

Учебник по обработке естественного языка или NLP (Natural Language Processing)

Источник: <https://isolution.pro/ru/t/natural-language-processing?alias=ucebник-po-obrabotke-estestvennogo-azyka>

Язык - это способ общения, с помощью которого мы можем говорить, читать и писать. Обработка естественного языка (NLP) - это подраздел компьютерных наук, который занимается искусственным интеллектом (AI), который позволяет компьютерам понимать и обрабатывать человеческий язык.

Это учебное пособие предназначено для выпускников, аспирантов и студентов-исследователей, которые либо проявляют интерес к этому предмету, либо имеют этот предмет как часть своей учебной программы. Читатель может быть как новичком, так и продвинутым учеником.

Читатель должен иметь базовые знания об искусственном интеллекте. Он / она также должен знать основные термины, используемые в грамматике английского языка, и концепции программирования Python.

Введение

Язык - это способ общения, с помощью которого мы можем говорить, читать и писать. Например, мы думаем, принимаем решения, планы и многое другое на естественном языке; именно на словах. Однако большой вопрос, который стоит перед нами в эпоху искусственного интеллекта, заключается в том, можем ли мы таким же образом общаться с компьютерами. Другими словами, могут ли люди общаться с компьютерами на своем естественном языке? Для нас сложно разрабатывать приложения NLP, потому что компьютерам нужны структурированные данные, а человеческая речь неструктурирована и часто неоднозначна по своей природе.

В этом смысле мы можем сказать, что обработка естественного языка (NLP) - это подраздел компьютерных наук, особенно искусственный интеллект (AI), который заботится о том, чтобы компьютеры могли понимать и обрабатывать человеческий язык. Технически, основная задача NLP – программировать компьютеры для анализа и обработки огромного количества данных на естественном языке.

История NLP

Мы разделили историю NLP на четыре этапа. У фаз есть отличительные особенности и стили.

Первый этап (этап машинного перевода) - конец 1940-х - конец 1960-х гг.

Работа, сделанная на этом этапе, была сосредоточена в основном на машинном переводе (МП). Этот этап был периодом энтузиазма и оптимизма.

Давайте теперь посмотрим все, что было в первой фазе.

- Исследования NLP начались в начале 1950-х годов после расследования Бута и Риченса и меморандума Уивера по машинному переводу в 1949 году.
- 1954 был годом, когда ограниченный эксперимент по автоматическому переводу с русского на английский был продемонстрирован в эксперименте Джорджтаун-IBM.
- В этом же году началось издание журнала MT (Машинный перевод).
- Первая международная конференция по машинному переводу (МП) состоялась в 1952 году, а вторая - в 1956 году.
- В 1961 году работа, представленная на Международной конференции в Теддингтоне по машинному переводу языков и анализу прикладного языка, была кульминацией этого этапа.

Вторая фаза (фаза влияния ИИ) - конец 1960-х - конец 1970-х гг.

На этом этапе сделанная работа в основном была связана с мировым знанием и его ролью в конструировании смысловых представлений и манипулировании ими. Вот почему эта фаза также называется фазой со вкусом AI.

В фазе было следующее:

- В начале 1961 года началась работа над проблемами обращения и построения базы данных или знаний. На эту работу оказал влияние AI.
- В том же году была разработана вопросно-ответная система BASEBALL. Вход в эту систему был ограничен, а языковая обработка была простой.
- Очень продвинутая система была описана Минским (1968). Эта система, по сравнению с системой ответов на вопросы BASEBALL, была признана и обеспечена необходимостью вывода на основе базы знаний при интерпретации и ответе на языковой ввод.

Третья фаза (грамматико-логическая) - конец 1970-х - конец 1980-х гг.

Этот этап можно охарактеризовать как грамматико-логический этап. Из-за провала практического построения системы на последнем этапе исследователи перешли к использованию логики для представления знаний и рассуждений в ИИ.

В третьей фазе было следующее:

- К концу десятилетия грамматико-логический подход помог нам с мощными универсальными обработчиками предложений, такими как Core Language Engine и Discourse Presentation Theory от SRI, которые предложили средства для решения более широкого дискурса.
- На этом этапе у нас есть некоторые практические ресурсы и инструменты, такие как парсеры, например, Alvey Natural Language Tools, а также более оперативные и коммерческие системы, например, для запросов к базе данных.
- Работа над лексикой в 1980-е гг. Также указала в сторону грамматико-логического подхода.

Четвертая фаза (лексическая и корпусная фаза) - 1990-е годы

Мы можем описать это как фазу лексики и корпуса. Эта фаза имела лексический подход к грамматике, который появился в конце 1980-х и стал оказывать все большее влияние. В этом десятилетии произошла революция в обработке естественного языка с появлением алгоритмов машинного обучения для языковой обработки.

Изучение человеческих языков

Язык - важнейший компонент человеческой жизни, а также самый фундаментальный аспект нашего поведения. Мы можем переживать это в основном в двух формах - письменной и устной. В письменной форме это способ передать наши знания от одного поколения к другому. В устной форме это основная среда, с помощью которой люди координируют друг с другом свое повседневное поведение. Язык изучается по различным академическим дисциплинам. Каждая дисциплина имеет свой собственный набор проблем и набор решений для их решения.

Рассмотрим следующую таблицу, чтобы понять это

Дисциплина	Проблемы	инструменты
Лингвисты	Как можно составить фразы и предложения из слов?	Интуиция о правильном построении и значении.
	Что ограничивает возможное значение предложения?	Математическая модель строения. Например, теоретико-модельная семантика, теория формального языка.
Психолингвисты	Как люди могут определить структуру предложений?	Экспериментальные методы в основном для измерения работоспособности людей.
	Как можно определить значение слов?	Статистический анализ наблюдений.

	Когда приходит понимание?	
Философы	Как слова и предложения приобретают смысл? Как объекты идентифицируются по словам? Что это значит?	Аргументация на естественном языке с помощью интуиции. Математические модели, такие как логика и теория моделей.
Компьютерные лингвисты	Как определить структуру предложения Как можно смоделировать знания и рассуждения? Как мы можем использовать язык для решения конкретных задач?	Алгоритмы Структуры данных Формальные модели представления и рассуждения. Методы искусственного интеллекта, такие как методы поиска и представления.

Двусмысленность и неопределенность в языке

Неоднозначность, обычно используемая при обработке естественного языка, может быть обозначена как способность быть понятым более чем одним способом. Проще говоря, мы можем сказать, что двусмысленность - это способность пониматься более чем одним способом. Естественный язык очень неоднозначен. NLP имеет следующие типы двусмысленности:

Лексическая двусмысленность

Двусмысленность отдельного слова называется лексической неоднозначностью. Например, обрабатывая слово **silver** как существительное, прилагательное или глагол.

Синтаксическая двусмысленность

Такая двусмысленность возникает, когда предложение разбирается по-разному. Например, предложение «Мужчина увидел девушку в телескоп». Неизвестно, видел ли мужчина девушку с телескопом или он видел ее в свой телескоп.

Семантическая неоднозначность

Подобная двусмысленность возникает, когда значение самих слов может быть неправильно истолковано. Другими словами, семантическая неоднозначность возникает, когда предложение содержит неоднозначное слово или фразу. Например, предложение «Автомобиль ударился о столб, когда он двигался» имеет семантическую двусмысленность, поскольку интерпретациями могут быть «Автомобиль, двигаясь, ударился о столб» и «Автомобиль ударился о столб, когда столб двигался».

Анафорическая двусмысленность

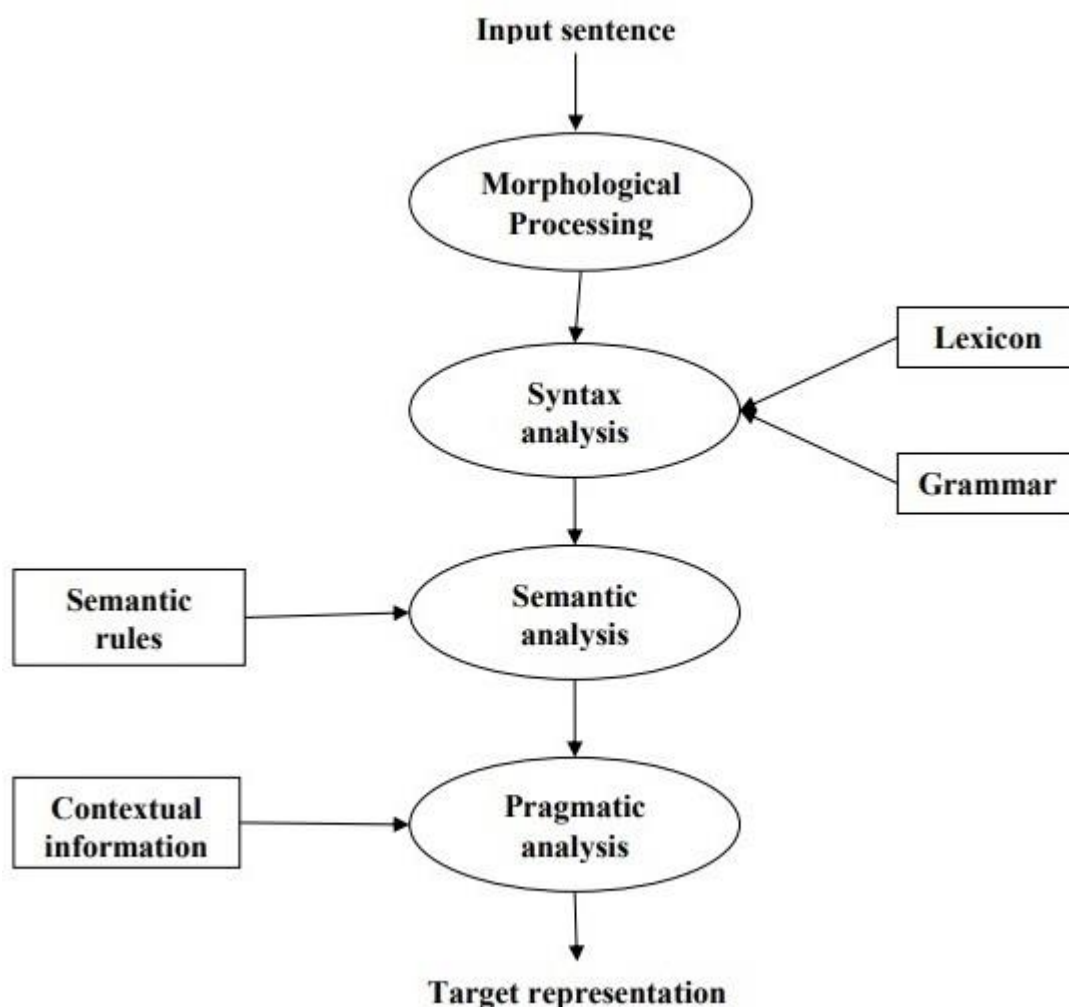
Подобная двусмысленность возникает из-за использования анафорных сущностей в дискурсе. Например, лошадь забежала в гору. Было очень круто. Вскоре она устала. Здесь анафорическое упоминание «она» в двух ситуациях вызывает двусмысленность.

Прагматическая двусмысленность

Подобная двусмысленность относится к ситуации, когда контекст фразы дает ей несколько интерпретаций. Проще говоря, мы можем сказать, что прагматическая двусмысленность возникает, когда утверждение не является конкретным. Например, предложение «Ты мне тоже нравишься» может иметь несколько интерпретаций: ты мне нравишься (точно так же, как ты нравишься мне), ты мне нравишься (точно так же, как кто-то другой).

Фазы NLP

На следующей диаграмме показаны фазы или логические шаги обработки естественного языка.



Морфологическая обработка

Это первая фаза NLP. Цель этого этапа - разбить фрагменты языкового ввода на наборы лексем, соответствующих абзацам, предложениям и словам. Например, такое слово, как **“uneasy”** можно разбить на два токена под слова как **“un-easy”**.

Синтаксический анализ

Это вторая фаза NLP. Цель этого этапа состоит в двух аспектах: проверить, правильно ли составлено предложение, и разбить его на структуру, которая показывает синтаксические отношения между разными словами. Например, предложение вроде **“The school goes to the boy”** будут отклонены синтаксическим анализатором или парсером.

Семантический анализ

Это третья фаза NLP. Цель этого этапа - определить точное значение, или вы можете сказать значение словаря из текста. Текст проверяется на осмысленность. Например, семантический анализатор отклонит предложение типа «Горячее мороженое».

Прагматический анализ

Это четвертая фаза NLP. Прагматический анализ просто соответствует реальным объектам / событиям, которые существуют в данном контексте, со ссылками на объекты, полученными на последнем этапе (семантический анализ). Например, предложение «Положить банан в корзину на полку» может иметь две смысловые интерпретации, и прагматический анализатор будет выбирать между этими двумя вариантами.

NLP - Анализ на уровне слов

В этой главе мы поймем анализ мирового уровня в обработке естественного языка.

Регулярные выражения

Регулярное выражение (RE, Regular Expression) - это язык для определения строк текстового поиска. RE помогает нам сопоставить или найти другие строки или наборы строк, используя специальный синтаксис, содержащийся в шаблоне. Регулярные выражения используются для поиска текстов как в UNIX, так и в MS WORD одинаково. У нас есть различные поисковые системы, использующие ряд функций RE.

Свойства регулярных выражений

Ниже приведены некоторые из важных свойств RE:

- Американский математик Стивен Коул Клини формализовал язык регулярных выражений.
- RE - это формула на специальном языке, с помощью которой можно указать простые классы строк, последовательность символов. Другими словами, мы можем сказать, что RE - это алгебраическое обозначение для характеристики набора строк.
- Регулярное выражение требует двух вещей: одна - это шаблон, который мы хотим искать, а другая - это корпус текста, по которому нам нужно искать.

Математически регулярное выражение можно определить следующим образом:

- ϵ является регулярным выражением, которое указывает, что язык имеет пустую строку.
- Φ - это регулярное выражение, которое означает, что это пустой язык.
- Если X и Y являются регулярными выражениями, то
 - X, Y
 - $X.Y$ (Concatenation of XY)
 - $X+Y$ (Union of X and Y)
 - X^*, Y^* (Kleen Closure of X and Y)

также являются регулярными выражениями.

- Если строка получена из приведенных выше правил, это также будет регулярное выражение.

Примеры регулярных выражений

В следующей таблице показано несколько примеров регулярных выражений -

Регулярные выражения	Обычный набор
$(0 + 10^*)$	$\{0, 1, 10, 100, 1000, 10000, \dots\}$
$(0^* 10^*)$	$\{1, 01, 10, 010, 0010, \dots\}$
$(0 + \epsilon) (1 + \epsilon)$	$\{\epsilon, 0, 1, 01\}$
$(a + b)^*$	Это будет набор строк a и b любой длины, который также включает в себя нулевую строку, т.е. $\{\epsilon, a, b, aa, ab, bb, ba, aaa, \dots\}$
$(a + b)^* abb$	Это будет набор строк a и b , оканчивающихся строкой abb , т.е. $\{abb, aabb, babb, aaabb, ababb, \dots\}$

$(11)^*$	Он будет состоять из четного числа единиц, который также включает пустую строку, например $\{\epsilon, 11, 1111, 111111, \dots\}$
$(aa)^* (bb)^* b$	Это будет набор строк, состоящих из четного числа букв a, за которыми следует нечетное количество символов b, т.е. $\{b, aab, aabbb, aabbbbb, aaaab, aaaabbb, \dots\}$
$(aa + ab + ba + bb)^*$	Это будет строка из a и b четной длины, которая может быть получена путем объединения любой комбинации строк aa, ab, ba и bb, включая null, т.е. $\{aa, ab, ba, bb, aaab, aaba, \dots\}$

Регулярные множества и их свойства

Его можно определить как набор, который представляет значение регулярного выражения и состоит из определенных свойств.

