомендации.

Мотивацией для создания новой модели поведения когнитивного ассистента является следующее. На данный момент диалоговые системы не показывают результатов, сравнимых с результатом общения с экспертом. Во-первых, текущие технологии не заточены на интерактивное общение с пользователем. А именно, существуют неплохие вопросно ответные системы [**?**] и есть результаты в создании диалоговых систем для общения на свободную тему (chit-chat) [**?**], однако практически не разработаны системы, задающие вопрос пользователю и обрабатывающие ответ для достижения собственных целей. Во-вторых, существующие диалоговые системы плохо заточены для работы в предметной области. Для решения этих задвч в частности используются базы знаний [**?**].

2 Модель ассистента

Определение 2.1. ***Графом когнитивного ассистента*** *называется ориентированный граф $G = (V, E)$, такой что выполнены следующие условия:*

1. Есть две вершины s и t, такие что $d_{in}(s) = d_{out}(t) = 0$, т.е. в s не входит не одного ребра, из t не выходит. Эти вершины называются стартовая и терминальная соответсвенно.

2. Каждой вершине $v \in V$ соответствует набор алгоритмов $(\mathcal{M}_v, \mathcal{G}_v, \mathcal{R}_v, \mathcal{I}_v, \mathcal{S}_v)$. Алгоритм $\mathcal{M}_v$ есть ***главный алгоритм****, алгоритм $\mathcal{G}_v$ есть* ***алгоритм генерации сообщения****, алгоритм $\mathcal{R}_v$ есть* ***алгоритм обработки запроса****, алгоритм $\mathcal{I}_v$ есть* ***алгоритм интерактивного общения****, алгоритм $\mathcal{S}_v$ есть textbfалгоритм принятия решения. Формальное описание будет дано ниже.*

Главный алгоритм $\mathcal{M}_v$ определяет порядок запуска остальных четырех алгоритмов.

Алгоритм генерации сообщения $\mathcal{G}_v$ есть алгоритм, который выводит единственное сообщение s пользователю. Это сообщение есть полезная информация, которую получит пользователь, в частности, этот алгоритм сообщит, что нужно сделать пользователю для достижения своей цели.

Алгоритм обработки запроса $\mathcal{R}_v$ есть алгоритм, который получает на вход сообщение пользователя, обрабатывает его и, если обнаруживает некоторый запрос, то начинает его обработку. Считается, что все возможные обрабатываемые запросы обработаны заранее и их конечное число. Наиболее важным запросом является запрос на аргументацию одного из ранее полученных сообщений, т.е. запрос на генерацию a, который убедит, что одно информация в полученном ранее сообщении поможет пользователю в достижении цели.

Алгоритм принятия решения $\mathcal{S}_v$ есть алгоритм, который используя алгоритм $\mathcal{I}_v$ и всю ранее полученную информацию, принимает решение по какому ребру пойти.

Алгоритм интерактивного общения $\mathcal{I}_v$ есть алгоритм, принимающий на вход три набора сообщений $(\{s_k^1\}, \{s_k^2\}, \{s_k^3\})$ - набор сообщений (или информации, извлеченных из них), полученных от пользователя в пройденных вершинах, набор сообщений (или информации, извлеченных из них), полученных от пользователя в текущей вершине, и набор сообщений, которые характеризуют текущие потребности когнитивного ассистента.

Данный алгоритм будет вызываться в $\mathcal{S}_v$ для пополнения информации о пользователе до такого уровня, что можно будет выбрать наилучшую для данного пользователя следующую вершину.

Существует четыре различных случая вершин и соответсвующих им алгоритмов:

1. Конечная вершина
2. Вершина с одним исходящим ребром
3. Вершина с более чем одним исходящим ребром
4. Стартовая вершина

Мы считаем, что если пользователь дошел до конечной вершины t, то он достиг своей цели. Поэтому в конечной вершине алгоритмы имеют простую структуру. А именно, $\mathcal{M}_t$ вызывает функцию $\mathcal{G}_t$, которая выводит некоторое прощальное сообщение и завершает работу. Остальные алгоритмы можно считать алгоритмами, которые завершают свою работу на любом входе за первый шаг, возвращая пустую строку (далее такие алгоритмы будем называть пустыми алгоритмами).

