Московский Государственный Университет
им. М.В. Ломоносова

Применение Скрытой Марковской Модели для извлечения информации из пользовательского запроса в приложении интеллектуального персонального помощника

Отчёт по спецкурсу “Математические методы анализа текстов”

Студенты:
Николай Иванов, Ирина Броварь
Группа: 425
Факультет: ВМК
Преподаватель:
Dr Мстислав Масленников

Москва, май 2013 г.

1. Постановка задачи

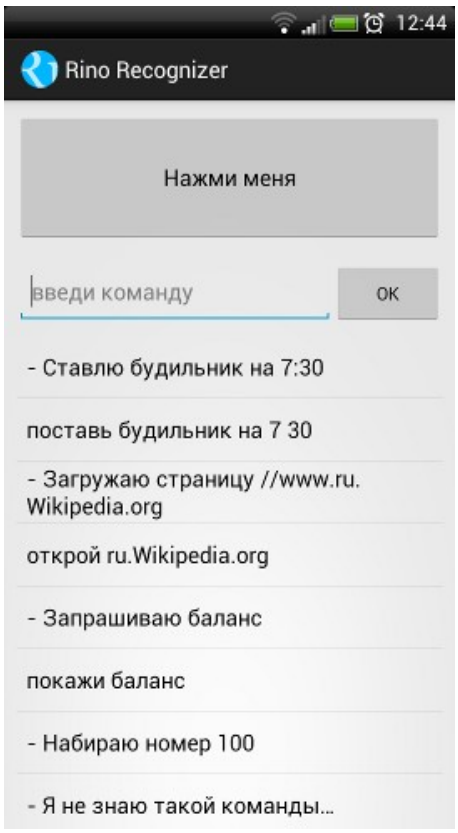
Вводная часть

Интеллектуальные персональные помощники – это приложения для мобильных устройств, основная задачей которых является работа с речевыми командами пользователя.

На начальном этапе работы приложения производится разпознавание речевой команды с целью получения ее текстового представления. Таким образом, после распознавания пользовательская команда записывается в виде последовательности слов, причем:

- в качестве разделителя слов используются только пробелы
- слова могут состоят как из латинских букв, так и из кириллицы
- все буквы строчные
- знаки препинаяния отсутствуют (если не считать те случаи, когда знаки являются частью распознаваемых слов, как, например, точка в слове “yandex.ru”)

Далее интеллектуальному помощнику нужно понять, какое действие хотел выполнить пользователь, и после этого произвести данное действие, например, запустить определенную программу, совершить звонок и т.п.



Приложение интеллектуального помощника, для которого мы хотим применить данный подход, в настоящий момент может выполнять следущие действия:

1. телефонный звонок (на определенный номер или абоненту из телефонной книги)
2. отпрака смс-сообщений
3. отправка электронных писем
4. загрузка web-страниц по предоставленному адресу
5. проверка баланса
6. установка будильника на определенное время

Для анализа пользовательской команды и извлечения из нее необходимой информации было предложено использовать Скрытую Марковскую Модель.

С этой целью было выделено 8 смысловых групп, на которые можно разбить все слова пользовательской команды:

1. command
2. name
3. tel
4. email
5. url
6. time
7. text
8. other

Список 1. Семантические группы слов

Таким образом, исходная постановка задачи извлечения информации из пользовательского запроса может быть формализована следующими образом:

> Применяя Скрытую Марковскую модель необходимо разметить исходную последовательность слов метками из Списка 1.

Гипотеза:

1.1 Примеры разметки:

позвони по телефону 8 917 123 45 67
позвони ире
набери номер 8 917 123 45 67
набери *100#
набери звездочка сто решетка
отправь ире смску с текстом привет как дела
проверь баланс
открой yandex.ru
открой страницу yandex.ru
отправь письмо с текстом привет как дела по адресу example@gmail.com
отправь смс привет как дела на номер 8 917 123 45 67
поставь будильник на 7 30

В этих примерах цвет слова соответствует его смысловой группе.

Таким образом, последовательность слов

позвони по телефону 89171234567

соответствует следующему списку токенов:

(позвони, command), (по), (телефону), (89171234567, tel).

2. Описание скрытой марковской модели

Скрытые состояния

Состояния классификатора СММ соответствуют меткам, которыми должна быть размечена исходная последовательность:

1. sCommand
2. sName
3. sTel
4. sEmail
5. sUrl
6. sTime
7. sText
8. sOther

2.1 Наблюдения

Скрытому состоянию соответствуют возможные наблюдения, которые описываются вектором признаков.

Свойством может быть бинарное значение, показывающее наличие/отсутствие слова в определенном наборе.

Набор описывает семантическую группу:

1. команды
2. имена
3. телефоны
4. email’ы
5. url’ы
6. время

Для определения, входит ли слово в определенный набор или нет, используются регулярные выражения:

```
command   (\w*звон\w+)|(набрать\w+)|(набер\w+) ...
```

```
name      (леш\w*)|(ван\w*)|(саш\w*)|(наст\w*) ...
tel       (\+?[\d\s]+)
email     ([^S]+@[^S]+)
url       ((\w+.)+(ru|com|org|net|su))
time      (\d+)
```

Пример наблюдения
Пусть слово распозналось как “позвони”,
тогда наблюдением для данного слова будет являться следующий вектор признаков
`(1,0,0,0,0,0,0)`
поскольку позвони входит только в набор “command”.