Свойства регулярных множеств

- Если мы сделаем объединение двух регулярных наборов, то полученный набор также будет регулярным.
- Если мы сделаем пересечение двух регулярных множеств, то полученный набор также будет регулярным.
- Если мы сделаем дополнение к регулярным множествам, то полученный набор также будет регулярным.
- Если мы сделаем разницу двух обычных наборов, то полученный набор также будет регулярным.
- Если мы сделаем перестановку регулярных множеств, то результирующий набор также будет регулярным.
- Если мы возьмем замыкание регулярных множеств, то полученное множество тоже будет регулярным.
- Если мы выполним конкатенацию двух регулярных наборов, то полученный набор также будет регулярным.

Конечные автоматы

Термин «автоматы», производный от греческого слова «αὐτόματη», означающего «самодействующий», является множественным числом «автомат», который может быть определен как абстрактное самоходное вычислительное устройство, которое автоматически выполняет заданную последовательность операций.

Автомат, имеющий конечное число состояний, называется конечным автоматом (FA, finite-state automaton, finite automaton или FSM, Finite-state machine).

Математически автомат может быть представлен 5-кортежем $(Q, \Sigma, \delta, q_0, F)$, где -

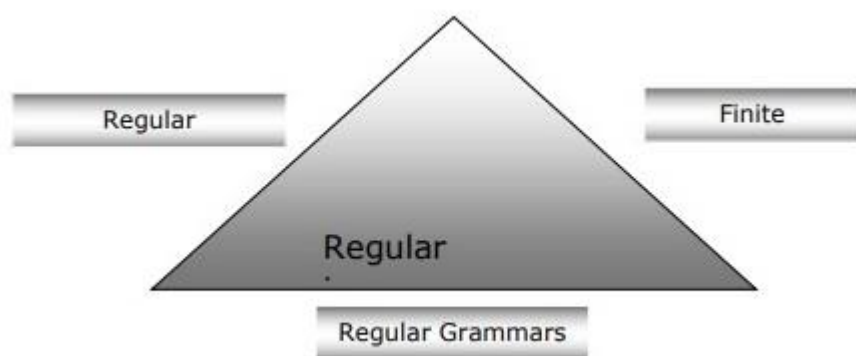
- Q - конечный набор состояний.
- Σ - конечный набор символов, называемый алфавитом автомата.
- δ - переходная функция
- q_0 - начальное состояние, из которого обрабатывается любой ввод ($q_0 \in Q$).
- F - это набор конечных состояний / состояний Q ($F \subseteq Q$).

Связь между конечными автоматами, регулярными грамматиками и регулярными выражениями

Следующие пункты дадут нам четкое представление о взаимосвязи между конечными автоматами, регулярными грамматиками и регулярными выражениями.

- Как мы знаем, конечные автоматы являются теоретической основой вычислительной работы, а регулярные выражения - одним из способов их описания.
- Можно сказать, что любое регулярное выражение может быть реализовано как FSA, а любое FSA может быть описано регулярным выражением.
- С другой стороны, регулярное выражение - это способ охарактеризовать разновидность языка, называемого регулярным языком. Следовательно, мы можем сказать, что регулярный язык может быть описан с помощью как FSA, так и регулярного выражения.
- Обычная грамматика, формальная грамматика, которая может быть регулярной справа или слева, является еще одним способом охарактеризовать регулярный язык.

Следующая диаграмма показывает, что конечные автоматы, регулярные выражения и регулярные грамматики являются эквивалентными способами описания регулярных языков.



Типы конечных автоматов (FSA)

Конечная автоматизация бывает двух типов. Посмотрим, какие бывают типы.

Детерминированный конечный автомат (DFA, deterministic finite automaton)

Его можно определить как тип конечной автоматизации, в котором для каждого входного символа мы можем определить состояние, в которое переместится машина. Он имеет конечное количество состояний, поэтому автомат называется Детерминированным конечным автоматом (ДКМ).

Математически DFA может быть представлен 5-кортежем $(Q, \Sigma, \delta, q_0, F)$, где -

- Q - конечный набор состояний.
- Σ - конечный набор символов, называемый алфавитом автомата.
- δ - функция перехода, где $\delta: Q \times \Sigma \rightarrow Q$.
- q_0 - начальное состояние, из которого обрабатывается любой ввод ($q_0 \in Q$).
- F - это набор конечных состояний / состояний Q ($F \subseteq Q$).

В то время как графически DFA может быть представлен диаграммами, называемыми диаграммами состояний, где

- Конечный набор состояний представлен **вершинами**.
- Переходы показаны помеченными **дугами**.
- Начальное состояние представлено **пустой входящей дугой**
- Конечное состояние представлено **двойным кругом**.

Пример DFA

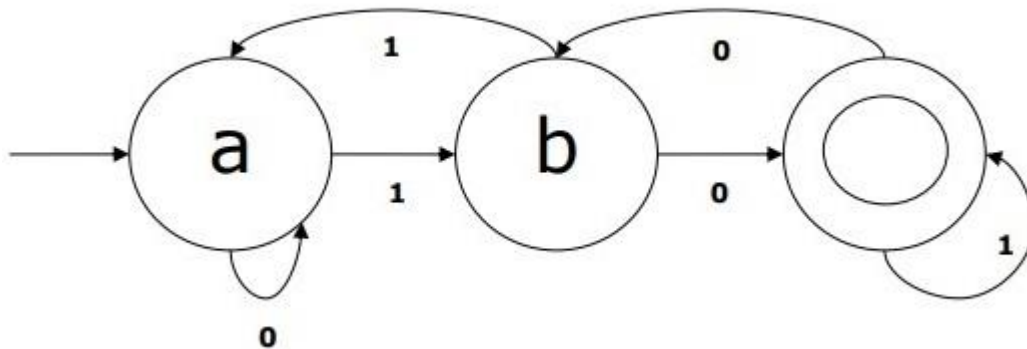
Предположим, что DFA

- $Q = \{a, b, c\}$,
- $\Sigma = \{0, 1\}$,
- $q_0 = \{a\}$,
- $F = \{c\}$,
- Функция перехода δ показана в таблице следующим образом -

Текущее состояние	Следующее состояние для входа 0	Следующее состояние для входа 1
a	a	b

b	b	a
c	c	c

Графическое представление этого DFA будет следующим:



Недетерминированный конечный автомат (NFA, Nondeterministic finite automaton)

Его можно определить как тип конечной автоматизации, когда для каждого входного символа мы не можем определить состояние, в которое переместится машина, то есть машина может перейти в любую комбинацию состояний. Он имеет конечное количество состояний, поэтому автомат называется недетерминированной конечной автоматизацией (NFA).

Математически NFA может быть представлена 5-кортежем $(Q, \Sigma, \delta, q_0, F)$, где -

- Q - конечный набор состояний.
- Σ - конечный набор символов, называемый алфавитом автомата.
- δ : -эта переходная функция , где $\delta: Q \times \Sigma \rightarrow 2^Q$.
- q_0 : -это начальное состояние, из которого обрабатывается любой ввод ($q_0 \in Q$).
- F : - это набор конечных состояний / состояний Q ($F \subseteq Q$).

В то время как графически (так же, как DFA), NFA может быть представлен диаграммами, называемыми диаграммами состояний, где -

- Набор состояний представлен **вершинами**.
- Переходы показаны помеченными **дугами**.
- Начальное состояние представлено **пустой входящей дугой**.
- Конечное состояние представлено **двойным кругом**.

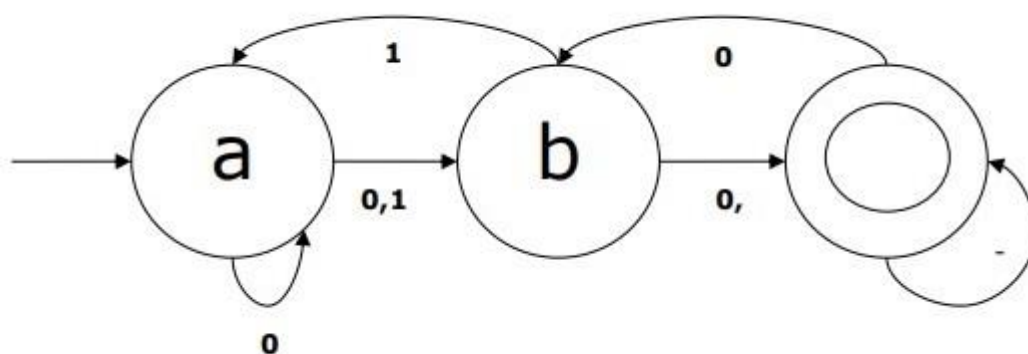
Пример NFA

Предположим, что NFA

- $Q = \{a, b, c\}$,
- $\Sigma = \{0, 1\}$,
- $q_0 = \{a\}$,
- $F = \{c\}$,
- Функция перехода δ показана в таблице следующим образом -

Текущее состояние	Следующее состояние для входа 0	Следующее состояние для входа 1
A	a, b	b
B	c	a, b
C	b, c	c

Графическое представление этого NDFA будет следующим:



Морфологический анализ

Термин «морфологический синтаксический анализ» относится к синтаксическому анализу морфем. Мы можем определить морфологический синтаксический анализ как проблему распознавания того, что слово разбивается на более мелкие значимые единицы, называемые морфемами, создавая для него своего рода лингвистическую структуру. Например, мы можем разбить слово *лисы* на два, *лиса* и *-es*. Мы видим, что слово *foxes* состоит из двух морфем, одна - *fox*, а другая - *-es*.

В другом смысле мы можем сказать, что морфология - это изучение -

- Формирование слов.
- Происхождение слов.
- Грамматические формы слов.
- Использование префиксов и суффиксов при образовании слов.
- Как формируются части речи (PoS) языка.

Типы морфем

Морфемы, самые маленькие смысловые единицы, можно разделить на два типа:

- Stems (стебли)
- Порядок слов

Стебли

Это основная значимая единица слова. Можно также сказать, что это корень слова. Например, в слове *foxes* основа - *fox*.

- **Affixes**- Как следует из названия, они добавляют словам некоторые дополнительные значения и грамматические функции. Например, в слове *лисы* аффикс - *es*.

Кроме того, аффиксы также можно разделить на следующие четыре типа:

- **Prefixes**- Как следует из названия, корень стоит перед префиксом. Например, в слове «расстегнуть ремень» используется префикс un.
- **Suffixes**- Как следует из названия, суффиксы следуют за корнем. Например, в слове кошки суффикс -s.
- **Infixes**- Как следует из названия, инфиксы вставляются внутрь основы. Например, слово cupful может иметь множественное число как cupsful, используя -s в качестве инфикса.
- **Circumfixes**- Они предшествуют стеблю и следуют за ним. Примеров циркумфиксов в английском языке очень мало. Очень распространенный пример - «A-ing», где мы можем использовать -A до начала, а -ing следует за корнем.

Порядок слов

Порядок слов будет определяться морфологическим анализом. Давайте теперь посмотрим на требования для создания морфологического парсера -

Словарный запас

Самым первым требованием для построения морфологического парсера является лексика, которая включает список основ и аффиксов вместе с основной информацией о них. Например, такая информация, как основа существительного или основа глагола и т. Д.

Морфотактика

По сути, это модель упорядочивания морфем. В другом смысле модель, объясняющая, какие классы морфем могут следовать за другими классами морфем внутри слова. Например, морфотактический факт состоит в том, что морфема множественного числа английского языка всегда следует за существительным, а не предшествует ему.

Орфографические правила

Эти правила написания используются для моделирования изменений, происходящих в слове. Например, правило преобразования у в ie в словах вроде city + s = города, а не города.

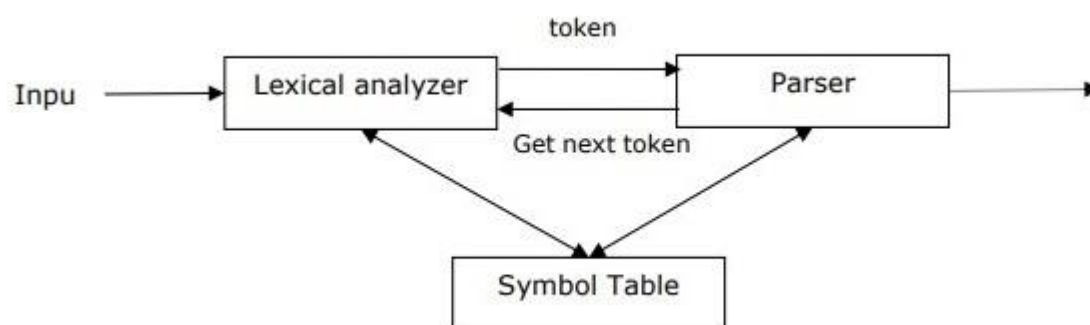
Обработка естественного языка - синтаксический анализ

Синтаксический анализ, синтаксический анализ или синтаксический анализ - это третья фаза NLP. Цель этого этапа - определить точное значение, или вы можете сказать значение словаря из текста. Анализ синтаксиса проверяет текст на осмысленность по сравнению с правилами формальной грамматики. Например, предложение «горячее мороженое» будет отклонено семантическим анализатором.

В этом смысле синтаксический анализ или синтаксический анализ можно определить как процесс анализа строк символов на естественном языке в соответствии с правилами формальной грамматики. Происхождение слова '*parsing*' от латинского слова '*pars*' что значит '*part*'.

Концепция парсера

Он используется для реализации задачи парсинга. Его можно определить как программный компонент, предназначенный для приема входных данных (текста) и предоставления структурного представления входных данных после проверки правильности синтаксиса в соответствии с формальной грамматикой. Он также строит структуру данных, как правило, в виде дерева синтаксического анализа или абстрактного синтаксического дерева или другой иерархической структуры.



Основные роли синтаксического анализа включают в себя:

- Чтобы сообщить о любой синтаксической ошибке.

- Чтобы исправить часто возникающую ошибку, чтобы можно было продолжить обработку оставшейся части программы.
- Чтобы создать дерево разбора.
- Создать таблицу символов.
- Создавать промежуточные представления (IR).

Типы парсинга

Вывод делит синтаксический анализ на следующие два типа:

- Анализ сверху вниз
- Анализ снизу вверх

Анализ сверху вниз

В этом виде синтаксического анализа синтаксический анализатор начинает построение дерева синтаксического анализа с начального символа, а затем пытается преобразовать начальный символ во входной. Наиболее распространенная форма синтаксического анализа сверху вниз использует рекурсивную процедуру для обработки ввода. Основным недостатком синтаксического анализа рекурсивного спуска является возврат с возвратом.

Анализ снизу вверх

В этом виде синтаксического анализа синтаксический анализатор начинает с входного символа и пытается построить дерево синтаксического анализатора до начального символа.

Понятие деривации

Чтобы получить входную строку, нам нужна последовательность производственных правил. Деривация - это набор производственных правил. Во время синтаксического анализа нам нужно определить нетерминал, который нужно заменить, а также определить производственное правило, с помощью которого будет заменен нетерминал.

Типы деривации

В этом разделе мы узнаем о двух типах производных, которые можно использовать, чтобы решить, какой нетерминал следует заменить производственным правилом:

Крайний левый вывод

В крайнем левом выводе текстовая форма ввода сканируется и заменяется слева направо. Предположительная форма в этом случае называется левой-сентенциальной формой.

Самая правая производная

В самом левом выводе текстовая форма ввода сканируется и заменяется справа налево. Предложенная форма в этом случае называется правой-сентенциальной формой.

Концепция дерева синтаксического анализа

Его можно определить как графическое изображение производной. Начальный символ вывода служит корнем дерева синтаксического анализа. В каждом дереве синтаксического анализа листовые узлы являются терминалами, а внутренние узлы - нетерминалами. Свойство дерева синтаксического анализа состоит в том, что обход по порядку дает исходную входную строку.