В случае вершины с одним исходящим ребром алгоритм $\mathcal{M}_v$ устроен следующим образом. Во-первых, он выводит сообщение при помощи $\mathcal{G}_v$. Затем вызывает алгоритм $\mathcal{S}_v$, который возвращает единственный возможный вариант. Алгоритм $\mathcal{I}_v$ есть пустой алгоритм. Алгоритм $\mathcal{R}_v$ в этом и в оставшихся случаях мы обсудим позже.

Алгоритм $\mathcal{M}_v$ в оставшихся пунктах устроен следующим образом. Сначала вызывается алгоритм $\mathcal{S}_v$, затем на основе принятого решения генерируется сообщение алгоритмом $\mathcal{G}_v$. Как принимается решение в $\mathcal{S}_v$ и как происходит общение через $\mathcal{I}_v$ мы обсудим после формализации пользователя.

3 Модель пользователя

Определение 3.1. *Пусть v - вершина, из которой выходит более одного ребра. Пусть $\Omega_v = \{z|(v,z) \in E\}$. Тогда Ω_v есть* ***множество решений****. Также считаем, что каждому элементу этого множества соответсвует*

1. некоторый набор признаков $\boldsymbol{x}_v \in X_v$

2. функция соответсвия $g_v : X_v \times F \to R_+$, где F - особенности пользователя (см. опреление 3.2).

Без ограничения общности здесь и далее мы будем считать, что $X_v \subset$

$\mathbb{R}^n$. Действительно, какими бы не были признаки (бинарные, категориальные и т.д.) существуют способы приведения их к такому виду.

Определение 3.2. ***Моделью пользователя*** *мы будем называть тройку* $u = (x_-, x_+, f)$, *где*

1. $\boldsymbol{x}_-, \boldsymbol{x}_+ \in \mathbb{R}^N, N = \sum\limits_{v:d_{out}(v)>1} \dim X_v$ *- некоторые ограничения, которые пользователь ставит на варианты решений для* Ω_v, *т.е. пользователь хочет, чтобы* $\boldsymbol{x}_-|_v \leq \boldsymbol{x}_v^* \leq \boldsymbol{x}_+|_v$. *Здесь* $\boldsymbol{x}_v^*$ *- решение, которое КА примет в вершине* v, $\boldsymbol{x}_\pm|_v$ *- та часть вектора* $\boldsymbol{x}_\pm$, *которая соответсвует вершине* v.

2. $\boldsymbol{f} \in F$ *- некоторые особенности пользователя.*

Сделаем несколько пояснений к этому определению.

Во-первых, пользователь может не знать, что означает k-ый признак варианта в вершине v или он ему может быть не интересен. В обоих этих случаях мы будем считать $x_\pm|_v^k = \pm\infty$.

Во-вторых, обсудим особенности пользователя F. Это некоторые характеристики пользователя, которые не связаны явно с характеристиками решений, однако можно установить связь, насколько решение соответствует этим характеристикам при помощи функции соответсвия g_v. Примерами таких характеристик могут быть следующие:

1. В случае если КА помогает купить пользователю некоторый предмет, то таковой характеристикой может быть то, зачем пользователю этот предмет.

2. В случае если покупка нуждается в регулярном обслуживании, которое стоит денег, то таковой характетристикой может быть заработок пользователя и/или сколько он готов тратить на это.

Заметим, что изначально КА не знает $\mathbf{u}$, и он может узнать его компоненты или получить оценку на них, используя алгоритм $\mathcal{I}_v$. Однако далее мы будем требовать, чтобы алгоритм не пытался узнать весь вектор $\mathbf{u}$, когда это необходимо. Это позволит уменьшить количество запросов к пользователю, а, следовательно, КА станет удобнее для использования.

Далее мы также будем считать, что $F \subset \mathbb{R}^d_+$.

4 Алгоритм принятия решения

Целью данного раздела является описание модели принятия решения. Рассмотрим некоторую вершину v со степенью $d_{\text{out}}(v) > 1$. Пусть X_v

- множество признаков решений в этой вершине, $g_v : X_v \times F \to R_+$ - функция соответсвия.

