8 Распознавание намерений

Чат-бот должен быть настоящим интеллектуалом. Например, диалоговый чат-бот должен уметь распознавать намерение пользователя, чтобы поддерживать с ним разговор, чат-бот, предназначенный для заказа еды, должен понимать требования пользователя, чтобы принять заказ. И хотя задача распознавания намерений уже рассматривалась в предыдущих главах, в этой главе мы обсудим ее более подробно.

Начнем с распознавания намерений пользователя с помощью выделения переходного глагола и прямого дополнения высказывания. Далее обсудим, как выяснить намерение пользователя по последовательности предложений, как распознать синонимы, выражающие различные возможные намерения, и определить намерение пользователя на основе семантического подобия.

Распознавание намерений с помощью выделения переходного глагола и прямого дополнения

Обычно распознавание намерения пользователя состоит из трех этапов: синтаксического разбора предложения на токены, соединения токенов маркированными дугами, отражающими синтаксические отношения, и прохода по этим дугам для выделения соответствующих токенов. Но во многих случаях для распознавания намерений пользователя достаточно выделить переходный глагол и прямое дополнение, как видно из разбора синтаксических зависимостей на рис. 8.1.

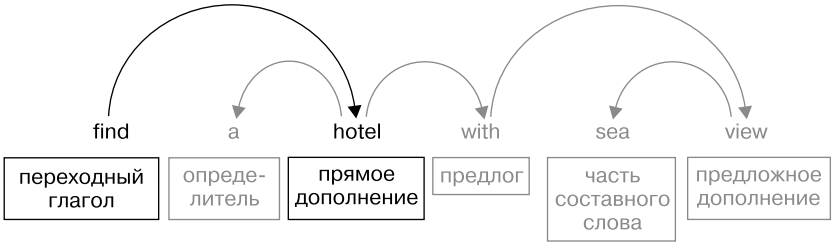

Рис. 8.1. Пример визуального представления синтаксической структуры предложения

Дуга, соединяющая переходный глагол с прямым дополнением, указывает, что пользователь намерен найти гостиницу (если объединить переходный глагол и прямое дополнение в одно слово, получится `findHotel`). В последующем коде эту структуру можно использовать в качестве *идентификатора намерения* (intent identifier), как показано ниже:

```
intent = extract_intent(doc)
if intent == 'orderPizza':
  print('We need you to answer some questions to place your order.')
  ...
elif intent == 'showPizza':
  print('Would you like to look at our menu?')
  ...
```

ПРИМЕЧАНИЕ

В главе 11 мы рассмотрим более детальные примеры использования идентификаторов намерения в коде приложения для чат-бота.

Иногда определить смысл пары «переходный глагол/прямое дополнение» не так и просто. Чтобы найти глагол и существительное, лучше всего описывающие намерение, приходится изучать синтаксические отношения переходного глагола и прямого дополнения.

В других случаях бывает, что пользователь не выражает свое намерение явным образом, так что приходится вычислять его предполагаемые намерения. В этом разделе обсудим стратегии выделения намерений с помощью структуры синтаксических зависимостей.

Получение пары «переходный глагол/прямое дополнение»

Начнем с выделения из предложения пары «переходный глагол/прямое дополнение» путем поиска для всех токенов меток зависимости, равных `dobj`. Найдем прямое дополнение и легко получим соответствующий переходный глагол в виде синтаксического главного элемента этого прямого дополнения, как показано в следующем сценарии:

```
import spacy
nlp = spacy.load('en')
❶ doc = nlp(u'show me the best hotel in berlin')
❷ for token in doc:
    if token.dep_ == 'dobj':
      print(❸token.head.text + token.text.capitalize())
```

В данном сценарии применяем конвейер к образцу предложения ❶, после чего проходим в цикле по токенам в поисках токена с меткой зависимости `dobj` ❷. Найдя его, определяем соответствующий переходный глагол как главный элемент прямого дополнения ❸. В этом примере также объединяем переходный глагол и его прямое дополнение, чтобы выразить намерение одним словом.