Понятие грамматики

Грамматика очень важна и важна для описания синтаксической структуры правильно сформированных программ. В литературном смысле они обозначают синтаксические правила разговора на естественных языках. Лингвистика пыталась дать определение грамматики с момента появления естественных языков, таких как английский, хинди и т. Д.

Теория формальных языков также применима в области компьютерных наук, в основном, в языках программирования и структуре данных. Например, в языке «C» точные правила грамматики определяют, как функции создаются из списков и операторов.

Математическая модель грамматики была дана **Noam Chomsky** в 1956 году, что эффективно для написания компьютерных языков.

Математически грамматика G может быть формально записана как 4-кортеж (N, T, S, P) , где -

- N или же V_N = набор нетерминальных символов, т. е. переменных.

- T или же Σ = набор терминальных символов.
- S = Начальный символ, где $S \in N$
- Обозначает Правила производства для терминалов, а также для нетерминалов. Он имеет вид $\alpha \rightarrow \beta$, где α и β - строки на $V_N \cup \Sigma$ и по крайней мере один символ α принадлежит V_N

Структура фразы или грамматика аудитории

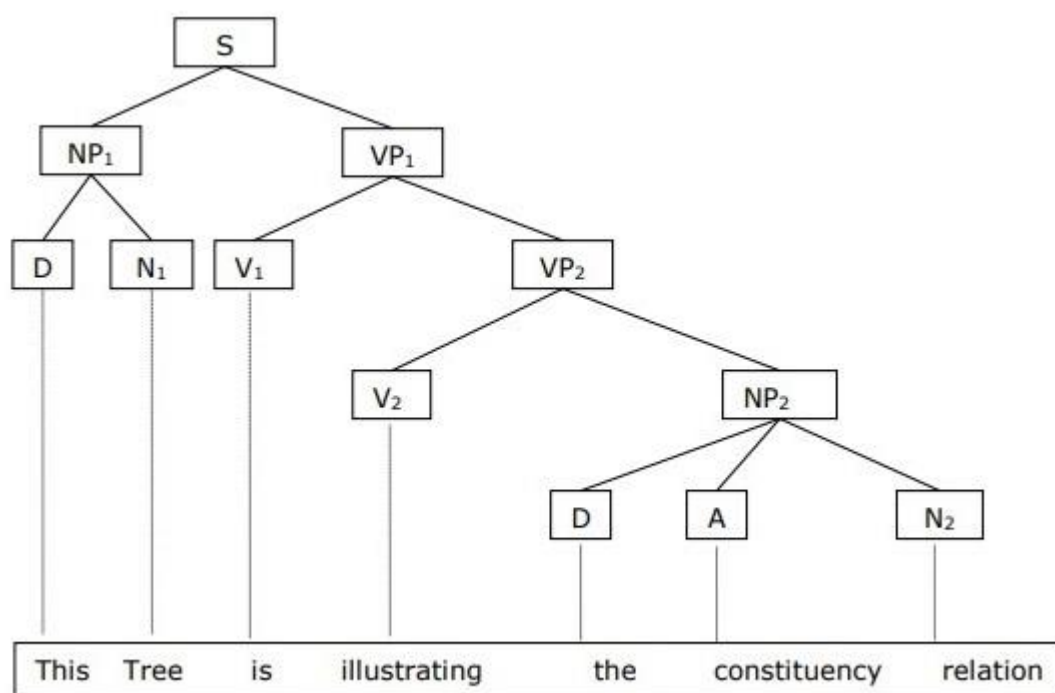
Грамматика фразовой структуры, введенная Ноамом Хомски, основана на отношении избирательного округа. Вот почему это также называется грамматикой избирательного округа. Это противоположно грамматике зависимостей.

пример

Перед тем, как привести пример грамматики избирательного округа, нам необходимо знать основные моменты, касающиеся грамматики избирательного округа и отношения избирательного округа.

- Все связанные структуры рассматривают структуру предложения с точки зрения отношения избирательного округа.
- Отношение избирательного округа происходит от деления подлежащего на предикат латыни, а также греческой грамматики.
- Основная структура предложения понимается с точки зрения **noun phrase NP** и **verb phrase VP**.

Мы можем написать предложение “**This tree is illustrating the constituency relation**” следующим образом -



Грамматика зависимостей

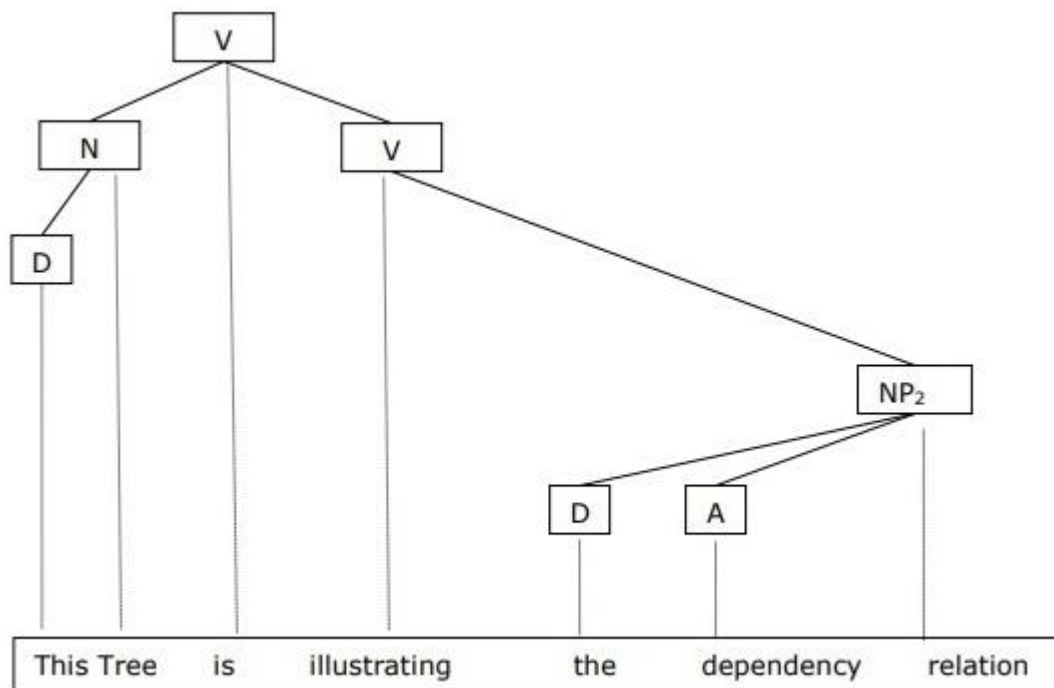
Он противоположен грамматике избирательного округа и основан на отношениях зависимости. Его представил Люсьен Тесньер. Грамматика зависимостей (DG) противоположна грамматике округа, потому что в ней отсутствуют фразовые узлы.

пример

Прежде чем приводить пример грамматики зависимостей, нам нужно знать основные моменты, касающиеся грамматики зависимостей и отношений зависимостей.

- В DG языковые единицы, то есть Слова, связаны друг с другом направленными связями.
- Глагол становится центром конструкции предложения.
- Все остальные синтаксические единицы связаны с глаголом посредством направленной ссылки. Эти синтаксические единицы называются **dependencies**.

Мы можем написать предложение “**This tree is illustrating the dependency relation**” следующим образом;



Дерево синтаксического анализа, использующее грамматику избирательного округа, называется деревом синтаксического анализа на основе избирательного округа; а дерево синтаксического анализа, использующее грамматику зависимостей, называется деревом синтаксического анализа на основе зависимостей.

Контекстная грамматика

Контекстно-свободная грамматика, также называемая CFG (Context-free grammar), представляет собой обозначение для описания языков и надмножество регулярной грамматики. Это можно увидеть на следующей диаграмме -



Определение CFG (контекстно-свободной грамматики)

CFG состоит из конечного набора грамматических правил со следующими четырьмя компонентами:

Набор нетерминалов

Он обозначается буквой V. Нетерминалы - это синтаксические переменные, которые обозначают наборы строк, которые дополнительно помогают определять язык, сгенерированный грамматикой.

Набор терминалов

Он также называется токенами и определяется Σ . Строки образуются из основных символов терминалов.

Набор постановок (продукций)

Обозначается буквой P. Набор определяет, как можно комбинировать терминалы и нетерминалы. Каждое производство (P) состоит из нетерминалов, стрелки и терминалов (последовательность терминалов). Нетерминалы называются левой стороной производства, а терминалы - правой стороной производства.

Начальный символ

Производство начинается с начального символа. Он обозначается символом S. Начальным символом всегда является нетерминальный символ.

Обработка естественного языка - семантический анализ

Целью семантического анализа является определение точного значения, или вы можете сказать значение словаря из текста. Работа семантического анализатора заключается в проверке текста на осмысленность.

Мы уже знаем, что лексический анализ также имеет дело со значением слов, тогда чем семантический анализ отличается от лексического анализа? Лексический анализ основан на меньшем токене, но, с другой стороны, семантический анализ фокусируется на более крупных фрагментах. Поэтому семантический анализ можно разделить на следующие две части:

Изучение значения отдельного слова

Это первая часть семантического анализа, в которой проводится изучение значения отдельных слов. Эта часть называется лексической семантикой.

Изучение сочетания отдельных слов

Во второй части отдельные слова будут объединены, чтобы придать смысл предложениям.

Самая важная задача семантического анализа - понять правильное значение предложения. Например, проанализируйте предложение **“Ram is great.”** В этом предложении говорящий говорит либо о Господе Рае, либо о человеке, которого зовут Рам. Вот почему важна работа семантического анализатора по улавливанию смысла предложения.

Элементы семантического анализа

Ниже приведены некоторые важные элементы семантического анализа:

Гипонимия

Это может быть определено как отношение между общим термином и экземплярами этого общего термина. Здесь общий термин называется гипернимом, а его экземпляры - гипонимами. Например, цвет слова - гипероним, а синий, желтый и т. Д. - гипонимы.

Омонимия

Его можно определить как слова, имеющие одинаковое написание или форму, но имеющие разное и несвязанное значение. Например, слово «летучая мышь» является омонимическим словом, потому что летучая мышь может быть инструментом для удара по мячу, или летучая мышь также является ночным летающим млекопитающим.

Полисемия

Полисемия - это греческое слово, означающее «много знаков». Это слово или фраза с другим, но родственным смыслом. Другими словами, мы можем сказать, что многозначность имеет то же написание, но другое и родственное значение. Например, слово «банк» - многозначное слово, имеющее следующие значения:

- Финансовое учреждение.
- Здание, в котором расположено такое заведение.
- Синоним слова «на что положиться».

Разница между многозначностью и омонимией

И многозначные, и омонимические слова имеют одинаковый синтаксис или написание. Основное различие между ними состоит в том, что в многозначности значения слов связаны, а в омонимии значения слов не связаны. Например, если мы говорим об одном и том же слове «Банк», мы можем написать значение «финансовое учреждение» или «берег реки». В этом случае это был бы пример омонима, потому что значения не связаны друг с другом.

Синонимия

Это отношение между двумя лексическими элементами, имеющими разные формы, но выражающими одинаковое или близкое значение. Примеры: «автор / писатель», «судьба / судьба».

Антонимия

Это отношение между двумя лексическими элементами, имеющими симметрию между их семантическими компонентами относительно оси. Сфера антонимии заключается в следующем -

- **Application of property or not** - Пример: «жизнь / смерть», «уверенность / неуверенность».
- **Application of scalable property** - Пример: "богатый / бедный", "горячий / холодный"
- **Application of a usage** - Пример: «отец / сын», «луна / солнце».

Представление о значениях

Семантический анализ создает представление о значении предложения. Но прежде чем углубляться в концепцию и подходы, связанные с представлением значения, нам необходимо понять строительные блоки семантической системы.

Строительные блоки семантической системы

В словесном представлении или представлении значения слов важную роль играют следующие строительные блоки:

- **Entities** - Он представляет личность, такую как конкретное лицо, место и т.д. Например, Харьяна. Индия, Рам - все сущности.
- **Concepts** - Он представляет собой общую категорию людей, таких как человек, город и т. Д.
- **Relations** - Он представляет отношения между сущностями и концепцией. Например, Рам - это человек.
- **Predicates** - Он представляет собой глагольные конструкции. Например, семантические роли и падежная грамматика являются примерами предикатов.

Теперь мы можем понять, что представление значения показывает, как собрать строительные блоки семантических систем. Другими словами, он показывает, как объединить сущности, концепции, отношения и предикаты для описания ситуации. Это также позволяет рассуждать о семантическом мире.

Подходы к представлениям о значениях

Семантический анализ использует следующие подходы для представления значения:

- Логика предикатов первого порядка (FOPL)
- Семантические сети
- Фреймы (Frames)
- Концептуальная зависимость (CD)
- Архитектура на основе правил
- Грамматика падежа
- Концептуальные графы (CG, Conceptual graph)

Необходимость смысловых представлений

Здесь возникает вопрос: зачем нам смысловое представление? Следующие причины этого:

Связывание лингвистических элементов с неязыковыми элементами

Самая первая причина заключается в том, что с помощью представления значений может быть выполнено соединение языковых элементов с неязыковыми элементами.

Представление разнообразия на лексическом уровне

С помощью смысловой репрезентации однозначные канонические формы могут быть представлены на лексическом уровне.

Может использоваться для рассуждений

Представление смысла может использоваться для обоснования проверки того, что истинно в мире, а также для вывода знаний из семантического представления.

Лексическая семантика

Первая часть семантического анализа, изучающая значение отдельных слов, называется лексической семантикой. Он также включает слова, подслова, аффиксы (части), составные слова и фразы. Все слова, подслова и т.д. Вместе называются лексическими элементами. Другими словами, мы можем сказать, что лексическая семантика - это отношения между лексическими элементами, значением предложений и синтаксисом предложения.

Ниже приведены шаги, связанные с лексической семантикой.

- Классификация лексических элементов, таких как слова, подслова, аффиксы и т. Д., Выполняется в лексической семантике.
- Декомпозиция лексических элементов, таких как слова, подслова, аффиксы и т. Д., Выполняется в лексической семантике.
- Также анализируются различия, а также сходство между различными лексико-семантическими структурами.

Устранение неоднозначности слов

Мы понимаем, что слова имеют разные значения в зависимости от контекста их использования в предложении. Если мы говорим о человеческих языках, то они тоже неоднозначны, потому что многие слова можно интерпретировать по-разному, в зависимости от контекста их появления.

Устранение неоднозначности смысла слова при обработке естественного языка (NLP) может быть определено как способность определять, какое значение слова активируется при использовании слова в конкретном контексте. Лексическая двусмысленность, синтаксическая или семантическая, - одна из самых первых проблем, с которыми сталкивается любая система NLP. Тегеры (от слова *tag*) части речи (POS) с высоким уровнем точности могут решить синтаксическую неоднозначность Word. С другой стороны, проблема разрешения семантической неоднозначности называется WSD (устранение неоднозначности смысла слов, *Word sense disambiguation*). Устранение семантической неоднозначности сложнее, чем устранение синтаксической неоднозначности.

Например, рассмотрим два примера различных значений слова **“bass”** -

- Я слышу басы.
- Он любит есть жареного окуня.

Возникновение слова **bass** имеет отчетливые значения. В первом предложении это означает тембр (*frequency*), а во-вторых, это означает рыбу. Следовательно, если WSD устранил неоднозначность, то правильное значение приведенных выше предложений может быть присвоено следующим образом:

- Я слышу басовый звук.
- Он любит есть жареного окуня / рыбу.

Оценка WSD

Оценка WSD требует следующих двух входных данных -

Словарь

Самым первым входом для оценки WSD является словарь, который используется для определения смыслов, которые необходимо устранить.

Тестовый корпус

Еще один ввод, требуемый WSD, - это аннотированный тестовый корпус, который имеет целевые или правильные чувства. Тестовые корпуса могут быть двух типов:

- **Lexical sample** - Этот вид корпусов используется в системе, где требуется устранить неоднозначность небольшого набора слов.