4.1 Постановка задачи оптимизации

КА общается с пользователем с параметрами $\mathbf{u} = (\mathbf{x}_-, \mathbf{x}_+, \mathbf{f}) \in U$. Определим $\mathbf{u}|_v = (\mathbf{x}_-|_v, \mathbf{x}_+|_v, \mathbf{f}) \in U|_v$, т.е. отбросим все признаки, которые не влияют на принятие решения в данной вершине. В данном разделе, мы везде будем использовать $\mathbf{u}|_v$,

Мы будем считать, что КА всегда узнает первыми запросами ограничения, требуемые пользователем, $(\mathbf{x}_-, \mathbf{x}_+)$. Если пользователю понадобилась помощь КА, то весьма вероятно, что он не имеет достаточно много информации об этих значениях. Поэтому можно считать, что этот ответ не потребует от него много сил.

Определим целевую функцию $s_v : X_v \times U|_v \to \mathbb{R}$:

$$s_{v,\mathbf{u}}(\mathbf{x}) = g_v(\mathbf{x}, \mathbf{f}) + \sum_{k=1}^{n} \lambda_k^- L(\mathbf{x}_-|_v^k, x_k) + \sum_{k=1}^{n} \lambda_k^+ G(\mathbf{x}_+|_v^k, x_k), \qquad (1)$$

где константы $\lambda_k^\pm \geq 0$, функции L и G есть барьерные функции, которые штрафуют за нарушение ограничений. Примерами таких функций могут быть индикаторные функции или их сглаженные аналоги.

Получается , что есть пользователь, с которым общается КА, и у этого пользователя есть некоторый вектор $\mathbf{u}$. КА, находясь в вершине v, может не знать ничего о векторе $\mathbf{u}$, однако, общаясь с пользователем при помощи алгоритма $\mathcal{I}_v$, может узнавать эти признаки. В таком случае, задача алгоритма $\mathcal{S}_v$ найти максимальное или близкое к нему значение функии $s_{v,\mathbf{u}}(\mathbf{x}) = s_v(\mathbf{x}, \mathbf{u})$ при фиксированном u на множестве $\{x_z | z \in \Omega_v\}$ при минимальном вызове алгоритма общения $\mathcal{I}_v$.

Задача максимизации s_v при заданном v является задачей дискретной оптимизации и в общем случае, как известно, является $\mathcal{NP}$-трудной. При "небольшом" размере множества вариантов Ω_v мы можем решать эту задачу при любом v перебором за разумное время. Случай, когда это не выполняется, будет рассмотрен в разделе 4.3. Далее мы будем считать, что есть эффективный метод, который решает задачу:

$$s_{v,\mathbf{u}}(\mathbf{x}) \to \max_{\mathbf{x} \in \{\mathbf{x} | z \in \Omega_v\}}. \qquad (2)$$

Заметим, что в общем случае мы не можем решить эту задачу, не зная значения $\mathbf{u}$. Примеры функций, при которых это возможно, а также метод оптимального задавания вопросов будут приведены ниже.

4.2 Алгоритм интерактивного общения

В данной секции мы хотим разработать метод генерирования запросов к пользователю, такой что количество этих запросов для уверенного нахождения наилучшего решения z из множества Ω_v было минимально.

Определение 4.1. ***Множество состояний системы*** *мы назовем следующий элемент*

$$\mathsf{P} = \left\{ \boldsymbol{x} \middle| x_k \in F_n \cup \{\perp\} \right\} \cup \{\mathbf{p_f}\},$$

где $\mathbf{p_f}$ *некоторый объект не из множества* $\left\{ \boldsymbol{x} \middle| x_k \in F_n \cup \{\perp\} \right\}$. *Элемент этого множества назовем состоянием системы* $\mathbf{p}$.

Определим далее:

$$F(\mathbf{p}) = \{\mathbf{f} \in F \Big| f_k = p_k \bigvee p_k = \perp\},$$

т.е. множество всех элементов F, значения компонент которых совпадает с известными компонентами $\mathbf{p}$.

Теперь пусть у нас имеется некоторая функция информативности H состояния p. Эта функция зависит от выбранной модели $s_{v,\mathbf{u}}$. Определим функцию информативности $H : \mathsf{P} \to \mathbb{R}$. Для этого нам понадобятся вспомогательные функии:

$$M(\mathbf{p}, z) = \max_{\mathbf{u} \in \{\mathbf{u} | \mathbf{f} \in F(\mathbf{p})\}} s_{v,\mathbf{u}}(\mathbf{x}_z), z \in \Omega_v$$

$$m(\mathbf{p}, z) = \min_{\mathbf{u} \in \{\mathbf{u} | \mathbf{f} \in F(\mathbf{p})\}} s_{v,\mathbf{u}}(\mathbf{x}_z), z \in \Omega_v$$