Замечание 1
Множество наборов почти совпадает с множеством состояний, и тем не менее это разные понятия.
Если распознанное слово встречается в наборе X, это говорит о том, что возможно данному слову соответствует скрытое состояние X, но это не обязательно так.

Рассмотрим пример:
`отправь смс васе с текстом перезвони`

Слово “перезвони” входит в набор “command”, однако этому слову должно соответствовать скрытое состояние “sText”, а не “sCommand”:
`отправь смс васе с текстом перезвони`

2.2 Последовательности наблюдений

Каждой распознанной строке ставится в соответствие последовательность наблюдений.

Пример последовательности наблюдений
Пусть после распознавания на вход анализатору пришла строка:
`отправь смс васе с текстом перезвони`

Вектор признаков для каждого слова:

```
отправь       (1,0,0,0,0,0,0)
```

```
смс           (1,0,0,0,0,0,0)
васе          (0,1,0,0,0,0,0)
с             (0,0,0,0,0,0,0)
текстом       (0,0,0,0,0,0,0)
перезвони     (1,0,0,0,0,0,0)
```

Таким образом, всей строке будет поставлен в соответствие следующий список векторов:

```
((1,0,0,0,0,0,0), (1,0,0,0,0,0,0), (0,1,0,0,0,0,0),
(0,0,0,0,0,0,0), (0,0,0,0,0,0,0), (0,0,0,0,0,0,0))
```

3. Классификатор

Далее описание идет с учетом применения библиотеки jahmm: jahmm.googlecode.com

3.1 Инициализация классификатора

Проанализировав имеющиеся размеченные последовательности слов, нужно приблизительно определить следующие начальные параметры классификатора:

1. Pi (k) - initial probability,
   т.е. вероятность нахождения в состоянии k

2. Opdf (k, a) - observation probability distribution function
   вероятность наблюдения признака a в состоянии k

3. A(i,j) - матрица вероятностей переходов из состояния i в состояние j.

Далее нужно создать объект классификатора hmm и инициализировать его полученными значениями.

3.2 Обучение классификатора

1. считать наблюдения для размеченных последовательностей слов из файла в переменную sequences
2. создать объект bwl, которые будет применять к модели алгоритм Баума-Велча для итеративного улучшения модели hmm
3. вызвать метод bwl.learn(hmm, sequences) для запуска процесса обучения

Для обучения использовался корпус Caller из вручную размеченных примеров. Всего размеченных команд – 97 (от 2 до 9 слов в каждой команде). Подкорпус для обучения CallerTrain состоял из 49 примеров, а подкорпус для тестирования CallerTest из 48 примеров.

Фрагмент файла с размеченными командами:

```
# позвони на номер 89171234567
[ 1 0 0 0 0 0 ] ; [ 0 0 0 0 0 0 ] ; [ 1 0 0 0 0 0 ] ; [ 0 0 1 0 0 1 ] ;

# набери +79171234567
[ 1 0 0 0 0 0 ] ; [ 0 0 1 0 0 0 ] ;

# набери номер +79171234567
[ 1 0 0 0 0 0 ] ; [ 1 0 0 0 0 0 ] ; [ 0 0 1 0 0 0 ] ;

# набери мне номер 89171234567
[ 1 0 0 0 0 0 ] ; [ 0 0 0 0 0 0 ] ; [ 1 0 0 0 0 0 ] ; [ 0 0 1 0 0 0 ] ;

# открой yandex.ru
[ 1 0 0 0 0 0 ] ; [ 0 0 0 0 1 0 ] ;

# открой в браузере yandex.ru
[ 1 0 0 0 0 0 ] ; [ 0 0 0 0 0 0 ] ; [ 0 0 0 0 0 0 ] ; [ 0 0 0 0 1 0 ] ;

# отправь васе сообщение
[ 1 0 0 0 0 0 ] ; [ 0 1 0 0 0 0 ] ; [ 1 0 0 0 0 0 ] ;

…
```

3.3 Применение классификатора

1. строка, полученная после распознавания, представляется в виде последовательности лексем
2. для каждой лексемы вычисляется вектор признаков
3. писок векторов признаков s передается методу int[] mostLikelyStateSequence(s) для получения наиболее вероятность последовательности скрытых состояний
4. разметить исходную последовательность лексем

Эксперименты проводились в нескольких конфигурациях:
(1) Раздельное обучение и тестирование с помощью программы jahmm. Программа запустилась, однако при работе программа осуществляет внутренние преобразования матриц. При этом работа программы состоит из нескольких итераций алгоритма Baum-Welch, при которых матрицы раскладываются по методу Холецкого. На входе алгоритма Холецкого матрицы должны быть положительно определёны. Однако, свойство

положительной определённости теряется через 5-6 итераций и программа заканчивает работу с ошибкой.

(2) Раздельное обучение и тестирование с помощью программы Mallet. Программа преобразует слова во внутренний словарь, и извлекает свойства слова на основе внутреннего словаря. При встрече незнакомого слова на корпусе для тестирования программа завершает работу с ошибкой.

(3) Тестирование и обучение с помощью программы Mallet на одинаковом корпусе. Были получены следующие результаты:

	P	**R**	**F1**
command	0.78	0.95	**0.86**
name	0.60	0.92	0.72
other	0.89	0.78	**0.83**
tel	0.54	1.00	0.70
url	0.50	0.31	0.38
time	0.17	0.10	0.13

Наилучший результат был получен для типов command и other. Это объясняется тем, что слово из группы command в большинстве примеров находится на первой позиции.