- **All-words** - Этот вид корпусов используется в системе, где ожидается устранение неоднозначности всех слов в фрагменте текущего текста.

Подходы и методы устранения неоднозначности слов (WSD)

Подходы и методы WSD классифицируются в зависимости от источника знаний, используемых при устранении неоднозначности.

Давайте теперь посмотрим на четыре обычных метода WSD:

Методы на основе словаря или знаний

Как следует из названия, для устранения неоднозначности эти методы в первую очередь полагаются на словари, сокровища и базу лексических знаний. Они не используют вещественные доказательства для устранения неоднозначности. Метод Леска - это основанный на словарях метод, представленный Майклом Леском в 1986 году. Определение Леска, на котором основан алгоритм Леска, выглядит следующим образом: **“measure overlap between sense definitions for all words in context”**. Однако в 2000 году Килгаррифф и Розенсвейг дали упрощенное определение Леска как **“measure overlap between sense definitions of word and current context”**, что также означает определение правильного значения для одного слова за раз. Здесь текущий контекст - это набор слов в окружающем предложении или абзаце.

Контролируемые методы

Для устранения неоднозначности методы машинного обучения используют для обучения корпуса с смысловыми аннотациями. Эти методы предполагают, что контекст сам по себе может предоставить достаточно свидетельств, чтобы устранить неоднозначность смысла. В этих методах слова «знание» и «рассуждение» считаются ненужными. Контекст представлен как набор «характеристик» слов. Он также включает информацию об окружающих словах. Машинное обучение опорных векторов и обучение на основе памяти - наиболее успешные подходы к обучению с учителем в WSD. Эти методы основаны на значительном количестве корпусов, помеченных вручную смысловыми метками, создание которых очень дорого.

Полу-контролируемые методы

Из-за отсутствия учебного корпуса большинство алгоритмов устранения неоднозначности смысла слов используют полууправляемые методы обучения. Это связано с тем, что полу-контролируемые методы используют как помеченные, так и немаркированные данные. Эти методы требуют очень небольшого количества аннотированного текста и большого количества простого неаннотированного текста. Техника, которая используется полууправляемыми методами, - это загрузка из исходных данных.

Неконтролируемые методы

Эти методы предполагают, что похожие смыслы возникают в аналогичном контексте. Вот почему чувства могут быть вызваны из текста путем кластеризации вхождений слов с использованием некоторой меры сходства контекста. Эта задача называется индукцией смысла слова или различием. Неконтролируемые методы имеют большой потенциал для преодоления узких мест в получении знаний из-за отсутствия зависимости от ручных усилий.

Приложения устранения неоднозначности слов (WSD)

Устранение неоднозначности слов (WSD) применяется почти во всех приложениях языковых технологий.

Давайте теперь посмотрим на масштабы WSD -

Машинный перевод

Машинный перевод или машинный перевод - наиболее очевидное применение WSD. В машинном переводе лексический выбор слов, которые имеют разные переводы для разных значений, выполняется WSD. Смыслы в машинном переводе представлены в виде слов на целевом языке. Большинство систем машинного перевода не используют явный модуль WSD.

Информационный поиск (IR, Information retrieval)

Информационный поиск (IR) можно определить как программу, которая занимается организацией, хранением, поиском и оценкой информации из репозитория документов, в частности текстовой информации. Система в основном помогает пользователям найти необходимую им информацию, но не дает явных ответов на вопросы. WSD используется для разрешения неоднозначности запросов, предоставляемых системе IR. Как и в случае с MT, текущие системы IR не используют модуль WSD явно и полагаются на концепцию, согласно которой пользователь вводит в запросе достаточно контекста, чтобы получать только соответствующие документы.

Текстовый анализ и извлечение информации (IE, Information Extraction)

В большинстве приложений WSD необходим для точного анализа текста. Например, WSD помогает интеллектуальной системе сбора данных отмечать правильные слова. Например, интеллектуальная медицинская система может нуждаться в маркировке «незаконных наркотиков», а не «медицинских препаратов».

Лексикография

WSD и лексикография могут работать вместе, поскольку современная лексикография основана на корпусах. С помощью лексикографии WSD предоставляет грубые эмпирические смысловые группировки, а также статистически значимые контекстные индикаторы смысла.

Трудности в устранении неоднозначности слов (WSD)

Ниже приведены некоторые трудности, с которыми сталкивается устранение неоднозначности слов (WSD):

Различия между словарями

Основная проблема WSD - определить смысл слова, потому что разные значения могут быть очень тесно связаны. Даже разные словари и тезаурусы могут по-разному делить слова на смыслы.

Разные алгоритмы для разных приложений

Еще одна проблема WSD в том, что для разных приложений может потребоваться совершенно другой алгоритм. Например, в машинном переводе это принимает форму выбора целевого слова; а при поиске информации не требуется смысловая инвентаризация.

Разница в оценках

Еще одна проблема WSD состоит в том, что системы WSD обычно тестируются путем сравнения результатов выполнения задачи с тем, как задача выполнена человеком. Это называется проблемой дисперсии среди оценок.

Словесная дискретность

Еще одна трудность WSD заключается в том, что слова не могут быть легко разделены на отдельные субсмыслы.

Обработка рассуждений на естественном языке

Самая сложная проблема ИИ - это обработка естественного языка компьютерами или, другими словами, *обработка естественного языка* - самая сложная проблема искусственного интеллекта. Если мы говорим об основных проблемах NLP, то одна из основных проблем NLP - это обработка дискурса (рассуждений) - построение теорий и моделей того, как высказывания слипаются, чтобы сформировать **coherent discourse** (последовательных суждений). На самом деле язык всегда состоит из объединенных, структурированных и связанных групп предложений, а не из отдельных и несвязанных предложений, как в фильмах. Эти связанные группы предложений называются дискурсом.

Концепция согласованности

Согласованность и структура дискурса во многом взаимосвязаны. Согласованность, наряду со свойством хорошего текста, используется для оценки качества вывода системы генерации естественного языка. Возникает вопрос: что значит связность текста? Предположим, мы собрали по одному предложению с каждой страницы газеты, тогда будет ли это дискурс? Конечно, нет. Это потому, что эти предложения не демонстрируют связности. Связный дискурс должен обладать следующими свойствами:

Отношение связности между высказываниями

Дискурс был бы последовательным, если бы между его высказываниями были значимые связи. Это свойство называется отношением когерентности. Например, должно быть какое-то объяснение, чтобы оправдать связь между высказываниями.

Отношения между сущностями

Еще одно свойство, делающее дискурс связным, - это то, что должны существовать определенные отношения с объектами. Такая согласованность называется согласованностью на основе сущностей.

Структура дискурса

Важный вопрос относительно дискурса заключается в том, какую структуру он должен иметь. Ответ на этот вопрос зависит от сегментации, которую мы применили к дискурсу. Сегментации дискурса можно определить как определение (распознавание) типов структур для широкого дискурса. Реализовать

сегментацию дискурса довольно сложно, но это очень важно для видов приложений **information retrieval**, **text summarization** и **information extraction**.

Алгоритмы сегментации дискурса

В этом разделе мы узнаем об алгоритмах сегментации дискурса. Алгоритмы описаны ниже

Неконтролируемая сегментация дискурса

Класс сегментации дискурса без учителя часто представляется как линейная сегментация. Разобраться в задаче линейной сегментации можно на примере. В этом примере есть задача разбить текст на блоки, состоящие из нескольких абзацев; единицы представляют собой отрывок из исходного текста. Эти алгоритмы зависят от сплоченности, которую можно определить как использование определенных лингвистических устройств для связывания текстовых единиц вместе. С другой стороны, сплоченность лексики - это сплоченность, на которую указывает связь между двумя или более словами в двух единицах, например использование синонимов.

Сегментация контролируемого дискурса

Более ранний метод не имеет границ сегментов, помеченных вручную. С другой стороны, контролируемая сегментация дискурса должна иметь данные обучения с границами. Приобрести такой же очень легко. В контролируемой сегментации дискурса важную роль играют дискурс-маркер или ключевые слова. Маркер дискурса или ключевое слово - это слово или фраза, которые служат для обозначения структуры дискурса. Эти маркеры дискурса зависят от предметной области.

Текстовая согласованность

Лексическое повторение - это способ найти структуру в дискурсе, но оно не удовлетворяет требованию связного дискурса. Чтобы достичь связного дискурса, мы должны сосредоточиться на конкретных отношениях согласованности. Как мы знаем, отношение когерентности определяет возможную связь между высказываниями в дискурсе. Хейб предложил такие отношения следующим образом:

Мы берем два срока S_0 и S_1 для представления значения двух связанных предложений -

Результат

Это означает, что состояние, утвержденное термином S_0 может вызвать состояние, заявленное S_1 . Например, два утверждения показывают результат отношений: Рам попал в огонь. Его кожа горела.

Объяснение

Это означает, что государство, утвержденное S_1 может вызвать состояние, заявленное S_0 . Например, два утверждения показывают отношения - Рам дрался с другом Шьяма. Он был пьян.

Параллелизм

Он выводит $p(a_1, a_2, \dots)$ из утверждения S_0 и $p(b_1, b_2, \dots)$ из утверждения S_1 . Здесь a_i и b_i одинаковы для всех i . Например, два утверждения параллельны - Рам хотел машину. Шьяму нужны были деньги.

Развитие

Он выводит одно и то же предложение P из обоих утверждений - S_0 и S_1 . Например, два утверждения показывают развитие отношения: Рам был из Чандигарха. Шьям был из Кералы.

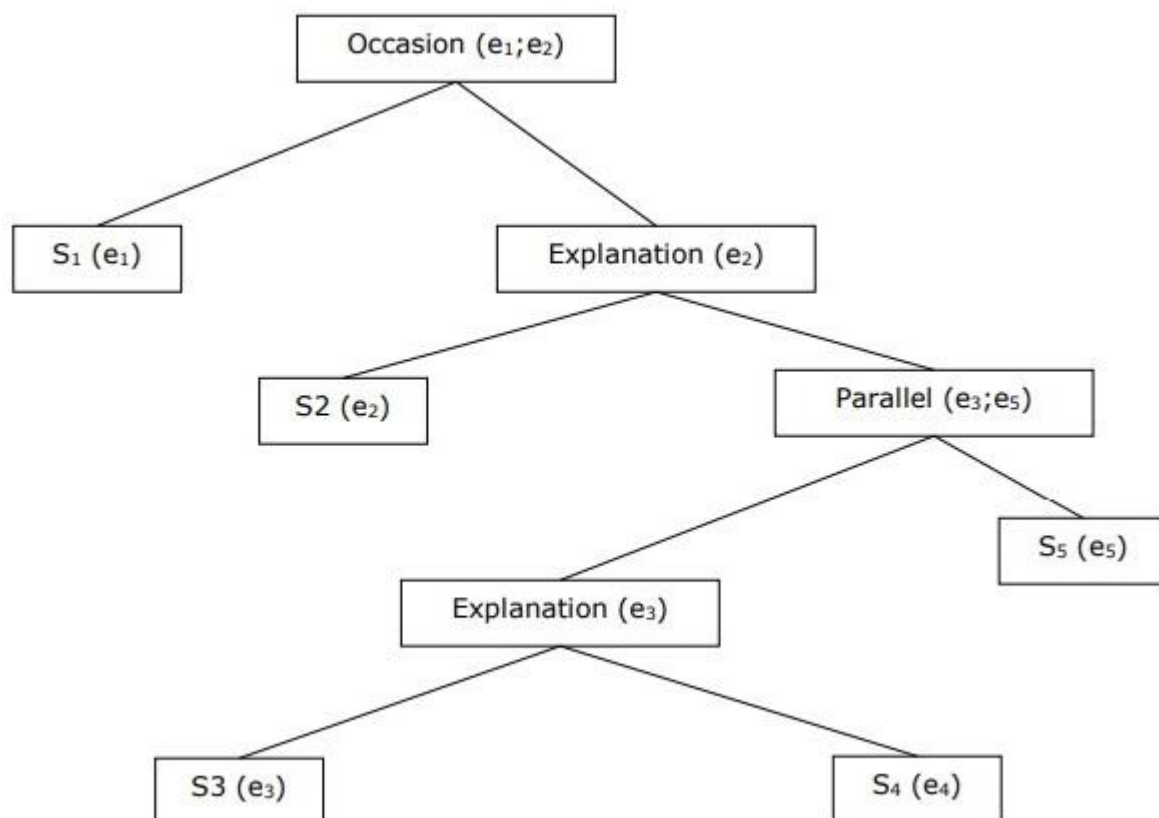
Повод

Это происходит, когда изменение состояния можно вывести из утверждения S_0 , конечное состояние которого можно вывести из S_1 и наоборот. Например, два утверждения показывают пример связи: Рам взял книгу. Он отдал его Шьяму.

Построение иерархической структуры дискурса

Связность всего дискурса также можно рассматривать с помощью иерархической структуры между отношениями когерентности. Например, следующий отрывок можно представить в виде иерархической структуры -

- S_1 - Рам пошел в банк, чтобы положить деньги.
- S_2 - Затем он сел на поезд до магазина одежды Шьяма.
- S_3 - Он хотел купить одежду.
- S_4 - У него нет новой одежды для вечеринки.
- S_5 - Он также хотел поговорить с Шьямом о его здоровье.



Разрешение ссылок

Интерпретация предложений из любого дискурса - еще одна важная задача, и для ее достижения нам необходимо знать, о ком или о какой сущности идет речь. Здесь ссылка на интерпретацию является ключевым элементом. **Reference** может быть определено как лингвистическое выражение для обозначения сущности или человека. Например, в отрывке Рам , менеджер банка ABC , увидел в магазине своего друга Шьяма. Он пошел ему навстречу, лингвистические выражения вроде «Рам», «Его», «Он» относятся к нему.

В той же ноте, **reference resolution** может быть определена как задача определения того, какие сущности относятся к каким языковым выражениям.

Терминология, используемая в справочном разрешении

Мы используем следующую терминологию в справочном разрешении -

- **Referring expression** - Выражение на естественном языке, которое используется для выполнения ссылки, называется выражением ссылки. Например, отрывок, использованный выше, является выражением ссылки.
- **Referent** - Речь идет о субъекте. Например, в последнем приведенном примере референтом является Рам.
- **Corefer** - Когда два выражения используются для обозначения одного и того же объекта, они называются corefers. Например, *Ram* и *he* кореферы.
- **Antecedent** - Термин имеет лицензию на использование другого термина. Например, *Ram* является антецедентом ссылки *he*.
- **Anaphora & Anaphoric** - Его можно определить как ссылку на объект, который ранее был введен в предложение. И ссылающееся выражение называется анафорическим.
- **Discourse model** - Модель, которая содержит представления сущностей, упомянутых в дискурсе, и отношения, в которые они вовлечены.

Типы ссылающихся выражений

Давайте теперь посмотрим на различные типы ссылающихся выражений. Пять типов ссылающихся выражений описаны ниже.

Неопределенные группы существительных

Такая ссылка представляет собой новые для слушателя сущности в контексте дискурса. Например, в предложении Рам однажды пришел принести ему немного еды - это неопределенное упоминание.

Определенные фразы существительных

В отличие от вышеизложенного, такой вид ссылки представляет собой объекты, которые не являются новыми или идентифицируемыми для слушателя в контексте дискурса. Например, в предложении - я читал "Таймс оф Индия" - "Таймс оф Индия" есть определенная ссылка.

Местоимения

Это форма определенной ссылки. Например, Рам рассмеялся так громко, как мог. Слово **he** представляет собой местоимение, относящееся к выражению.

Демонстрации

Они демонстрируют и ведут себя иначе, чем простые определенные местоимения. Например, это (it) и это (this) – указательные местоимения.

Имена

Это простейший тип ссылающегося выражения. Это также может быть имя человека, организации и местонахождения. Например, в приведенных выше примерах Ram - это выражение, обозначающее имя.