В силу того, что мы хотим решить задачу максимизации 1, то критерием информативности будет то, насколько уверенно мы можем сказать, что максимум этой функции при не полностью определенном $\mathbf{u}$ близок к максимуму при полностью определенном. Поэтому функция H зависит от модели выбора решения. Рассмотрим несколько вариантов:

1. Минимаксный подход. $z_1 = \arg\max_{z\in\Omega_v} m(p,z)$ (в случае, если таких значений несколько, то выберем с максимальным значением $M(p,z)$). В таком случае, функцией информативности будет являться:

$$H(p) = m(p,z_1) - \max_{z\in\Omega_v\setminus z_1} M(p,z),$$

т.е. функция информативности показывает, насколько значение целевого функционала могло бы увеличиться, если бы уточнить информацию.

2. Вероятностный подход. Если ввести вероятностное пространство относительно множества элементарных исходов $F(\mathbf{p})$ и предположить, что многомерная случайная величина $\|s_{v,\mathbf{u}}(z)\|_{z_k}$ имеет известное распределение f на $[m(\mathbf{p},z), M(\mathbf{p},z)]$, то можно использовать вероятностные соображения для определения функции информативности. Пусть z_1 есть элемент из Ω_v, с максимальным матожиданием и минимальным размером носителя распределения. Пусть z_2 есть элемент из $\Omega_v\setminus\{z_1\}$, выбранный по тем же правилам. Введем обозначения, $m_k = m(\mathbf{p},z_k), M_k = M(\mathbf{p},z_k)$ и определим функцию информативности как, то что z_1 оптимальнее z_2 не менее, чем на ϵ по значению функции:

$$H(\mathbf{p}) = \mathbb{P}\left(s_{v,\mathbf{u}}(z_2) < s_{v,\mathbf{u}}(z_1) + \epsilon\right)$$

$$H(\mathbf{p}) = \int\limits_{\max(m_1, m_2-\epsilon)}^{M_1} ds_1 \int\limits_{m_2}^{\min(M_2, s_1+\epsilon)} f_{z_1,z_2}(s_1,s_2)ds_2$$

В частности, для равномерного распределения получаем:

...

Также определим функцию штрафа γ за то, что для состояния $\mathbf{p}$. Эта функция должна выражать наше желание получить информацию за минимальное количество запросов. В таком случае можно считать, что у нас задана функция $\gamma(|\mathbf{p}|)$, не убывающая с ростом $\mathbf{p}$. Тогда награждение для состояния $\mathbf{p}$ мы определим, как

$$r(\mathbf{p}) = H(\mathbf{p}) - \gamma(|\mathbf{p}|)$$

$$r(\mathbf{p}_f) = 0$$

Также введем множество действий $A = \{a_j | j = \overline{1, \dim F}\} \cup \{a_f\}$. Действие a_f переводит любое состояние $\mathbf{p}$ в финальное состояние $\mathbf{p}_f$ и далее он

не будет обсуждаться. Каждый элемент этого множества позволяет перейти из состояния $\mathbf{p}$ в одно состояние из множества

$$a_j(\mathbf{p}) = \begin{cases} \left\{ \mathbf{p}' \middle| \mathbf{p} \backslash \mathbf{p}' = (j, y) \right\}, \text{ если } (j, y) \notin \mathbf{p}, \forall y \\ \text{пустое множество, иначе.} \end{cases} \tag{3}$$

Введем также модель переходов между состояниями:

$$T(\mathbf{p}, a, \mathbf{p}') = \begin{cases} 0, \text{ если } \mathbb{P}(a(\mathbf{p}) = \mathbf{p}') = 0 \\ f(\mathbf{p}'|\mathbf{p}), \text{ иначе} \end{cases} \tag{4}$$

Теперь мы можем сформлурировать задачу обучения с подкреплением. Нашей моделью будет $(\mathsf{P}, p_0, A, T, R)$, где $p_0 = \|\perp\|_k$. При этом любое состояние может быть как финитным, так и не финитным.

Заметим, что множество $\mathbf{F}$, а, следовательно, и множество состояний S не обязаны быть конечными. В то же время множество действий A в сформулированной задаче всегда конечно, и $|A| = \dim F$.

4.3 Ограничения вычислительной мощности

...

4.4 Целевая функция

...

5 Оценка качества

...

6 КА для задачи покупки автомобиля

...

7 Заключение

...