Результаты работы сценария выглядят так:

```
showHotel
```

Учтите, что не все предложения, содержащие пару «переходный глагол/прямое дополнение», выражают какое-либо намерение. Например, предложение *He gave me a book* просто констатирует факт. Подобные предложения можно отфильтровать путем проверки характеристик глагола и отбора только тех предложений, в которых есть глаголы настоящего времени и не третьего лица. Впрочем, такие предложения — редкость при общении пользователя с чат-ботом, принимающим заказы.

Выделение множественных намерений с помощью token.conjuncts

Иногда встречаются предложения, выражающие несколько намерений. Например, такое:

```
I want a pizza and cola.
```

В данной ситуации пользователь хочет заказать пиццу и кока-колу. Однако в большинстве случаев эти намерения можно считать частью одного составного намерения. Хотя пользователь хочет заказать разнотипные товары, подобное предложение обычно считают одним заказом с несколькими позициями. В этом примере можно распознать намерение, состоящее из пары «переходный глагол/прямое дополнение» `orderPizza`, но при этом из него можно выделить `pizza` и `cola` в качестве отдельных позиций размещаемого заказа.

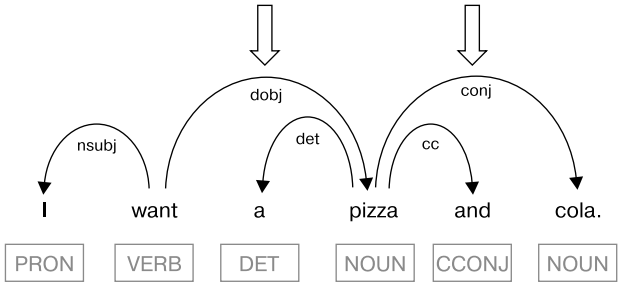

Рис. 8.2. Дерево зависимостей предложения, содержащего прямое дополнение и его конъюнкт

На схеме есть две стрелки, указывающие на дуги прямого дополнения *pizza* и его конъюнкта *cola*. *Конъюнкт* (conjunct) существительного — это другое существительное, присоединенное к первому посредством союза, такого как «и», «или» и т. д. Для выделения прямого дополнения и его конъюнкта можно воспользоваться следующим кодом:

```
doc = nlp(u'I want a pizza and cola.')
# Выделяем прямое дополнение и его конъюнкт
for token in doc:
  if token.dep_ == 'dobj':
    dobj = [token.text]
    conj = [t.text for t in ❶token.conjuncts]
# Составляем список выделенных элементов
❷ dobj_conj = dobj + conj
print(dobj_conj)
```

Выделяем конъюнкт прямого дополнения с помощью свойства `conjuncts` объекта `Token`, соответствующего этому прямому дополнению ❶. После получения прямого дополнения и его конъюнктов объединяем их в один список ❷.

Результат работы сценария выглядит так:

```
['pizza', 'cola']
```

Чтобы сформулировать намерение, необходимо выделить и глагол. Простейший способ его получения, при уже выделенном прямом дополнении, — найти синтаксический главный элемент этого прямого дополнения (с подобным примером вы уже встречались в подразделе «Получение пары “переходный глагол/прямое дополнение”» на с. 162):

```
verb = dobj.head
```

Далее, с помощью свойства `text` глагола и прямого дополнения можно составить идентификатор намерения.

Попробуйте сами

В приведенном на с. 163 сценарии вы получили доступ к конъюнкту прямого дополнения через свойство `conjuncts` объекта `Token`. В новом сценарии замените эту строку кодом, который бы извлекал конъюнкт в процессе поиска дуги с меткой `conj` при обходе наружу от прямого дополнения. Сделать это можно внутри цикла, в котором ищется прямое дополнение посредством поиска дуги с меткой `dobj`. Не забудьте убедиться, что главный элемент дуги `conj` соответствует прямому дополнению.

Выделение намерения с помощью списков слов

В некоторых случаях намерение пользователя лучше всего описывают не переходный глагол и прямое дополнение, а другие токены. Обычно они связаны отношением с переходным глаголом или прямым дополнением: просто нужно будет сделать еще один шаг и найти слова, лучше всего описывающие намерение, путем исследования синтаксических отношений переходного глагола и прямого дополнения.