Задачи разрешения ссылок

Две задачи разрешения ссылок описаны ниже.

Разрешение Coreference

Это задача поиска выражений в тексте, которые относятся к одному и тому же объекту. Проще говоря, это задача поиска кореферных выражений. Набор выражений кореферринга называется цепочкой кореференции. Например - Он, Главный Менеджер и Его - это относящиеся выражения в первом отрывке, приведенном в качестве примера.

Ограничение на разрешение Coreference

В английском языке основная проблема разрешения кореферентности - это местоимение it. Причина этого в том, что это местоимение имеет много применений. Например, это может означать он и она. Местоимение это также относится к вещам, которые не относятся к конкретным вещам. Например, идет дождь. Это действительно хорошо.

Разрешение прономинальной анафоры

В отличие от разрешения кореферентности, разрешение местоименной анафоры можно определить как задачу поиска antecedenta для отдельного местоимения. Например, местоимение принадлежит ему, и задача разрешения местоименной анафоры состоит в том, чтобы найти слово «Рам», потому что «Рам» является antecedentом.

Тегирование (маркировка) частей речи (PoS, part-of-speech tagging)

Маркировка - это разновидность классификации, которую можно определить как автоматическое присвоение описания токенам. Здесь дескриптор называется тегом, который может представлять одну из частей речи, семантической информации и так далее.

Теперь, если мы говорим о тегировании части речи (PoS), то это можно определить как процесс присвоения одной из частей речи данному слову. Обычно это называется POS-тегами. Проще говоря, мы можем сказать, что POS-теги - это задача пометить каждое слово в предложении соответствующей частью речи. Мы уже знаем, что части речи включают существительные, глаголы, наречия, прилагательные, местоимения, союзы и их подкатегории.

Большая часть тегов POS подпадает под теги POS на основе правил, стохастические теги POS и теги на основе преобразования.

Маркировка POS на основе правил

Один из старейших методов тегирования - это тегирование POS на основе правил. Тегеры на основе правил используют словарь или лексику для получения возможных тегов для тегирования каждого слова. Если слово имеет более одного возможного тега, то тегеры на основе правил используют написанные от руки правила для определения правильного тега. Устранение неоднозначности также может быть выполнено в тегах на основе правил путем анализа лингвистических характеристик слова вместе с

предшествующими и последующими словами. Например, предположим, что если предыдущее слово слова - артикль, то слово должно быть существительным.

Как следует из названия, вся такая информация в тегах POS на основе правил кодируется в форме правил. Эти правила могут быть либо -

- Правила шаблона контекста
- Или, как регулярное выражение, скомпилированное в конечный автомат, пересекающееся с лексически неоднозначным представлением предложения.

Мы также можем понять тегирование POS на основе правил по его двухэтапной архитектуре -

- **First stage** - На первом этапе он использует словарь, чтобы присвоить каждому слову список потенциальных частей речи.
- **Second stage** - На втором этапе он использует большие списки рукописных правил устранения неоднозначности, чтобы отсортировать список до одной части речи для каждого слова.

Свойства POS-тегов на основе правил

Тегеры POS на основе правил обладают следующими свойствами:

- Эти тегеры ориентированы на знания.
- Правила в тегах POS на основе правил создаются вручную.
- Информация закодирована в виде правил.
- У нас есть ограниченное количество правил, примерно около 1000.
- Сглаживание и языковое моделирование явно определены в тегах на основе правил.

Стохастическая маркировка POS

Другой метод тегирования - это Stochastic POS Tagging. Теперь возникает вопрос, какая модель может быть стохастической. Модель, включающую частоту или вероятность (статистику), можно назвать стохастической. Любое количество различных подходов к проблеме тегирования части речи можно назвать стохастическим тегером.

В простейшем стохастическом тегере применяются следующие подходы к тегированию POS:

Подход оценки частоты использования слов

В этом подходе стохастические тегеры устраняют неоднозначность слов на основе вероятности того, что слово встречается с конкретным тегом. Мы также можем сказать, что тег, который чаще всего встречается со словом в обучающем наборе. Это тег, присвоенный неоднозначному экземпляру этого слова. Основная проблема такого подхода в том, что он может привести к недопустимой последовательности тегов.

Вероятности последовательности тегов

Это еще один подход к стохастической маркировке, при которой устройство для маркировки вычисляет вероятность появления данной последовательности тегов. Это также называется подходом n-грамм. Он называется так, потому что лучший тег для данного слова определяется вероятностью, с которой он встречается с n предыдущими тегами.

Свойства стохастической POST-маркировки

Стохастические теги POS обладают следующими свойствами:

- Эта маркировка POS основана на вероятности появления тега.
- Требуется тренировочный корпус
- Не было бы вероятности для слов, которых нет в корпусе.
- Он использует другой корпус тестирования (кроме корпуса обучения).
- Это простейшая маркировка POS, потому что она выбирает наиболее частые теги, связанные со словом в обучающем корпусе.

Тегирование на основе преобразования

Маркировка на основе преобразования также называется маркировкой Brill. Как пример - обучение на основе преобразования (TBL, transformation based tagging), который представляет собой основанный на правилах алгоритм для автоматической привязки POS к заданному тексту. TBL позволяет нам иметь лингвистические знания в удобочитаемой форме, преобразует одно состояние в другое состояние с помощью правил преобразования.

Он черпает вдохновение из обоих объясненных ранее тегеров - основанных на правилах и стохастических. Если мы видим сходство между тегами на основе правил и преобразованием, то, как и в случае с тегами на основе правил, оно также основано на правилах, которые определяют, какие теги должны быть назначены каким словам. С другой стороны, если мы видим сходство между стохастиком и теггером преобразования, то, как и стохастик, это метод машинного обучения, в котором правила автоматически индуцируются из данных.

Работа по обучению на основе трансформации (TBL)

Чтобы понять принцип работы и концепцию тегеров на основе преобразований, нам необходимо понять работу обучения на основе преобразований. Рассмотрим следующие шаги, чтобы понять работу TBL:

- **Start with the solution** - TBL обычно начинается с решения проблемы и работает циклически.
- **Most beneficial transformation chosen** - В каждом цикле TBL выберет наиболее выгодную трансформацию.
- **Apply to the problem** - Преобразование, выбранное на последнем шаге, будет применено к проблеме.

Алгоритм остановится, когда выбранное преобразование на шаге 2 не добавит дополнительных значений или если больше не будет выбранных преобразований. Такой способ обучения лучше всего подходит для задач классификации.

Преимущества обучения на основе трансформации (TBL)

Преимущества TBL следующие:

- Мы изучаем небольшой набор простых правил, и этих правил достаточно для тегирования.
- Разработка и отладка в TBL очень просты, потому что выученные правила легко понять.
- Сложность тегирования снижается, поскольку в TBL чередуются правила, полученные машиной, и правила, созданные человеком.
- Маркировщик на основе преобразования работает намного быстрее, чем марковский маркер.

Недостатки обучения на основе трансформации (TBL)

Недостатки TBL следующие:

- Обучение на основе преобразования (TBL) не обеспечивает вероятностей тегов.
- В TBL время обучения очень долгое, особенно на больших корпусах.

Скрытая марковская модель (HMM, hidden Markov model) POS Tagging

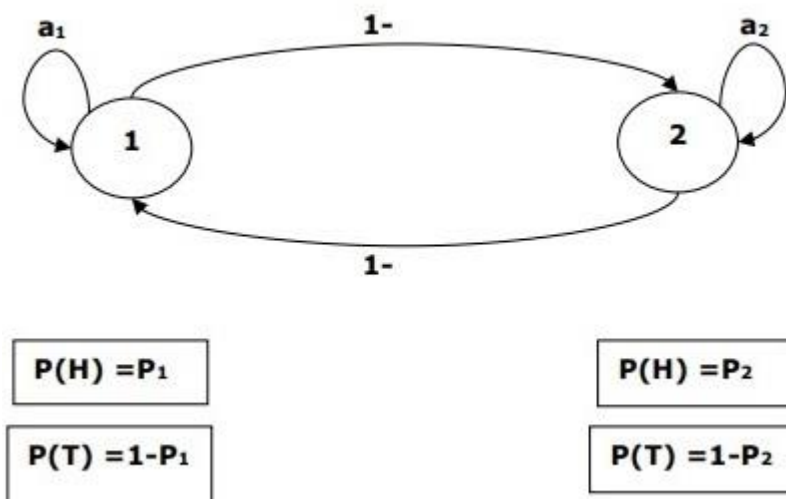
Прежде чем углубляться в теги HMM POS, мы должны понять концепцию скрытой марковской модели (HMM).

Скрытая марковская модель

Модель HMM может быть определена как стохастическая модель с двумя вложениями, в которой лежащий в основе стохастический процесс скрыт. Этот скрытый случайный процесс можно наблюдать только с помощью другого набора стохастических процессов, который производит последовательность наблюдений.

Пример

Например, проводится последовательность экспериментов по подбрасыванию скрытой монеты, и мы видим только последовательность наблюдений, состоящую из орла и решки. Фактические детали процесса - сколько монет было использовано, порядок их выбора - скрыты от нас. Наблюдая за этой последовательностью орла и решки, мы можем построить несколько HMM для объяснения этой последовательности. Ниже приведена одна из форм скрытой марковской модели для этой проблемы.



Мы предположили, что в HMM есть два состояния, и каждое из состояний соответствует выбору разных смещенных монет. Следующая матрица дает вероятности перехода состояний -

Вот,

- a_{ij} = вероятность перехода из одного состояния в другое от i до j .
- $a_{11} + a_{12} = 1$ и $a_{21} + a_{22} = 1$
- P_1 = вероятность выпадения орла первой монеты, т.е. смещение первой монеты.
- P_2 = вероятность выпадения орла второй монеты, то есть смещение второй монеты.

Мы также можем создать модель HMM, предполагая, что есть 3 или более монет.

Таким образом, мы можем охарактеризовать HMM следующими элементами:

- N - количество состояний в модели (в приведенном выше примере $N = 2$, только два состояния).
- M , количество различных наблюдений, которые могут появиться с каждым состоянием в приведенном выше примере $M = 2$, т. Е. H или T).
- A , распределение вероятностей перехода состояний - матрица A в приведенном выше примере.
- P , распределение вероятностей наблюдаемых символов в каждом состоянии (в нашем примере P_1 и P_2).
- I , начальное состояние распределения.

Использование HMM для тегов POS

Процесс маркировки POS - это процесс поиска последовательности тегов, которая с наибольшей вероятностью сгенерировала данную последовательность слов. Мы можем смоделировать этот процесс POS, используя скрытую марковскую модель (HMM), где **tags** являются **hidden states** который произвел **observable output**, т.е. **words**.

С математической точки зрения, в тегах POS мы всегда заинтересованы в поиске последовательности тегов (C), которая максимизирует -

$$P(C|W)$$

Где,

$$C = C_1, C_2, C_3 \dots C_T$$

$$W = W_1, W_2, W_3, W_T$$

С другой стороны, факт в том, что нам нужно много статистических данных, чтобы разумно оценивать такие последовательности. Однако, чтобы упростить задачу, мы можем применить некоторые математические преобразования наряду с некоторыми предположениями.

Использование HMM для маркировки POS - это особый случай байесовского вмешательства. Следовательно, мы начнем с повторения проблемы, используя правило Байеса, которое гласит, что вышеупомянутая условная вероятность равна -

$$(\text{PROB}(C_1, \dots, C_T) * \text{PROB}(W_1, \dots, W_T | C_1, \dots, C_T)) / \text{PROB}(W_1, \dots, W_T)$$

Мы можем исключить знаменатель во всех этих случаях, потому что мы заинтересованы в нахождении последовательности C, которая максимизирует указанное выше значение. Это не повлияет на наш ответ. Теперь наша проблема сводится к поиску последовательности C, которая максимизирует -

$$\text{PROB}(C_1, \dots, C_T) * \text{PROB}(W_1, \dots, W_T | C_1, \dots, C_T) \quad (1)$$

Даже после уменьшения проблемы в приведенном выше выражении потребуется большой объем данных. Мы можем сделать разумные предположения о независимости двух вероятностей в приведенном выше выражении для преодоления проблемы.

Первое предположение

Вероятность тега зависит от предыдущего (модель биграмм) или двух предыдущих (модель триграммы) или предыдущих n тегов (модель n-грамм), что математически можно объяснить следующим образом:

$$\text{PROB}(C_1, \dots, C_T) = \prod_{i=1..T} \text{PROB}(C_i | C_{i-n+1} \dots C_{i-1}) \quad (\text{n-gram model})$$

$$\text{PROB}(C_1, \dots, C_T) = \prod_{i=1..T} \text{PROB}(C_i | C_{i-1}) \quad (\text{bigram model})$$

Начало предложения можно объяснить, приняв начальную вероятность для каждого тега.

$$\text{PROB}(C_1 | C_0) = \text{PROB}_{\text{initial}}(C_1)$$

Второе предположение

Вторая вероятность в уравнении (1) выше может быть аппроксимирована, если предположить, что слово появляется в категории, независимой от слов в предшествующих или последующих категориях, что можно математически объяснить следующим образом:

$$\text{PROB}(W_1, \dots, W_T | C_1, \dots, C_T) = \prod_{i=1..T} \text{PROB}(W_i | C_i)$$

Теперь, основываясь на двух вышеупомянутых предположениях, наша цель сводится к нахождению последовательности C, которая максимизирует

$$\prod_{i=1..T} \text{PROB}(C_i | C_{i-1}) * \text{PROB}(W_i | C_i)$$

Теперь возникает вопрос, действительно ли нам помогло преобразование проблемы в приведенную выше форму. Ответ - да, есть. Если у нас есть большой корпус с тегами, то две вероятности в приведенной выше формуле можно рассчитать как -

$$\text{PROB}(C_{i=\text{VERB}} | C_{i-1=\text{NOUN}}) = (\# \text{ of instances where Verb follows Noun}) / (\# \text{ of instances where Noun appears}) \quad (2)$$

$$\text{PROB}(W_i | C_i) = (\# \text{ of instances where } W_i \text{ appears in } C_i) / (\# \text{ of instances where } C_i \text{ appears}) \quad (3)$$

Обработка естественного языка - Начало

В этой главе мы обсудим возникновение естественного языка в обработке естественного языка. Для начала давайте сначала разберемся, что такое грамматика естественного языка.

Грамматика естественного языка

Для лингвистики язык - это группа произвольных голосовых знаков. Мы можем сказать, что язык творческий, управляемый правилами, врожденными и универсальными одновременно. С другой стороны, это тоже по-человечески. Природа языка у разных людей разная. Существует много неправильных представлений о природе языка. Вот почему очень важно понимать значение неоднозначного термина **'grammar'**. В лингвистике термин грамматика может быть определен как правила или принципы, с помощью которых работает язык. В широком смысле мы можем разделить грамматику на две категории:

Описательная грамматика

Набор правил, в которых лингвисты и грамматик формулируют грамматику говорящего, называется описательной грамматикой.

Перспективная грамматика

Это совершенно другое понимание грамматики, которое пытается поддерживать стандарт правильности языка. Эта категория не имеет ничего общего с реальной работой языка.

Компоненты языка

Язык обучения делится на взаимосвязанные компоненты, которые являются как условными, так и произвольными разделами лингвистического исследования. Объяснение этих компонентов следующее

Фонология

Самый первый компонент языка - фонология. Это изучение звуков речи определенного языка. Происхождение слова можно проследить до греческого языка, где «телефон» означает звук или голос. Фонетика, раздел фонологии, изучает звуки речи человеческого языка с точки зрения их производства, восприятия или их физических свойств. IPA (Международный фонетический алфавит) - это инструмент, который регулярно представляет человеческие звуки при изучении фонологии. В IPA каждый письменный символ представляет один и только один речевой звук и наоборот.

Фонемы

Его можно определить как одну из звуковых единиц, отличающих одно слово от другого в языке. В лингвистике фонемы пишутся между косыми чертами. Например, фонема /k/ встречается в таких словах, как kit, skit.

Морфология

Это второй компонент языка. Это изучение структуры и классификации слов на определенном языке. Слово происходит из греческого языка, где слово «морфе» означает «форма». Морфология рассматривает принципы образования слов в языке. Другими словами, как звуки объединяются в значимые единицы, такие как префиксы, суффиксы и корни. Также рассматривается, как слова могут быть сгруппированы в части речи.

Лексема

В лингвистике абстрактная единица морфологического анализа, которая соответствует набору форм, принимаемых одним словом, называется лексемой. То, как лексема используется в предложении, определяется ее грамматической категорией. Лексема может быть индивидуальной или многословной. Например, слово «разговор» является примером отдельной словарной лексемы, которая может иметь множество грамматических вариантов, таких как разговоры, разговоры и разговоры. Многословная лексема может состоять более чем из одного орфографического слова. Например, высказаться, протянуть и т. Д. Являются примерами многословных лексем.

Синтаксис

Это третий компонент языка. Это изучение порядка и расположения слов в более крупных единицах. Слово восходит к греческому языку, где слово *suntassein* означает «наводить порядок». Он изучает типы предложений и их структуру, придаточных предложений, фраз.

Семантика

Это четвертый компонент языка. Это исследование того, как передается смысл. Значение может быть связано с внешним миром или может быть связано с грамматикой предложения. Это слово восходит к греческому языку, где слово *semaínein* означает «обозначать», «показывать», «сигнализировать».

Прагматика

Это пятая составляющая языка. Это изучение функций языка и его использования в контексте. Происхождение слова можно проследить до греческого языка, где слово «прагма» означает «дело», «дело».

Грамматические категории

Грамматическая категория может быть определена как класс единиц или функций в грамматике языка. Эти единицы являются строительными блоками языка и имеют общий набор характеристик. Грамматические категории также называют грамматическими признаками.

Перечень грамматических категорий описан ниже.

Число

Это простейшая грамматическая категория. У нас есть два термина, относящиеся к этой категории - единственное и множественное число. Единственное число - это понятие «один», а множественное - понятие «более одного». Например, собака / собаки, это / эти.

Пол

Грамматический род выражается вариацией личных местоимений и третьего лица. Примеры грамматических родов единственного числа - он, она, оно; формы первого и второго лица - я, мы и ты; 3-е лицо множественного числа они - либо общий, либо средний род.

Лицо

Еще одна простая грамматическая категория - это лицо. При этом признаются следующие три условия:

- **1st person** - Говорящий распознается как первое лицо.
- **2nd person** - Человек, который является слушателем или собеседником, признается вторым лицом.
- **3rd person** - Человек или предмет, о котором мы говорим, распознается как третье лицо.

Падеж

Это одна из самых сложных грамматических категорий. Это может быть определено как указание на функцию именной группы (NP) или отношение именной фразы к глаголу или другим существительным фразам в предложении. У нас есть следующие три падежа, выраженные личными и вопросительными местоимениями:

- **Nominative case** - Это функция субъекта. Например, я, мы, ты, он, она, оно, они и кто в именительном падеже.
- **Genitive case** - Это функция владельца. Например, my / my, our / ours, his, her / hers, its, their / theirs, которые имеют родительный падеж.
- **Objective case** - Это функция объекта. Например, я, мы, вы, он, она, они, кому объективны.

Степень

Эта грамматическая категория относится к прилагательным и наречиям. Он имеет следующие три условия -

- **Positive degree** - Он выражает качество. Например, большой, быстрый, красивый - положительные степени.
- **Comparative degree** - Он выражает большую степень или интенсивность качества в одном из двух предметов. Например, сравнительно больше, быстрее, красивее.
- **Superlative degree** - Он выражает наибольшую степень или интенсивность качества одного из трех или более предметов. Например, самый большой, самый быстрый, самый красивый - это превосходные степени.

Определенность и неопределенность

Обе эти концепции очень просты. Как мы знаем, определенность представляет собой референт, который известен, знаком или идентифицируется говорящим или слушателем. С другой стороны, неопределенность представляет собой референт, который неизвестен или незнаком. Понятие можно понять в сочетании артикля с существительным -

- **definite article-** *The*
- **indefinite article-** *a / an*

Время

Эта грамматическая категория связана с глаголом и может быть определена как лингвистическое указание времени действия. Время устанавливает отношение, поскольку указывает время события по отношению к моменту разговора. В целом, он бывает следующих трех типов -

- **Present tense-** Представляет возникновение действия в настоящий момент. Например, Рам много работает.
- **Past tense-** Представляет возникновение действия до настоящего момента. Например, шел дождь.
- **Future tense-** Представляет возникновение действия после настоящего момента. Например, пойдет дождь.

Аспект

Эту грамматическую категорию можно определить как взгляд на событие. Он может быть следующих типов:

- **Perfective aspect**- Вид воспринимается как цельный и законченный в разрезе. Например, простое прошедшее время, такое как **yesterday I met my friend**, на английском языке является совершенным в аспекте, поскольку он рассматривает событие как законченное и цельное.
- **Imperfective aspect**- Взгляд воспринимается как продолжающийся и неполный в аспекте. Например, время причастия настоящего времени как **I am working on this problem**, на английском языке несовершенный аспект, поскольку он рассматривает событие как неполное и продолжающееся.

Наклонение

Эту грамматическую категорию немного сложно определить, но ее можно просто сформулировать как показатель отношения говорящего к тому, о чем он / она говорит. Это также грамматическая особенность глаголов. Он отличается от грамматических времен и грамматического аспекта. Примеры наклонений: указательный, вопросительный, повелительный, повелительный, сослагательный, потенциальный, оптативный, герундий и причастия.

Согласование

Его еще называют конкордом. Это происходит, когда слово изменяется в зависимости от других слов, к которым оно относится. Другими словами, это подразумевает согласование значения какой-либо грамматической категории между разными словами или частями речи. Ниже приведены согласования, основанные на других грамматических категориях -

- **Agreement based on Person** - Согласование между подлежащим и глаголом. Например, мы всегда используем «Я есть» (I am) и «Он есть» (He is), но никогда не используем «Он есть» (He am) и «Я есть» (I is).
- **Agreement based on Number** – Эта согласование между подлежащим и глаголом. В этом случае существуют особые формы глаголов для первого лица единственного числа, второго лица множественного числа и так далее. Например, 1-е лицо единственного числа: Я действительно, 2-е лицо множественного числа: Мы действительно, 3-е лицо единственного числа: мальчик поет, 3-е лицо множественного числа: мальчики поют.
- **Agreement based on Gender** - В английском языке родовые отношения между местоимениями и антецедентами совпадают. Например, Он достиг своего места назначения. Корабль достиг пункта назначения.
- **Agreement based on Case** - Такой вид согласованности не является существенной особенностью английского языка. Например, кто пришел первым - он или его сестра?

Разговорный синтаксис языка

Письменная и устная английская грамматика имеют много общих черт, но, вместе с тем, они различаются по ряду аспектов. Следующие особенности различают устную и письменную грамматику английского языка:

Репарандум и исправление

Эта поразительная особенность отличает устную и письменную грамматику английского языка друг от друга. По отдельности он известен как феномен отказа, а в совокупности - как феномен ремонта. Недостатки включают использование следующего -

- **Fillers words (заполнители слов)** - Иногда между предложениями мы используем несколько слов-заполнителей. Их называют наполнителями филлерной паузы. Примеры таких слов: uh и um.
- **Reparandum and repair (репарандум и исправление)** - Повторяющийся отрезок слов между предложениями называется репарандумом. В том же сегменте измененное слово называется исправлением. Рассмотрим следующий пример, чтобы понять это -

Does ABC airlines offer any one-way flights uh one-way fares for 5000 rupees?

В приведенном выше предложении рейс в одну сторону - это ремонт, а билет за 5000 рупий - это исправление.

Перезапуск

После паузы или заполнителя происходит перезапуск. Например, в приведенном выше предложении перезапуск происходит, когда говорящий начинает спрашивать о рейсах в одну сторону, затем останавливается, исправляет себя, заполняя паузу, а затем снова начинает спрашивать о билетах в одну сторону.

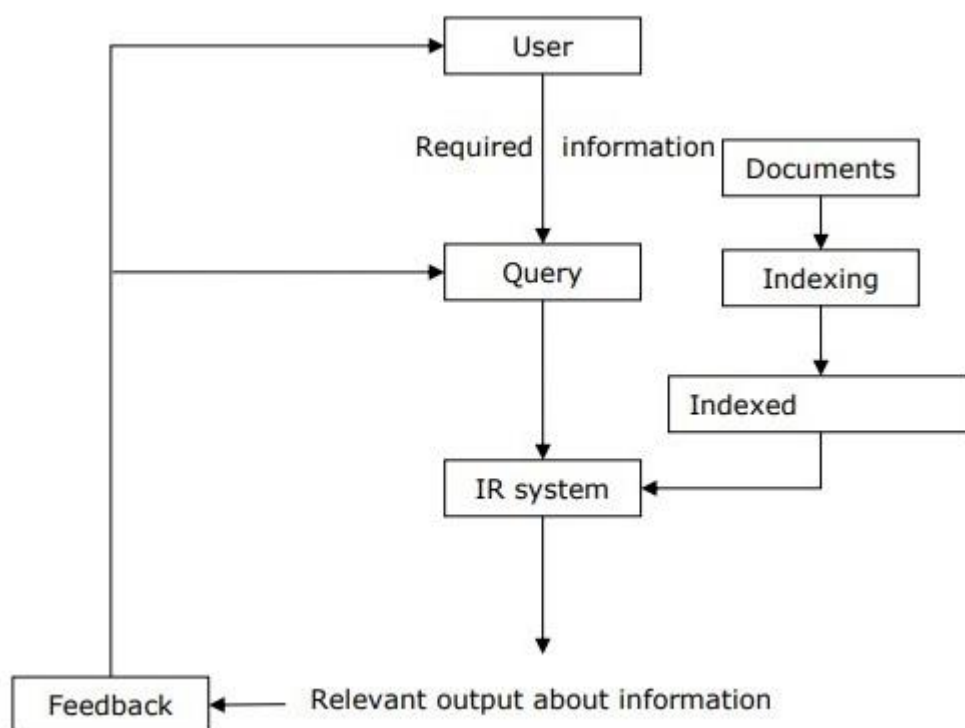
Фрагменты слов

Иногда мы произносим предложения с более мелкими фрагментами слов. Например, **wwha-what is the time?** Здесь слова **w-wha** фрагменты слова.

Поиск информации (Information Retrieval)

Информационный поиск (IR, Information Retrieval) можно определить как программу, которая занимается организацией, хранением, поиском и оценкой информации из репозитория документов, в частности текстовой информации. Система помогает пользователям найти нужную им информацию, но не дает явных ответов на вопросы. Он сообщает о существовании и местонахождении документов, которые могут содержать требуемую информацию. Документы, удовлетворяющие требованиям пользователя, называются соответствующими документами. Совершенная IR-система найдет только нужные документы.

С помощью следующей диаграммы мы можем понять процесс поиска информации (IR):



Из приведенной выше диаграммы ясно, что пользователю, которому нужна информация, придется сформулировать запрос в форме запроса на естественном языке. Затем IR-система ответит, извлекая соответствующий вывод в форме документов о требуемой информации.

Классическая проблема в системе информационного поиска (ИИ)

Основная цель IR-исследования - разработать модель для извлечения информации из хранилищ документов. Здесь мы собираемся обсудить классическую проблему, названную **ad-hoc retrieval problem**, относящиеся к ИК-системе.

При произвольном поиске пользователь должен ввести запрос на естественном языке, который описывает требуемую информацию. Затем IR-система вернет необходимые документы, связанные с желаемой информацией. Например, предположим, что мы ищем что-то в Интернете, и он дает несколько точных страниц, которые соответствуют нашему требованию, но могут быть и некоторые нерелевантные страницы. Это связано с проблемой специального поиска.

Аспекты специального поиска

Ниже приведены некоторые аспекты специального поиска, которые рассматриваются в исследовании IR.

- Как пользователи с помощью обратной связи по релевантности могут улучшить исходную формулировку запроса?
- Как реализовать объединение баз данных, т.е. как результаты из разных текстовых баз данных могут быть объединены в один набор результатов?
- Как обращаться с частично поврежденными данными? Какие модели подходят для таких же?

Модель поиска информации (IR)

Математически модели используются во многих научных областях, имея целью понять некоторые явления в реальном мире. Модель поиска информации предсказывает и объясняет, что пользователь найдет в соответствии с заданным запросом. Модель IR - это в основном шаблон, который определяет вышеупомянутые аспекты процедуры поиска и состоит из следующего:

- Макет для документов.
- Модель для запросов.
- Функция сопоставления, которая сравнивает запросы с документами.

Математическая модель поиска состоит из -

D - Представления для документов.

R - Представление для запросов.

F - Каркас моделирования для D, Q вместе с отношениями между ними.

R (q,di)- Функция подобия, которая упорядочивает документы по запросу. Его еще называют рейтингом.

Типы модели поиска информации (IR)

Модель информационной модели (IR) можно разделить на следующие три модели:

Классическая IR-модель

Это самая простая и легкая в реализации модель IR. Эта модель основана на математических знаниях, которые также были легко распознаны и поняты. Boolean, Vector и Probabilistic - это три классические модели IR.

Неклассическая IR-модель

Это полностью противоположно классической IR-модели. Такие модели IR основаны на принципах, отличных от подобия, вероятности, булевых операций. Информационно-логическая модель, модель теории ситуаций и модели взаимодействия являются примерами неклассической IR-модели.

Альтернативная IR-модель

Это усовершенствование классической IR-модели с использованием некоторых специфических методов из других областей. Кластерная модель, нечеткая модель и модели скрытого семантического индексирования (LSI, Latent Dirichlet allocation – Латентное размещение Дирихле) являются примером альтернативной модели IR.

Конструктивные особенности информационно-поисковых (ИИ) систем

Давайте теперь узнаем об особенностях конструкции ИК-систем

Инвертированный индекс

Первичная структура данных большинства IR-систем имеет форму инвертированного индекса. Мы можем определить инвертированный индекс как структуру данных, которая перечисляет для каждого слова все документы, которые его содержат, и частоту появления в документе. Это упрощает поиск «совпадений» слова запроса.

Удаление стоп-слов

Стоп-слова - это часто используемые слова, которые вряд ли будут полезны для поиска. У них меньший смысловой вес. Все такие слова находятся в списке, называемом стоп-листом. Например, артикли «a», «an», «the» и такие предлоги, как «in», «of», «for», «at» и т. Д. Являются примерами стоп-слов. Размер инвертированного индекса можно значительно уменьшить с помощью стоп-листа. По закону Ципфа стоп-лист, состоящий из нескольких десятков слов, уменьшает размер инвертированного индекса почти вдвое. С другой стороны, иногда удаление стоп-слова может привести к удалению термина, который полезен для поиска. Например, если мы удалим букву «A» из «Витамин А», это не будет иметь никакого значения.

Стемминг

Стемминг, упрощенная форма морфологического анализа, представляет собой эвристический процесс извлечения базовой формы слов путем отсечения концов слов. Например, слова «смеяться», «смеется», «смеяться» будут связаны с корнем слова «смеяться».

В наших последующих разделах мы обсудим некоторые важные и полезные модели IR.