В качестве примера рассмотрим следующее высказывание:

```
I want to place an order for a pizza.
```

В этом предложении намерение лучше всего описывают слова *want* и *pizza*, хотя ни одно из них не является переходным глаголом или прямым дополнением. Впрочем, из дерева зависимостей данного высказывания видно, что *want* и *pizza* относятся к переходному глаголу *place* и прямому дополнению *order* соответственно. Упомянутое дерево зависимостей приведено на рис. 8.3.

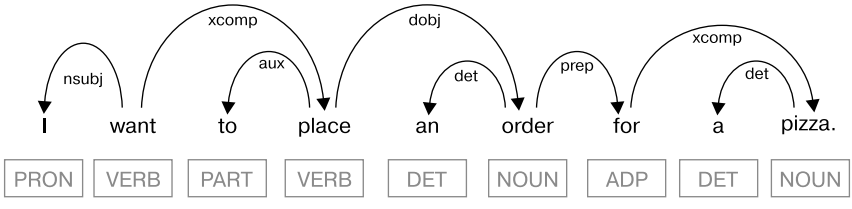

Рис. 8.3. Дерево зависимостей высказывания, переходный глагол и прямое дополнение которого не передают намерение пользователя

Для выделения слов *want* и *pizza* из высказывания воспользуемся списком заранее определенных слов и найдем их в высказывании.

Опытный программист может усомниться в эффективности «жесткого» прописывания в коде подобного списка, ведь есть вероятность, что список окажется очень длинным, особенно для использования в ряде различных контекстов. Но предназначенный для конкретной задачи, например заказа пиццы, список, скорее всего, будет на удивление коротким, так что такой подход довольно эффективен. Он реализован в следующем коде:

```
# Применяем конвейер к образцу предложения
doc = nlp(u'I want to place an order for a pizza.')
# Выделяем прямое дополнение и его переходный глагол
dobj = ''
tverb = ''
for token in doc:
❶ if token.dep_ == 'dobj':
    dobj = token
    tverb = token.head
# Выделяем глагол для описания намерения
intentVerb = ''
verbList = ['want', 'like', 'need', 'order']
❷ if tverb.text in verbList:
  intentVerb = tverb
❸ else:
```

```
    if tverb.head.dep_ == 'ROOT':
      intentVerb = tverb.head
  # Выделяем дополнение для описания намерения
  intentObj = ''
  objList = ['pizza', 'cola']
❹ if dobj.text in objList:
    intentObj = dobj
  else:
    for child in dobj.children:
      if child.dep_ == 'prep':
        ❺ intentObj = list(child.children)[0]
          break
    ❻ elif child.dep_ == 'compound':
          intentObj = child
          break
  # Выводим в консоль высказанное в образце предложения намерение
  print(intentVerb.text + intentObj.text.capitalize())
```

Как всегда, начинаем с поиска и выделения прямого дополнения и его переходного глагола ❶. Затем проверяем, присутствуют ли они в соответствующем списке предопределенных слов. Здесь, конечно, используем упрощенные списки: `verbList` содержит глаголы, которыми может воспользоваться покупатель при размещении заказа, а `objList` содержит прямые дополнения — возможные позиции меню. Начинаем с проверки переходного глагола ❷. Если его нет в списке допустимых глаголов ❸, проверяем смысловой глагол (`ROOT`) предложения, играющий роль главного элемента для переходного глагола. Эта реализация работает, даже если переходный глагол является смысловым глаголом предложения, поскольку главным элементом для смыслового глагола (`ROOT`) является он сам.

Далее обращаемся к прямому дополнению ❹. Если оно не входит в список допустимых слов, проверяем его синтаксические дочерние элементы. Начинаем с предлога этого прямого дополнения. Если таковой существует, берем его дочерний элемент (он у него может быть только один) ❺, который и будет всегда предложным дополнением.

Чтобы данный подход можно было применять к широкому спектру случаев, недостаточно только искать предлоги среди дочерних элементов прямого дополнения. Такая логика не годится, например, для следующего высказывания: *I want to place a pizza order*. Здесь отсутствует ветвь дочернего элемента-предлога. Вместо него у прямого до-