Булева модель

Это самая старая модель поиска информации (IR). Модель основана на теории множеств и булевой алгебре, где документы представляют собой наборы терминов, а запросы - это логические выражения для терминов. Булеву модель можно определить как -

- **D** - Набор слов, т. Е. Терминов индексации, присутствующих в документе. Здесь каждый член либо присутствует (1), либо отсутствует (0).
 - **Q** - Логическое выражение, где термины - это термины индекса, а операторы - логические произведения - И, логическая сумма - ИЛИ и логическая разница - НЕ
 - **F** - Булева алгебра над наборами терминов, а также над наборами документов
- Если мы говорим об обратной связи по релевантности, то в логической модели IR прогноз релевантности можно определить следующим образом:
- **R** - Документ считается релевантным выражению запроса тогда и только тогда, когда он удовлетворяет выражению запроса как -

$$((v) \wedge \wedge \sim h)$$

Мы можем объяснить эту модель термином запроса как недвусмысленное определение набора документов.

Например, термин запроса **“economic”** определяет набор документов, которые индексируются термином **“economic”**.

Итак, каков будет результат после объединения терминов с логическим оператором AND? Он будет определять набор документов, который меньше или равен наборам документов любого из отдельных терминов. Например, запрос с условиями **“social”** и **“economic”** создаст набор документов, проиндексированных с обоими условиями. Другими словами, документ установлен на пересечении обоих наборов.

Итак, каков будет результат после объединения терминов с помощью логического оператора ИЛИ? Он будет определять набор документов, который больше или равен наборам документов любого из отдельных терминов. Например, запрос с условиями **“social”** или же **“economic”** создаст набор документов, которые проиндексированы с помощью термина **“social”** или же **“economic”**. Другими словами, набор документов представляет собой объединение обоих наборов.

Преимущества булевого режима

Преимущества булевой модели заключаются в следующем:

- Самая простая модель, в основе которой лежат наборы.
- Легко понять и реализовать.
- Он извлекает только точные совпадения
- Это дает пользователю ощущение контроля над системой.

Недостатки булевой модели

Недостатки булевой модели следующие:

- Функция подобия модели является логической. Следовательно, частичных совпадений не будет. Это может раздражать пользователей.
- В этой модели использование логического оператора имеет гораздо большее влияние, чем критическое слово.
- Язык запросов выразительный, но и сложный.
- Нет ранжирования найденных документов.

Векторная модель пространства

В связи с указанными выше недостатками булевой модели Джерард Солтон и его коллеги предложили модель, основанную на критерии подобия Луна. Критерий сходства, сформулированный Луном, гласит: «Чем больше двух представлений согласовано в данных элементах и их распределении, тем выше будет вероятность того, что они представляют аналогичную информацию».

Чтобы лучше понять модель векторного пространства, обратите внимание на следующие важные моменты:

- Индексные представления (документы) и запросы рассматриваются как векторы, встроенные в евклидово пространство большой размерности.
- Мерой сходства вектора документа с вектором запроса обычно является косинус угла между ними.

Формула измерения косинусного сходства

Косинус - это нормализованное скалярное произведение, которое можно вычислить с помощью следующей формулы –

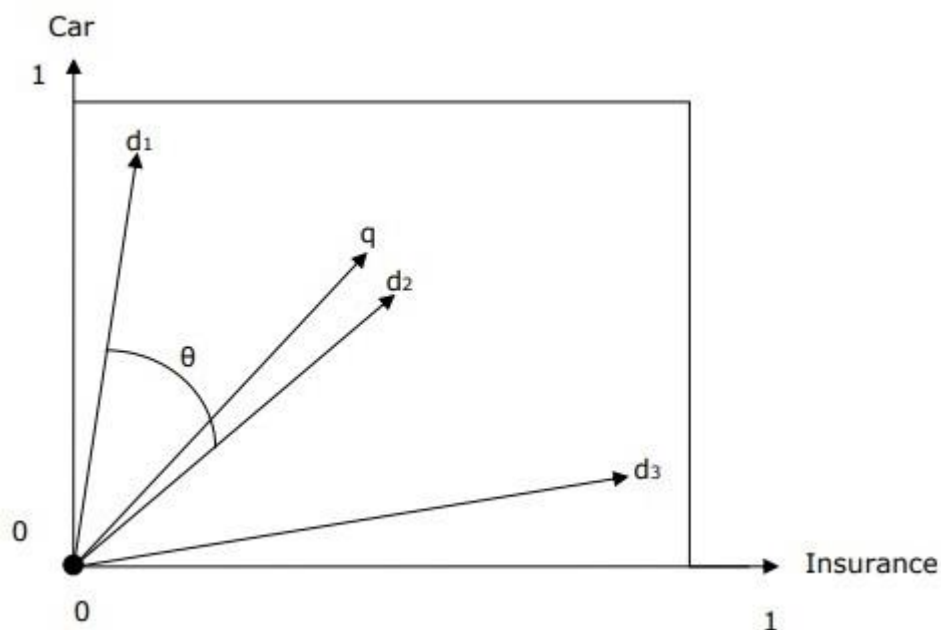
$$\text{Оценка } \text{lgrou} \text{ vecd } \text{vecq } \text{rgrou} = \frac{\sum_{k=1}^m d_k \cdot q_k}{\sqrt{\sum_{k=1}^m d_k^2} \cdot \sqrt{\sum_{k=1}^m q_k^2}}$$

$$\text{Оценка } \text{lgrou} \text{ vecd } \text{vecq } \text{rgrou} = 1 \text{ when } d = q$$

$$\text{Оценка } \text{lgrou} \text{ vecd } \text{vecq } \text{rgrou} = 0 \text{ when } d \text{ and } q \text{ share no items}$$

Представление векторного пространства с запросом и документом

Запрос и документы представлены двумерным векторным пространством. Условия **car** и **insurance**. В векторном пространстве есть один запрос и три документа.



Документом, получившим наивысший рейтинг в терминах «автомобиль» и «страхование», будет документ **d₂** потому что угол между **q** и **d₂** самый маленький. Причина этого в том, что и концепция автомобиля, и страховка являются заметными в **d₂** и, следовательно, имеют большой вес. На другой стороне, **d₁** и **d₃** также упомянуты оба термина, но в каждом случае один из них не является центральным термином в документе.

Взвешивание терминов

Взвешивание терминов означает веса членов в векторном пространстве. Чем выше вес члена, тем сильнее влияние этого члена на косинус. Более важным элементам модели следует присвоить больший вес. Теперь возникает вопрос, как это смоделировать.

Один из способов сделать это - посчитать слова в документе как его вес термина. Однако, как вы думаете, будет ли это эффективным методом?

Другой метод, который более эффективен, - использовать **term frequency (tf_{ij})**, **document frequency (df_i)** и **collection frequency (cf_i)**.

Частота термина (tf_{ij})

Его можно определить как количество вхождений **w_i** в **d_j**. Информация, которая фиксируется частотой термина, - это то, насколько слово выделяется в данном документе или, другими словами, мы можем сказать, что чем выше частота термина, тем больше это слово является хорошим описанием содержания этого документа.

Частота документа (df_i)

Его можно определить как общее количество документов в коллекции, в которой встречается **w_i**. Это показатель информативности. Семантически ориентированные слова будут встречаться в документе несколько раз, в отличие от семантически несфокусированных слов.

Частота сбора (cf_i)

Его можно определить как общее количество вхождений **w_i** в коллекции.

Математически $df_i \leq cf_i \text{ и } \sum_j tf_{ij} = cf_i$

Формы взвешивания документов по частоте

Давайте теперь узнаем о различных формах частотного взвешивания документа. Формы описаны ниже -

Коэффициент частоты термина

Также классифицируется как коэффициент частоты, что означает, если термин **t** часто появляется в документе, тогда запрос, содержащий **t**, должен получить этот документ. Мы можем комбинировать слова **term frequency (tf_{ij})** и **document frequency (df_i)** в один вес следующим образом -

$$weight\ left(i, j\ right) = \begin{cases} (1 + \log(tf_{ij})) \log \frac{N}{df_i} & \text{if } tf_{i,j} \geq 1 \\ 0 & \text{if } tf_{i,j} = 0 \end{cases}$$

где N - общее количество документов.

Обратная частота документов (idf)

Это еще одна форма частотного взвешивания документа, часто называемая взвешиванием idf или обратным взвешиванием частоты документа. Важным моментом взвешивания idf является то, что редкость термина в коллекции является мерой его важности, а важность обратно пропорциональна частоте встречаемости.

Математически,

$$idf_t = \log \left(1 + \frac{N}{n_t} \right)$$

$$idf_t = \log \left(\frac{N}{n_t} \right)$$

где

N = документы в коллекции

n_t = документы, содержащие термин t

Улучшение пользовательских запросов

Основной целью любой системы поиска информации должна быть точность - создание соответствующих документов в соответствии с требованиями пользователя. Однако здесь возникает вопрос, как мы можем улучшить результат, улучшив стиль формирования запросов пользователя. Конечно, вывод любой IR-системы зависит от запроса пользователя, и хорошо отформатированный запрос даст более точные результаты. Пользователь может улучшить свой запрос с помощью **relevance feedback**, важный аспект любой IR-модели.

Отзыв о релевантности

В качестве обратной связи по релевантности используются выходные данные, изначально возвращенные заданным запросом. Этот исходный вывод можно использовать для сбора информации о пользователе и определения того, подходит ли этот вывод для выполнения нового запроса. Отзывы можно классифицировать следующим образом -

Явная обратная связь

Его можно определить как обратную связь, полученную от значимых оценщиков. Эти оценщики также укажут релевантность документа, полученного в результате запроса. Чтобы повысить производительность поиска запроса, информацию обратной связи о релевантности необходимо интерполировать с исходным запросом.

Оценщики или другие пользователи системы могут явно указать релевантность, используя следующие системы релевантности:

- **Binary relevance system** - Эта система обратной связи по релевантности указывает, что документ либо релевантен (1), либо нерелевантен (0) для данного запроса.
- **Graded relevance system**- Система обратной связи с оцененной релевантностью указывает на релевантность документа для данного запроса на основе оценки с использованием цифр, букв или описаний. Описание может быть таким, как «нерелевантно», «в некоторой степени актуально», «очень актуально» или «актуально».

Неявная обратная связь

Это обратная связь, вытекающая из поведения пользователя. Поведение включает в себя время, в течение которого пользователь просматривает документ, какой документ выбран для просмотра, а какой нет, действия по просмотру и прокрутке страниц и т. Д. Одним из лучших примеров неявной обратной связи является **dwelling time**, который является мерой того, сколько времени пользователь тратит на просмотр страницы, на которую указывает ссылка в результатах поиска.

Псевдо-обратная связь

Это также называется слепой обратной связью. Это метод автоматического локального анализа. Ручная часть обратной связи по релевантности автоматизирована с помощью обратной связи по псевдорелевантности, так что пользователь получает улучшенную производительность поиска без расширенного взаимодействия. Основное преимущество этой системы обратной связи состоит в том, что она не требует оценщиков, как в системе обратной связи с явной релевантностью.

Рассмотрите следующие шаги для реализации этой обратной связи -

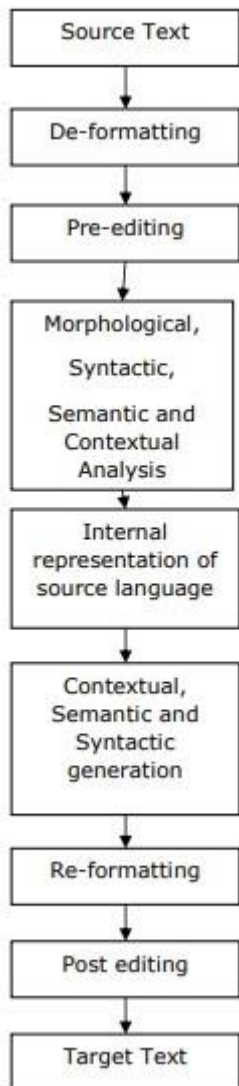
- **Step 1**- Во-первых, результат, возвращаемый исходным запросом, должен восприниматься как релевантный результат. Диапазон релевантных результатов должен входить в топ 10-50 результатов.
- **Step 2** - Теперь выберите 20-30 лучших терминов из документов, используя, например, частоту термина (tf) - вес, обратную частоту документа (idf).
- **Step 3**- Добавьте эти термины в запрос и сопоставьте возвращенные документы. Затем верните наиболее подходящие документы.

Приложения NLP

Обработка естественного языка (NLP) - это развивающаяся технология, которая создает различные формы ИИ, которые мы видим в настоящее время, и ее использование для создания бесшовного, а также интерактивного интерфейса между людьми и машинами будет по-прежнему оставаться главным приоритетом для сегодняшнего и завтрашнего дня. когнитивные приложения. Здесь мы собираемся обсудить некоторые из очень полезных приложений NLP.

Машинный перевод

Машинный перевод (МП), процесс перевода одного исходного языка или текста на другой язык, является одним из наиболее важных приложений NLP. Мы можем понять процесс машинного перевода с помощью следующей блок-схемы -



Типы систем машинного перевода

Существуют разные типы систем машинного перевода. Давайте посмотрим, какие бывают разные типы.

Двухязычная система машинного перевода

Двухязычные системы машинного перевода выполняют переводы между двумя конкретными языками.

Многоязычная система машинного перевода

Многоязычные системы машинного перевода выполняют переводы между любой парой языков. Они могут быть как однонаправленными, так и двунаправленными.

Подходы к машинному переводу (МП, MT - Machine Translation)

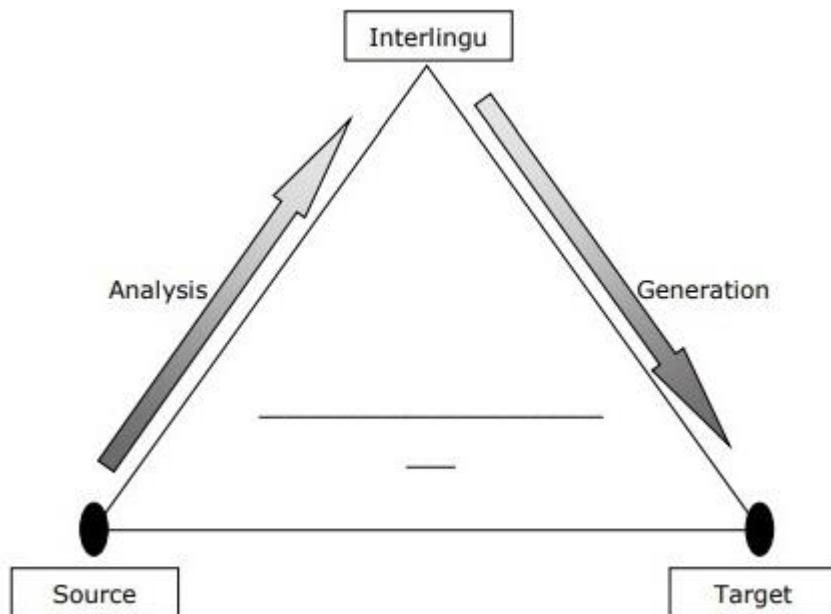
Давайте теперь узнаем о важных подходах к машинному переводу. Подходы к МП следующие:

Прямой подход

Это менее популярный, но самый старый подход МП. Системы, использующие этот подход, способны переводить SL (исходный язык) непосредственно на TL (целевой язык). Такие системы являются двухязычными и однонаправленными по своей природе.

Интерлингва подход

Системы, использующие подход интерлингва, переводят SL на промежуточный язык, называемый интерлингва (IL), а затем переводят IL в TL. Подход Интерлингва можно понять с помощью следующей пирамиды МП



Трансферный подход

Этот подход включает три этапа.

- На первом этапе тексты исходного языка (SL) преобразуются в абстрактные SL-ориентированные представления.
- На втором этапе представления, ориентированные на SL, преобразуются в представления, ориентированные на эквивалентный целевой язык (TL).
- На третьем этапе формируется окончательный текст.

Эмпирический подход МП

Это новый подход к МП. По сути, он использует большой объем необработанных данных в виде параллельных корпусов. Необработанные данные состоят из текста и их переводов. Методы машинного перевода на основе аналогов, примеров и памяти используют эмпирический подход МП.

Борьба со спамом

Одна из самых распространенных проблем в наши дни - это нежелательные электронные письма. Это делает спам-фильтры еще более важными, потому что это первая линия защиты от этой проблемы.

Система фильтрации спама может быть разработана с использованием функций NLP с учетом основных ложноположительных и ложноотрицательных проблем.

Существующие модели NLP для фильтрации спама

Ниже приведены некоторые существующие модели NLP для фильтрации спама.

Моделирование N-грамм

Модель N-Gram - это N-символьный фрагмент более длинной строки. В этой модели N-граммы нескольких разной длины одновременно используются при обработке и обнаружении спама.

Stemming слов

Спамеры, генераторы спам-писем, обычно изменяют один или несколько символов атакующих слов в своих спам-сообщениях, чтобы они могли нарушить фильтры спама на основе содержимого. Вот почему мы можем сказать, что фильтры на основе содержимого бесполезны, если они не могут понять значение слов или фраз в электронном письме. Чтобы устранить такие проблемы при фильтрации спама, разработана основанная на правилах техника выделения слов, которая может сопоставить слова, которые похожи и звучат одинаково.

Байесовская классификация

Теперь это стало широко используемой технологией для фильтрации спама. Распространенность слов в электронном письме измеряется по сравнению с их типичным появлением в базе данных нежелательных (спам) и законных (любительских) сообщений электронной почты с помощью статистических методов.

Автоматическое обобщение

В эту цифровую эпоху самое ценное - это данные, или, можно сказать, информация. Однако действительно ли мы получаем полезную информацию и требуемое количество информации? Ответ - «НЕТ», потому что информация перегружена, и наш доступ к знаниям и информации намного превышает наши возможности ее понять. Мы остро нуждаемся в автоматическом резюмировании текста и информации, потому что поток информации через Интернет не прекратится.

Резюмирование текста можно определить как метод создания короткого и точного резюме более длинных текстовых документов. Автоматическое суммирование текста поможет нам получить актуальную информацию за меньшее время. Обработка естественного языка (NLP) играет важную роль в разработке автоматического резюмирования текста.

Вопрос-ответ

Еще одно основное применение обработки естественного языка (NLP) - это вопросы и ответы. Поисковые системы предоставляют нам информацию о мире, но им все еще не хватает ответов на вопросы, задаваемые людьми на их естественном языке. У нас есть крупные технологические компании, такие как Google, которые тоже работают в этом направлении.

Ответы на вопросы - это дисциплина компьютерных наук в области ИИ и NLP. Он ориентирован на создание систем, которые автоматически отвечают на вопросы, задаваемые людьми на их естественном языке. Компьютерная система, понимающая естественный язык, обладает способностью программной системы переводить предложения, написанные людьми, во внутреннее представление, чтобы система могла генерировать правильные ответы. Точные ответы можно получить, выполнив синтаксический и семантический анализ вопросов. Лексический пробел, двусмысленность и многоязычие - вот некоторые из проблем, с которыми сталкивается NLP при построении хорошей системы ответов на вопросы.

Анализ настроений

Еще одно важное приложение обработки естественного языка (NLP) - анализ тональности. Как следует из названия, анализ настроений используется для определения настроений среди нескольких сообщений. Он также используется для определения настроения, при котором эмоции не выражены явно. Компании используют анализ настроений, приложение обработки естественного языка (NLP), чтобы определять мнение и настроения своих клиентов в Интернете. Это поможет компаниям понять, что их клиенты думают о продуктах и услугах. Компании могут судить о своей общей репутации по сообщениям клиентов с помощью анализа настроений. Таким образом, мы можем сказать, что помимо определения простой полярности, анализ настроений понимает настроения в контексте, чтобы помочь нам лучше понять, что стоит за выраженным мнением.

Обработка естественного языка - Python

В этой главе мы узнаем о языковой обработке с использованием Python.

Следующие функции отличают Python от других языков:

- **Python is interpreted** - Нам не нужно компилировать нашу программу Python перед ее выполнением, потому что интерпретатор обрабатывает Python во время выполнения.
- **Interactive** - Мы можем напрямую взаимодействовать с интерпретатором для написания наших программ Python.
- **Object-oriented** - Python является объектно-ориентированным по своей природе, и на этом языке легче писать программы, потому что с помощью этой техники программирования он инкапсулирует код внутри объектов.
- **Beginner can easily learn** - Python также называют языком для начинающих, потому что он очень прост для понимания и поддерживает разработку широкого спектра приложений.

Предпосылки

Последняя выпущенная версия Python 3 - Python 3.7.1 доступна для Windows, Mac OS и большинства разновидностей ОС Linux.

- Для Windows мы можем перейти по ссылке www.python.org/downloads/windows/, чтобы загрузить и установить Python.
- Для MAC OS мы можем использовать ссылку www.python.org/downloads/mac-osx/.
- В случае Linux разные версии Linux используют разные менеджеры пакетов для установки новых пакетов.

- Например, чтобы установить Python 3 в Ubuntu Linux, мы можем использовать следующую команду из терминала -

```
$sudo apt-get install python3-minimal
```

Чтобы узнать больше о программировании на Python, прочтите базовое руководство по [Python 3](#) - [Python 3](#)

Начало работы с NLTK

Мы будем использовать библиотеку Python NLTK (Natural Language Toolkit) для анализа текста на английском языке. Набор инструментов для естественного языка (NLTK) - это набор библиотек Python, разработанных специально для идентификации и маркировки частей речи, встречающихся в тексте на естественном языке, таком как английский.

Установка NLTK

Перед тем как начать использовать NLTK, нам необходимо его установить. С помощью следующей команды мы можем установить его в нашей среде Python -

```
pip install nltk
```

Если мы используем Anaconda, то пакет Conda для NLTK можно создать с помощью следующей команды -

```
conda install -c anaconda nltk
```

Скачивание данных NLTK

После установки NLTK еще одной важной задачей является загрузка его предустановленных текстовых репозиторий, чтобы его можно было легко использовать. Однако перед этим нам нужно импортировать NLTK так же, как мы импортируем любой другой модуль Python. Следующая команда поможет нам импортировать NLTK -

```
import nltk
```

Теперь загрузите данные NLTK с помощью следующей команды -

```
nltk.download()
```

Установка всех доступных пакетов NLTK займет некоторое время.

Другие необходимые пакеты

Некоторые другие пакеты Python, например **gensim** и **pattern** также очень необходимы для анализа текста, а также для создания приложений обработки естественного языка с использованием NLTK. пакеты могут быть установлены, как показано ниже -

Gensim

gensim - это надежная библиотека семантического моделирования, которую можно использовать во многих приложениях. Мы можем установить его с помощью следующей команды -

```
pip install gensim
```

Паттерн

Его можно использовать для изготовления правильно работающего **gensim** пакета. Следующая команда помогает в установке шаблона -

```
pip install pattern
```

Токенизация

Токенизацию можно определить как процесс разбиения данного текста на более мелкие единицы, называемые токенами. Жетонами могут быть слова, числа или знаки препинания. Это также можно назвать сегментацией слов.

пример

Input - Кровать и стул - это виды мебели.

Output:

Bed	and	chair	are	types	of	furniture
-----	-----	-------	-----	-------	----	-----------

У нас есть разные пакеты для токенизации, предоставляемые NLTK. Мы можем использовать эти пакеты в зависимости от наших требований. Пакеты и детали их установки следующие:

Пакет `sent_tokenize`

Этот пакет можно использовать для разделения вводимого текста на предложения. Мы можем импортировать его, используя следующую команду -

```
from nltk.tokenize import sent_tokenize
```

Пакет `word_tokenize`

Этот пакет можно использовать для разделения вводимого текста на слова. Мы можем импортировать его, используя следующую команду -

```
from nltk.tokenize import word_tokenize
```

Пакет `WordPunctTokenizer`

Этот пакет можно использовать для разделения вводимого текста на слова и знаки препинания. Мы можем импортировать его, используя следующую команду -

```
from nltk.tokenize import WordPuncttokenizer
```

Стемминг

По грамматическим причинам язык включает множество вариаций. Вариации в том смысле, что язык, английский, а также другие языки, имеют разные формы слова. Например, такие слова, как **democracy**, **democratic**, и **democratization**. Для проектов машинного обучения очень важно, чтобы машины понимали, что эти разные слова, как указано выше, имеют одинаковую базовую форму. Вот почему при анализе текста очень полезно извлекать базовые формы слов.

Выделение корней - это эвристический процесс, помогающий извлекать базовые формы слов путем отсечения их концов.

Различные пакеты для стемминга, предоставляемые модулем NLTK, следующие:

Пакет `PorterStemmer`

Этот пакет определения корней использует алгоритм Портера для извлечения базовой формы слов. С помощью следующей команды мы можем импортировать этот пакет -

```
from nltk.stem.porter import PorterStemmer
```

Например, **'write'** будет выходом слова **'writing'** задано в качестве входных данных для этого стеммера.

Пакет `LancasterStemmer`

Этот пакет определения корней использует алгоритм Ланкастера для извлечения базовой формы слов. С помощью следующей команды мы можем импортировать этот пакет -

```
from nltk.stem.lancaster import LancasterStemmer
```

Например, **'writ'** будет выходом слова **'writing'** задано в качестве входных данных для этого стеммера.

Пакет SnowballStemmer

Этот пакет выделения корней использует алгоритм Snowball для извлечения базовой формы слов. С помощью следующей команды мы можем импортировать этот пакет -

```
from nltk.stem.snowball import SnowballStemmer
```

Например, **'write'** будет выходом слова **'writing'** задано в качестве входных данных для этого стеммера.

Лемматизация

Это еще один способ извлечения базовой формы слов, обычно направленный на удаление флексивных окончаний с помощью словарного и морфологического анализа. После лемматизации основная форма любого слова называется леммой.

Модуль NLTK предоставляет следующий пакет для лемматизации -

Пакет WordNetLemmatizer

Этот пакет извлечет базовую форму слова в зависимости от того, используется ли оно как существительное или как глагол. Для импорта этого пакета можно использовать следующую команду -

```
from nltk.stem import WordNetLemmatizer
```

Подсчет POS-тегов - разбиение

Выявление частей речи (POS) и коротких фраз может быть выполнено с помощью разбиения на части. Это один из важных процессов обработки естественного языка. Поскольку мы знаем о процессе токенизации для создания токенов, разбиение на части фактически предназначено для маркировки этих токенов. Другими словами, мы можем сказать, что можем получить структуру предложения с помощью процесса разбиения на части.

пример

В следующем примере мы реализуем фрагменты существительных и фраз, категорию фрагментов, которые будут находить фрагменты именных фраз в предложении, используя модуль NLTK Python.

Рассмотрим следующие шаги, чтобы реализовать разбиение именных фраз:

Step 1: Chunk grammar definition

На этом этапе нам нужно определить грамматику для фрагментов. Он будет состоять из правил, которым мы должны следовать.

Step 2: Chunk parser creation

Далее нам нужно создать парсер чанков. Он проанализирует грамматику и выдаст результат.

Step 3: The Output

На этом этапе мы получим результат в виде дерева.

Запуск сценария NLP

Начните с импорта пакета NLTK -

```
import nltk
```

Теперь нам нужно определить предложение.

Где,

- DT - определитель
- VBP - это глагол
- JJ - прилагательное
- IN - это предлог
- NN - существительное

```
sentence = [("a", "DT"), ("clever", "JJ"), ("fox", "NN"), ("was", "VBP"),  
            ("jumping", "VBP"), ("over", "IN"), ("the", "DT"), ("wall", "NN")]
```

Далее грамматика должна быть дана в виде регулярного выражения.

```
grammar = "NP:{<DT>?<JJ>*<NN>}"
```

Теперь нам нужно определить синтаксический анализатор для анализа грамматики.

```
parser_chunking = nltk.RegexpParser(grammar)
```

Теперь парсер проанализирует предложение следующим образом:

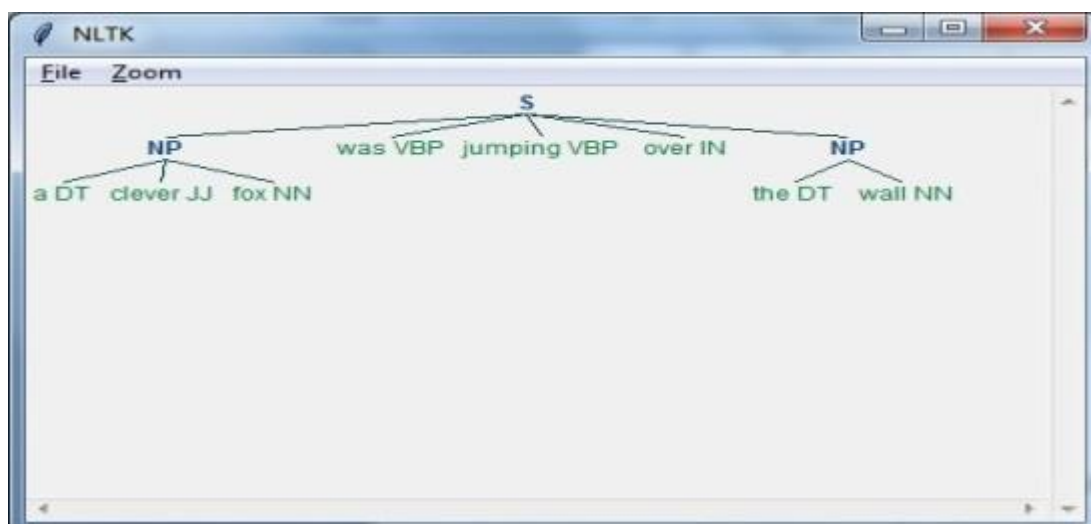
```
parser_chunking.parse(sentence)
```

Затем вывод будет в переменной следующим образом: -

```
Output = parser_chunking.parse(sentence)
```

Теперь следующий код поможет вам нарисовать результат в виде дерева.

```
output.draw()
```



Используемые термины

NLP, Natural Language Processing - обработка естественного языка

AI, Artificial Intelligence – искусственный интеллект, ИИ

MT, Machine Translation – машинный перевод, МП

Morphological Processing – морфологическая обработка

Syntax Analysis – синтаксический анализ

Semantic Analysis – семантический анализ

Pragmatic Analysis – прагматический анализ

RE, Regular Expression – регулярное выражение

FSA, finite-state automaton – конечный автомат

FA, finite automaton – конечный автомат, синоним

FSM, Finite-state machine – конечный автомат, синоним

DFA, deterministic finite automaton – детерминированный конечный автомат

NDFA / NFA, Nondeterministic finite automaton – недетерминированный конечный автомат

Stems – стебли (основа)

Affix – аффикс

Prefix – префикс

Suffix – суффикс

Infix – инфикс

Circumfix – циркумфикс

Parser – синтаксический анализатор

Parsing – синтаксический анализ

CFG, Context-free grammar – контекстно-свободная грамматика, КСГ

Hyponymy – гипонимия

Homonymy – омонимия

Polysemy – полисемия

Synonymy – синонимия

Antonymy – антонимия

FOPL, First-order Predicate Logic - логика предикатов первого порядка

CD, Conceptual dependency – концептуальная зависимость

CG, Conceptual graph - концептуальные графы

WSD, Word sense disambiguation – устранение неоднозначности смысла слов

POS-tagging, или PoS-tagging, или POST, Part-of-speech tagging – тегеры части речи

IR, Information retrieval - информационный поиск

IE, Information Extraction - извлечение информации

Coherent discourse – последовательные (когерентные) суждения

Referring expression – ссылающееся выражение

Referent – референт

Corefer – корефер

Antecedent – antecedent

Anaphora & Anaphoric - Анафора и анафорика

Discourse model – модель дискурса

TBL, transformation based tagging – тегирование на основе преобразования

HMM, hidden Markov model - скрытая марковская модель

Reparandum and repair – репарандум и исправление

Stemming – стемминг, процесс нахождения основы слова для заданного исходного слова